KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Aistė Šimkutė

VIENOS ELEKTRONINIO BALSAVIMO SISTEMOS ANALIZĖ IR MODIFIKAVIMAS

Baigiamasis magistro projektas

Vadovas
Lekt. dr. Kęstutis Lukšys

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

VIENOS ELEKTRONINIO BALSAVIMO SISTEMOS ANALIZĖ IR MODIFIKAVIMAS

Baigiamasis magistro projektas

Taikomoji matematika (621G10003)

Vadovas
Lekt. dr. Kęstutis Lukšys

Recenzentas
Lekt. dr. Aleksejus Michalkovič

Projektą atliko
Aistė Šimkutė

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Matematikos ir gamtos mokslų fakultetas

(Fakultetas)

Aistė Šimkutė

(Studento vardas, pavardė)

Taikomoji matematika (621G10003)

(Studijų programos pavadinimas, kodas)

„Vienos elektroninio balsavimo sistemos analizė ir modifikavimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2016 m. gegužės 19 d.

Kaunas

Patvirtinu, kad mano, Aistės Šimkutės, baigiamasis projektas tema „Vienos elektroninio balsavimo sistemos analizė ir modifikavimas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjusi.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka)	*(parašas)*

Šimkutė, Aistė. Vienos elektroninio balsavimo sistemos analizė ir modifikavimas. Magistro baigiamasis projektas / vadovas lekt. dr. Kęstutis Lukšys; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.
Mokslo kryptis ir sritis: Fiziniai mokslai, matematika.
Reikšminiai žodžiai: *elektroninis balsavimas, sumos unikalumas, diofantinė lygtis.*
Kaunas, 2016. 76 p.

SANTRAUKA

Pagrindinis šio darbo tikslas modifikuoti dviejų pasirinkimų elektroninio balsavimo sistemą į kelių pasirinkimų sistemą. Pasirinkta nagrinėjama sistema paskelbta Cao Gang straipsnyje „An electronic voting scheme based on secure multi-party computation“. Šioje sistemoje balsų, kuriuos atitinka kandidatams priskirti skaičiai, suma ir rezultatas nustatomi panaudojus milijonieriaus problemą. Esant dviem kandidatams ir fiksuotam rinkėjų skaičiui, rezultato suma visada unikali. Modifikuojant sistemą būtina nustatyti reikalavimus kandidatams priskiriamoms reikšmėms, kad ši suma užtikrintų vienareikšmiškumą. Šie reikalavimai nustatyti empiriniu būdu panaudojus sukurtą programinę priemonę.

Taip pat atlikta pasirinktos originalios bei modifikuotos sistemos saugumo analizė. Nustatyta, jog pradinės ir pakeistos sistemos saugumas nesiskiria. Ji užtikrina tinkamumo, unikalumo ir teisingumo reikalavimus. Patikrinamumą garantuoja tik iš dalies. Rinkėjo privatumas užtikrinamas tik tada, kai nustatome papildomas sąlygas rinkėjams išduodamoms reikšmėms balsų maskavimui. Jos privalo būti nemažesnės nei didžiausia kandidatams priskiriama reikšmė. Sistemos tikslumas ir vientisumas nėra užtikrinami.

Modifikuotos sistemos rezultato paieškai milijonieriaus problemos naudoti nebeįmanoma. Tam buvo panaudotas pilno perrinkimo metodas, tiesinės diofantinės lygties sprendimo algoritmas bei sveikųjų skaičių dalyba. Nustatyta, jog diofantinės lygties algoritmas rezultato paieškai nėra tinkamas. O efektyviausias metodas yra sveikųjų skaičių dalyba.

Šimkutė, Aistė. *Analysis and Modification of One Electronic Voting System.* Master's thesis in applied mathematics / supervisor lect. dr. Kęstutis Lukšys. The Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.
Research area and field: Physical sciences, Mathematics
Key words: *electronic voting, unique sum, diofantine equation.*
Kaunas, 2016. 76 p.

SUMMARY

The aim of this paper is to modify two choice electronic voting system to more choices system. Chosen system is published by Cao Gang and called "An electronic voting scheme based on secure multi-party computation". At this system the sum and result is determined by millionaire's problem. The result contains the numbers assigned for candidates. When there are just two choices and fixed number of voters the sum of result is always unique. In modified system there are possibility that sum could be inconclusively. Therefore, we must establish requirements for unique sum. These requirements are found by empirical method using created software.

Security analysis of original and modified systems is performed. Both systems security requirements are the same. The eligibility, uniqueness and fairness are ensured completely. Verifiability is guaranteed partly. The privacy of voter is ensured then we establish some additional requirements for values that are used by conceal vote. Those values must be not less than the biggest value of candidate. Accuracy and integrality can be breached.

We cannot use the millionaire's problem to find a result of election. For that reason there is used brute force search, an algorithm to solve diofantine linear equation and integer division. There is founded that diofantine linear equation algorithm is not eligible for result finding. The most effective method is integer division.

TURINYS

SANTRUMPOS

DDoS – dinaminis paslaugų apribojimas (*angl. dynamic denial of service attacks*). Būdas kenkti serveriui, kad jis nepajėgtų aptarnauti vartotojų užklausų.

DRE – elektroninio įrašymo mašinos (*angl. direct recording electronic*).

FTP – rinkmenų perdavimo protokolas (*angl. file transfer protocol*).

ISO – tarptautinė standartizacijos organizacija (*angl. international standard organization*).

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮŽANGA

Elektroninis balsavimas gali tapti priemone, padarančia balsavimo procesą efektyvesniu ir sukeliančiu daugiau pasitikėjimo. Tinkamai įgyvendintos elektroninio balsavimo sistemos gali padidinti balsavimo biuletenio saugumą, pagreitinti balsavimo rezultatų paiešką ir supaprastinti balsavimo procesą. Tačiau neapdairiai sukurtos ir įgyvendintos elektroninio balsavimo sistemos gali sužlugdyti ar neigiamai paveikti rinkimus taip padidinant rinkėjų nepasitikėjimą jais.

Progresuojant kompiuterijos, komunikacijos bei kriptografijos technologijoms vis daugiau dėmesio skiriama elektroninio balsavimo sistemų vystymui. Šiomis dienomis jau yra sukurta daugybė šių sistemų, tačiau daugelis jų netenkina vienokių ar kitokių balsavimams itin svarbių savybių. Dėl šių priežasčių tik maža dalis pasaulio valstybių naudoja elektroninį balsavimą.

Šiame darbe plačiai pristatoma viena pasirinkta balsavimo sistema. Ji pritaikyta tik dviem galimiems pasirinkimams, tad pagrindinis šio darbo tikslas – pritaikyti pasirinktą elektroninio balsavimo sistemą keliems galimiems pasirinkimams. Šiam tikslui pasiekti keliami tokie uždaviniai:

- Nustatyti modifikuotos sistemos pokyčius, kurie suteiktų galimybę rinktis daugiau nei iš dviejų variantų.
- Parinkti efektyvų algoritmą modifikuotos sistemos balsavimo rezultato nustatymui.
- Ištirti pradinės ir modifikuotos pasirinktos sistemos saugumą.

1. LITERATŪROS APŽVALGA

Šiame skyriuje apžvelgiamos įvairios esamos ar buvusios balsavimo technologijos ir sistemos. Elektroninių balsavimų istorija, jų trūkumai bei pranašumai. Plačiau apžvelgiama Estijos nuotolinio elektroninio balsavimo sistema. Apibūdinami pagrindiniai reikalavimai elektroninio balsavimo sistemoms. Taip pat aprašoma pasirinkta, saugiu daugiašaliu skaičiavimu paremta, elektroninio balsavimo sistema.

1.1. Balsavimo technologijos

Rinkimai – pagrindinė demokratijos priemonė. Pirmieji savo balsų pagalba dalį valdžios bei įstatymų priimdavo graikai daugiau nei prieš 2500 metų. Augant gyventojų skaičiui ir vystantis technologijoms keitėsi ir balsavimo būdai. Galima išskirti kelias skirtingas balsavimo technologijas, tai popieriniai biuleteniai, mechaninės svirtinės mašinos, perforuotos kortelės (*angl. punch-cards*), žymų skaitymas (*angl. marksense*), DRE.

Popierinių biuletenių sistema visiems geriausiai žinoma. Ji naudoja oficialią balsavimo kortelę su atspausdintais visų kandidatų vardais ar kitais galimais pasirinkimais. Rinkėjai nustatytu būdu pažymi savo pasirinkimą ir įmeta biuletenį į tam paskirtą slapto balsavimo dėžę. Ši sistema, priimta 1865 m. Viktorijos valstijoje Australijoje, pirmą kartą panaudota valstijos rinkimuose Niujorke 1889 m. [7]

Mechaninėse svirtinėse mašinose kandidato vardas ar pasirinkimas yra priskirtas konkrečiai juostelei ir svirties padėčiai stačiakampyje svirčių masyve mašinos priekyje. Svirtis įjungia mašiną ir užtraukia rinkimų kabinos užuolaidą. Rinkėjas savo balsą užfiksuoja patraukdamas svirtį ties reikiama juostele. Atitraukus užuolaidą svirtis automatiškai grįžta į pradinę padėtį. Kiekvieną kartą svirčiai atsistačius, prijungtas prie mašinos skaičiavimo ratas padaro vieną dešimtąją apsisukimo. Šis ratas atlieka skaičiavimus pagal poziciją. Kol ratas nepadaro pilno apsisukimo, naudojama pozicija „vienetai“. Po kiekvieno pilno rato apsisukimo pasilieka skaičiavimo pozicija „dešimtys“. Vėliau pasiekiami „šimtai“. Jeigu mechaninės jungtys veikia tinkamai ir skaičiavimo ratas pradžioje nustatytas į nulinę padėtį, galutinis balsų skaičius nustatomas pagal kiekvieno skaičiavimo rato poziciją [5, 7].

Mechaninę svirtinę mašiną išrado Tomas Edisonas, siekdamas užkirsti kelią balsų klastojimui. Pirmasis oficialus tokios mašinos panaudojimas buvo 1892 m. Niujorke. Iki 1930 metų, jos buvo beveik kiekviename dideliame Jungtinių Valstijų mieste, o 1960 m. daugiau nei pusė šalies balsų buvo apskaičiuoti mechaninėmis svirtinėmis mašinomis. Dabar jas pakeitė optinis skenavimas ar DRE balsavimo sistemos [7].

Balsavimai su perforuotomis kortelėmis vykdomi panaudojus korteles bei mažą iškarpinės prietaisą, registruojantį balsus. Rinkėjai išmuša skyles kortelėse (su tam skirtu perforavimo prietaisu) ties kandidatais, kurių nepasirenka. Po balsavimo rinkėjas gali biuletenį patalpinti į balsadėžę arba paduoti į kompiuterinį balsų kortelių prietaisą. Pirmosios perforuotos kortelės ir kompiuterizuotos skaičiavimo mašinos buvo panaudotos 1964 m. Gruzijoje [5].

Žymų skaitymo balsavimo sistemos leidžia rinkėjams užregistruoti savo pasirinkimą užpildant apskritimą, stačiakampį, ovalą ar rodyklę biuletenyje su kandidatų vardais ar kitais pasirinkimais. Tuomet biuletenis įmetamas į balsadėžę arba paduodamas į kompiuterizuotą tabuliavimo prietaisą. Šis parenka tamsiausią žymę kaip balsą, panaudojęs „tamsaus ženklo logiką“. Žymų skaitymo sistema dažniausiai priskiriama optinio skenavimo sistemoms [7].

Balsuojant su DRE galimi pasirinkimai rinkėjui matomi mašinos priekyje. Rinkėjas tiesiogiai pateikia pasirinkimą į elektroninį prietaisą panaudojęs liečiamąjį ekraną, mygtukus ar panašų įrenginį. Rinkėjų balsai saugomi šių mašinų atminties kortelėse ir pridedami prie kitų rinkėjų pasirinkimų [5].

1.2. Elektroniniai balsavimai

Elektroninio balsavimo sistema yra tokia sistema, kurioje rinkėjai identifikuojami, autentifikuojami bei jų balsai surenkami tiesiogiai elektroniniu būdu (internetu ar kitomis ryšio priemonėmis) [1]. Išskiriamos dvi pagrindinės elektroninio balsavimo rūšys [4]:

- E. balsavimas, kurį tiesiogiai prižiūri vyriausybinės ar nepriklausomos balsavimo institucijos (pvz., elektroninėmis balsavimo mašinomis, esančiose nustatytose balsavimo vietose).
- Nuotolinis e. balsavimas, kurio metu rinkėjų balsai surenkami internetu (balsavimas panaudojus kompiuterį, mobilųjį telefoną ar kitas priemones).

1.2.1. Elektroninis balsavimas mašinomis

Elektroninio balsavimo pradžia galima laikyti 1892 m., kai pasirodė pirmosios mechaninės balsavimo mašinos. Įvairios balsų surinkimo ar apskaičiavimo mašinos plačiai naudojamos Jungtinėse Valstijose. 2008 metais dažniausiai naudojama buvo optinio skenavimo sistemos technologija. O paprastų popierinių biuletenių rezultatų be techninių prietaisų apskaičiavimą sudarė tik kiek daugiau nei pusę procento visų rinkimų metodų. Taip pat JAV plačiai naudojama DRE [8].

Europoje elektroninio balsavimo mašinas išbandė, tačiau nusprendė toliau jų nebenaudoti Airija, Italija, Norvegija ir Jungtinė Karalystė. Taip pat šių mašinų naudojimą nutraukė Olandijoje ir Vokietijoje. Tik dvejose Europos šalyse elektroninio balsavimo mašinos yra naudojamos – Belgijoje ir Prancūzijoje, tačiau nė vienoje iš jų nenaudojama visoje valstybėje. Prancūzijoje – tik keliuose miestuose [8].

Elektroninio balsavimo mašinomis buvo atsisakoma dėl lėto jų veikimo, pilnų sistemos gedimų. Atminties kortelės ar kiti e. balsavimo įrenginiai, kuriuose kaupiami rinkėjų balsai turi ribotą atmintį, dėl to dažnai būdavo neįrašomi ir prarandami balsai.

Pietų Amerikoje ir Azijoje elektroninis balsavimas paplitęs labiau nei Europoje ir turi didesnį susidomėjimą apie jo panaudojimą ateityje. Elektroninio balsavimo mašinos naudojamos Brazilijoje, Venesueloje, Indijoje. Visose šių šalių vietovėse [8].

1.2.1.1. Elektroninio balsavimo mašinomis privalumai

Elektroninis balsavimas mašinomis yra pranašesni už rinkimus, naudojant įprastus popierinius biuletenius dėl kelių aspektų [8]:

- Galimybė susidoroti su kompleksiniais rinkimais. Elektroninio balsavimo technologijos dažniausiai nesunkiai susidoroja su sudėtiniais rinkimais. Pavyzdžiui, esant keliems rinkimams vienu metu.
- Prieinamumas. Elektroninio balsavimo technologijos gali turėti įtaką biuletenio prieinamumui. Tokios technologijos labiau patrauklios žmonėms, turintiems didesnį kompiuterinį raštingumą (pvz., jauniems rinkėjams). Taip pat tiems, kuriems yra sunku dalyvauti šiame procese, ypač žmonėms su negalia. Balsavimo mašinos gali būti suprojektuotos pagal neįgalių žmonių poreikį taip, jog jiems nereikėtų kito žmogaus pagalbos. Taip užtikrinant jų balso paslaptį.
- Mažesnis balsavimo punkto personalas. Esant paprastesniam procesui, kai nebereikia išduoti biuletenių ir nebūtina nuolat stebėti balsavimo urnos, atsiranda galimybė sumažinti balsavimo punkto personalo skaičių. Taip pat, panaudojus technologiją, kuri apskaičiuoja rezultatus, gerokai sumažėja darbo balsavimo punkto personalui.
- Neteisingų biuletenių pašalinimas. Kai kuriose šalyse dalis biuletenių būna klaidingai užpildyti ir yra pašalinami. Naudojant elektronines technologijas programinė įranga gali būti sukonfigūruota tik galiojančių ir teisingų balsų įskaičiavimui.
- Rezultato skaičiavimo greitis. Svarbus elektroninių technologijų pranašumas yra tas, jog rezultatas tampa prieinamas iškarto baigus rinkimus be ilgo skaičiavimo proceso.

- Tiksli rezultatų lentelė. Kai rezultatai įrašyti elektroniniu būdu ir perduoti tabuliacijai rinkimų valdymo institucijai, žmogiškoji duomenų įrašymo klaidų tikimybė gerokai sumažėja.

1.2.1.2. Elektroninio balsavimo mašinomis trūkumai

Nors elektroniniai balsavimai mašinomis turi nemažai pranašumų, tačiau galima išskirti ir daug trūkumų [8]:

- Skaidrumo trūkumas. Skaidrumas yra pagrindinė patikimų rinkimų dalis. Balsuojant naudojantis popieriniais biuleteniais stebėtojai žino, kiek kokių biuletenių yra išduota, rinkėjai įmeta savo biuletenį į balsavimo dėžę ir jie būna suskaičiuojami. Balsuojant elektroninio balsavimo technologijomis neįmanoma stebėti kelio, kuriuo rinkėjo balsas keliauja.
- Pasitikėjimas. Skaidrumo trūkumas elektroninio balsavimo technologijose sudaro didelę pasitikėjimo stoką. Rinkimų valdymo institucijos privalo užtikrinti balsavimo proceso skaidrumą. Šis trūkumas lėmė, jog dalyje šalių balsavimas, panaudojus elektronines technologijas, nutrūko. Norėdamos užtikrinti rinkėjų pasitikėjimą rinkimų valdymo institucijos gali įtraukti atsitiktinę rezultatų patikrą ar publikuoti elektroninio balsavimo ir skaičiavimo technologijos šaltinio programinį kodą.
- Rezultato patikra. Balsavimas popieriniais biuleteniais turi didelį pranašumą dėl balsų suskaičiavimo. Esant reikalui, rezultatas gali būti patikrintas perskaičiuojant balsus. Daugelis elektroninio balsavimo mašinų tokios galimybės neturi.
- Balsavimo slaptumas. Pagrindinis tarptautinis rinkimų standartas teigia, jog turėtų būti neįmanoma sužinoti už ką kiekvienas rinkėjas balsavo. Elektroninio balsavimo technologijos gali pažeisti šį reikalavimą. Jeigu rinkėjai fiksuojami stebėtojų, žinoma, kokia tvarka jie balsuoja, tuomet galima nustatyti už ką rinkėjas balsavo. Taip pat, elektroninio balsavimo sistemos, kurios pirmiausia identifikuoja rinkėją dažnai suteikia galimybę susieti balsą su rinkėju.
- Elektroninio balsavimo mašinų nustatymo procedūros. Procedūros, kurios turi būti atliekamos prieš ir po balsavimo, gali būti sunkiai suprantamos pirmininkaujantiems pareigūnams.
- Gedimo pasekmės. Elektroninės balsavimo mašinos gali sugesti prieš ar balsavimo metu. Vadinasi, būtina turėti atsarginį variantą ir užtikrinti, jog nutrūkus balsavimo procesui nebūtų prarasti jau surinkti balsai.
- Saugumas. Skirtingi saugumo išbandymai yra pateikiami lyginant elektroninio balsavimo technologijas su popierinių biuletenių sistema. Pavyzdžiui, elektroninis rezultatų perdavimas tabuliacijai suteikia galimybę sistemos nulaužimui ir suklastotų rezultatų įterpimui.

- Kaina. Elektroninio balsavimo mašinos kainuoja 300 – 5000 dolerių. Daugeliui šalių tai būtų didžiulė investicija. Šioms mašinoms taip pat reikalinga elektra, programinė įranga, kurią būtina prižiūrėti. Jos gali sugesti, o visa tai sudaro nemažus papildomus kaštus.

1.2.2. Nuotolinis elektroninis balsavimas

Balsavimą internetu išbandė ar naudoja žymiai mažiau pasaulio šalių nei elektroninį balsavimą mašinomis. Pirmosios šį būdą išbandė Jungtinės Valstijos 2000 m. Jos ir dar dvi valstybės, Norvegija ir Indija, nuotolinį e. balsavimą vis dar bando. Jungtinė Karalystė šį būdą išbandė ir nusprendė nebenaudoti. Australijoje, Kanadoje, Prancūzijoje ir Šveicarijoje balsavimas internetu naudojamas tik tam tikruose rinkimuose, pavyzdžiui renkant vietinę valdžią. Balsavimo internetu bandymus nutraukė Ispanija ir Olandija. Vienintelė šalis – Estija, visiems rinkimams visoje šalyje naudoja nuotolinį elektroninį balsavimą [8].

Pirminės elektroninio balsavimo internetu sistemos naudotos Jungtinėje Karalystėje ir Prancūzijoje buvo konceptualiai nesudėtingos ir neužtikrino daug saugumo reikalavimų. Rinkėjo autentifikavimo ir patikrinimo metodai pasirodė vėlesnėse sistemose [8].

Daugeliu atveju, balsavimo internetu sistemos sudarytos taip, jog jas būtų galima naudoti iš asmeninio kompiuterio nekontroliuojamoje aplinkoje (pvz. namuose, biuruose, viešuose erdvėse). Visais atvejais, išskyrus Estiją, nuotoliniam balsavimui nereikalinga jokia papildoma techninė įranga. Estijoje reikalingas asmens tapatybės kortelės skaitytuvas [8].

1.2.2.1. Nuotolinio elektroninio balsavimo privalumai ir trūkumai

Daugelyje nuotolinio elektroninio balsavimo sistemų išskiriami keturi pagrindiniai balsavimo etapai: registracijos, rinkėjo patikrinimo, balsų apskaičiavimo ir jų surinkimo bei apdorojimo. Taigi galima išskirti kiekvieno šio etapo privalumus bei trūkumus [9].

Registracija. Šiame etape užtikrinama, jog tik turintys tam teisę rinkėjai gali balsuoti.

Daugeliui rinkėjų registracija internetu yra gerokai patogesnė, ypač žmonėms su negalia ar gyvenantiems užsienyje. Taip pat registracija nuotoliniu būdu suteikia ekonominį efektyvumą, skaidrumą ir kontroliuojamą procesą. O skaitmeninei kartai internetinis prieinamumas prie paslaugos visuomet yra priimtinesnis.

Be suteikiamų privalumų registruojantis internetu egzistuoja nedidelės rizikos. Skaitmeninės atakos gali paveikti sistemos prieinamumą, konfidencialumą ar autentifikavimą. DDoS gali perkrauti serverius ir taip sutrikdyti rinkėjo registraciją. Įsibrovėliai gali perskaityti asmeninę informaciją, pateikti neteisingus duomenis ar pakeisti rinkėjo informaciją.

Norint apsisaugoti nuo DDoS tereikia tinkamai suprojektuoti tinklus stipresniam jų pralaidumui kritiniu metu, pavyzdžiui, prieš registracijos pabaigos terminą. Kriptografija, saugi programinė įranga ir stipri slaptažodžių prieinamumo kontrolė (pvz., įtraukiant biometrinius duomenis, tokius kaip pirštų antspaudai) gali padėti apsisaugoti nuo sistemos įsibrovėlių. Taip pat užtikrinti registracijos saugumą gali padėti ir ne techninės kontrolės priemonės, pavyzdžiui, atsiunčiant rinkėjui registracijos korteles, su kuriomis jis gali patvirtinti informaciją.

Rinkėjo patikrinimas. Šis etapas rinkimų dieną patvirtina rinkėjo tapatybę ir galimybę balsuoti.

Balsuojant internetu patikrinimas turi būti įtrauktas į programinę įrangą. Jeigu viskas vykdoma teisinga, tai e. balsavimas gali neabejotinai tiksliai identifikuoti kiekvieną rinkėją ir pasilikti pačią naujausią informaciją. Tinkamai veikiantis patikrinimas garantuoja, jog tik vienas kiekvieno rinkėjo balsas bus įskaitomas, o rinkėjo tapatybė nustatoma ir autorizuojama realiu laiku.

Skaitmeninės atakos rinkėjo patikrinimo etape gali pažeisti sistemos prieinamumą ir autentifikavimą. Jeigu naudojami skirtingi serveriai rinkėjų patikrinimui ir balsų apskaičiavimui, tuomet patikrinimo serveriai gali tapti atskiru taikiniu DDoS. Įsilaužėliai taip pat gali užimti teisėtų rinkėjų vietą vykdydami informacijos išgavimo atakas, apgaule atskleisdami rinkėjo įgaliojimų duomenis.

Pagrindinis būdas apsaugoti patikrinimo etapo informaciją yra naudoti stiprią slaptažodžių prieinamumo kontrolę, pavyzdžiui su biometriniais duomenimis ar lustinę kortelę ir asmeninį PIN kodą.

Balsavimas. Turi garantuoti, jog balsai yra tikslūs ir anonimiški.

Šiame etape DDoS ir įsibrovėliai sukelia labai didelę sistemos prieinamumo, konfidencialumo ar autentifikavimo pažeidžiamumo riziką. DDoS gali perkrauti serverius ir sustabdyti balsavimus, ypač jei rinkimai vykdomi vieną dieną. Įsibrovėliai gali apsimesti teisėtais rinkėjais ir palikti suklastotus balsus ar stebėti tinklo srautą matant kiekvieno rinkėjo pasirinkimą.

Apsaugoti balsų apskaičiavimo etapą nuo galimų grėsmių galima tokias pačiais metodais kaip ir rinkėjų registracijos etapą. Dėl papildomo saugumo rinkėjai savo kompiuteryje gali naudoti sukonfigūruotą paleidžiamą USB ar CD, kuris uždarytų prieigą kenksmingoms programoms. Taip pat saugumą galėtų padidinti pratęstas balsavimo laiko periodas.

Balsų surinkimas ir apdorojimas. Šiame etape centralizuotai surenkami ir sujungiami balsai iš įvairių balsavimo vietų ir apskaičiuojami rezultatai. Būtina užtikrinti, jog būtų įmanoma atsekti įrašus, jei įvyktų klaida ar reiktų balsų perskaičiavimo.

Nuotolinis elektroninis balsavimas, panaudojus tinklo ryšius, suteikia galimybę sujungti visus balsus bet kokiu atstumu. Taip pat balsai suskaičiuojami automatiškai ir greitai bei esant poreikiui, lengva rezultatus rodyti realiu laiku.

Didelę riziką balsų surinkimo etape sudaro įsibrovėliai, galintys įsilaužti į serverius ir pakeisti jau esamus balsus. Šią riziką gali sumažinti kriptografiniai metodai, saugi programinė įranga ar kiti metodai, apsaugantys nuo balsų klastojimo ar leidžiantys pastebėti, jog duomenys buvo pakeisti.

1.3. Nuotolinio elektroninio balsavimo sistemų reikalavimai

Saugūs ir patikimi elektroniniai balsavimai internetu privalo tenkinti šias pagrindines savybes [2 – 4, 13]:

- Privatumas *(angl. privacy).* Visi balsai yra slapti, niekas negali sužinoti už ką rinkėjas balsavo. Maksimalus elektroninio balsavimo sistemų privatumas pasiekiamas tada, kai rinkėjų anonimiškumas pažeidžiamas tik bendradarbiaujant visoms šalims (rinkėjams ir institucijoms)
- Vientisumas *(angl. integrality).* Visi legalūs balsai turi būti įskaičiuoti teisingai.
- Tinkamumas *(angl. eligibility)*. Tik turintys teisę balsuoti rinkėjai gali atiduoti savo balsą.
- Unikalumas *(angl. uniqueness).* Gali būti užskaitomas tik vienas kiekvieno galinčio balsuoti rinkėjo balsas.
- Patikrinamumas *(angl verifability)*. Balsavimo rezultatas gali būti apskaičiuotas pasibaigus balsavimui ir niekas negali šio rezultato suklastoti. Rinkėjas turi turėti galimybę patikrinti, ar jo balsas yra teisingai įrašytas ir įskaičiuotas į galutinį rezultatą. Galima išskirti dviejų rūšių patikrinamumą – individualų ir universalų. Individualaus patikrinimo atveju tik rinkėjas gali patikrinti, ar jo balsas įskaičiuotas. Universalus patikrinimas po rezultatų paskelbimo suteikia galimybę peržiūrėti, ar visi galiojantys balsai yra įskaičiuoti tinkamai, kiekvienam. Šio reikalavimui užtikrinimui būtina susieti rinkėją su jo balsu, o tai prieštarauja saugumo reikalavimui. Vis dėl to, patikrinamumas itin svarbus rinkėjo pasitikėjimui elektroninio balsavimo sistema.
- Tikslumas *(angl. accuracy)*. Rinkėjo balsas negali būti pakeistas ar neįskaičiuotas. Netinkami balsai turi būti pašalinami. Rinkimų rezultatai turi būti suskaičiuojami vienareikšmiškai. Universalaus patikrinamumo savybė tiesiogiai susijusi su tikslumu.
- Teisingumas *(angl. fairness)*. Niekas negali sužinoti tarpinių balsavimo rezultatų.

1.4. Elektroninio balsavimo sistemų rūšys

Elektroninį balsavimą sudaro konceptualus projektas ir pagrindinė e. balsavimo sistema. Ji yra svarbiausia elektroninio balsavimo dalis užtikrinanti visų būtinų reikalavimų tenkinimą. Daugelis e. balsavimo sistemų naudoja kriptografinius metodus ir principus. Pagal juos elektroninio balsavimo sistemos gali būti išskirtos į tris pagrindines grupes [10]:

- Sistemos paremtos homomorfiniu šifravimu (*angl. homomorphic encryption*).
- Pagrįstos maišymo tinklais (*angl. mixing nets*)..
- Paremtos aklaisiais parašais (*angl. blind signatures*).

1.4.1. E. balsavimo sistemos, paremtos homomorfiniu šifravimu

Elektroninėse balsavimo sistemose pagrįstose homomorfiniu šifravimu visi užšifruoti balsai yra surenkami ir susumuojami. Galiausiai suma visų užšifruotų balsų yra iššifruojama ir galima atkurti rezultatą [3]. Tai galima atlikti dėl homomorfizmo savybių. Matematinis homomorfizmo apibrėžimas:

> ***1.1. Apibrėžimas.*** Grupių $< G; * >$ ir $< G'; \circ >$ atvaizdis $f: < G; * > \rightarrow < G'; \circ >$ vadinamas homomorfizmu, jeigu $f(a * b) = f(a) \circ f(b)$.

Vadinasi, remiantis šiuo apibrėžimu, galima teigti, jog elektroninėse balsavimo sistemose suma visų užšifruotų balsų lygi užšifruotai reikšmių sumai. Toks šifravimo metodas e. balsavimuose naudingas dėl rezultato suradimo neatskleidžiant rinkėjų pasirinktų balsų, taip užtikrinant rinkėjų anonimiškumą [11].

Galima nesunkiai apibūdinti pagrindinius homomorfinio šifravimo principus. Tarkime operatorius $\oplus$ priklauso paprastą tekstą turinčiai grupei, o operatorius $\otimes$ - turinčiai šifruotą tekstą. $E_r(m)$ žymi žinutės m šifravimą panaudojus parametrą r. Šifravimo sistema ($\otimes$,$\oplus$) – homomorfinė, jei duotiems $c_1 = E_{r_1}(m_1)$ ir $c_2 = E_{r_2}(m_2)$ egzistuoja toks r, kad

$$c_1 \otimes c_2 = E_r(m_1 \oplus m_2).$$

Norint atlikti homomorfinį šifravimą galima taikyti ElGamalio algoritmą. Tarkime p, q yra dideli pirminiai skaičiai, tokie, kad $q|p-1$, o G_q yra žiedo $\mathbb{Z}_p^*$ pogrupis. Šioje sistemoje balsai yra $m_1 = g$ ir $m_0 = 1/g$ (taip/ne), kur g yra pogrupio G_q generatorius. Institucijos atsitiktinai sugeneruoja privatųjį raktą s ir apskaičiuoja viešąjį raktą $h = g^s \bmod p$. Viešasis raktas skelbiamas atvirai. Rinkėjas sugeneruoja atsitiktinį skaičių $\alpha \in \mathbb{Z}_p^*$. Tada jo balso forma

$(x_i, y_i) = (g^{\alpha}, h^{\alpha} \cdot G^{b})$, kur $b \in \{1, -1\}$ ir neinteraktyvus galiojimo įrodymas. Po galutinio termino institucijos apskaičiuoja

$$(X, Y) = (\prod_{i=1}^{n} x_i \bmod p, \prod_{i=1}^{n} y_i \bmod p)$$

visiems galiojantiems balsams. Galiausiai institucijos bendrai apskaičiuoja $W \equiv \frac{Y}{X^{s}} \bmod p$ ir gauna $W \equiv G^{T} \bmod p$, kur T yra skirtumas tarp balsų taip ir balsų ne [3].

1.4.2. E. balsavimo sistemos, pagrįstos maišymo tinklais

Pagrindinė maišymo tinklų sistemų idėja yra maišymo metodų panaudojimas. Pavyzdžiui, maišymo tinklai gali paimti įvestus duomenis ir juos sumaišyti. Taigi rezultatai priklauso nuo įvestos informacijos kombinacijos. Toks metodas tinka norint atskirti balsus nuo konkretaus rinkėjo. Kitu atveju, maišymo sistemos, gali būti naudojamos sumaišant visus galimus balsus, kuriuos rinkėjas gali palikti, su rinkėjo pasirinkimu. Taip rinkėjo balsas tampa paslėptas ir saugus [12].

Elektroninio balsavimo sistemą, paremtą maišymo tinklais pirmasis pristatė David Chaum 1983 m. Jis pasiūlė naudoti kelis skirtingus metodus. Kiekvienas jų turi savo įvedimo parametrus ir išveda atitinkamos kombinacijos reikšmes. Taip pat maišymo metodai susieja informaciją tik tarp įvedimo ir išvedimo parametrų. Vis dėl to, jei egzistuoja n maišymo metodų, mažiausiai vienas iš jų išlaiko informaciją tarp įvedimo ir išvedimo reikšmių neatskleistą, taip visa sistema išlaiko slaptumą ir nenuspėjamumą. Tai vienas pagrindinių maišymo tinklais pagrįstų sistemų privalumų, tačiau kartu ir trūkumas. Pavyzdžiui, jei bent vienas metodas nebeveikia, griūva visa sistema.

Maišymo tinklai gali būti išskirti į peršifravimo ir iššifravimo tinklus. Iššifravimo tinklo [21] įvedimo parametrai yra biuleteniai, užšifruoti kiekvieno balsavime dalyvaujančio maišymo tinklo viešuoju raktu. Kai vykdomas iššifravimas, kiekvienas maišymo tinklas pirmiausiai iššifruoja įvestus biuletenius panaudojęs savo privačiuosius raktus, o tada grąžina rezultatą. Kai naudojami peršifravimo maišymo tinklai [22], originalus visų rinkėjų n biuletenis užšifruojamas viešuoju raktu, pavyzdžiui ElGamalio. Peršifravimo maišymo tinkluose iššifravimo ir peršifravimo procesai yra išskirti. Pirmiausia kiekvienas maišymo tinklas peršifruoja jau užšifruotas ElGamalio raktu reikšmes $\{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n\}$ ir grąžina atsitiktinai išrikiuotas reikšmės. Gaunamas naujas rinkinys $\{\varepsilon'_1, \varepsilon'_2, \dots, \varepsilon'_n\}$. Po to vėl veikia maišymo tinklas arba atliekamas iššifravimas. Pasibaigus visiems peršifravimo etapams iššifravimo dalių kvorumas bendrai atkuria ElGamalio privatųjį raktą ir su juo atkuria visus biuletenius [20].

1.4.3. E. balsavimo sistemos, paremtos aklaisiais parašais

Aklųjų parašų idėja buvo pristatyta David Chaum. Sistemos, naudojančios akluosius parašus, gali garantuoti balsų anonimiškumą. Aklieji parašai – tai skaitmeniniai parašai, kurių informacija yra paslepiama prieš pasirašymą.

Galimas aklųjų parašų panaudojimas elektroninio balsavimo sistemoje [11]:

- Institucija turi privatųjį ir viešąjį raktą: privatusis – (n, d), viešasis – (n, e).
- Rinkėjas sugeneruoja atsitiktinę reikšmę r. Tokią, kad r ir n yra reliatyviai pirminiai.

Panaudodamas savo sugeneruotą reikšmę r ir institucijos viešojo rakto reikšmę e, rinkėjas sukuria akląjį balsą x:

$$x = (r^e v) \bmod n.$$

- Institucija pasirašo žinutę, panaudojusi savo privatųjį raktą:

$$t = x^d \bmod n.$$

- Institucija grąžina pasirašytą „balsą“ t rinkėjui.

$$t = x^d \bmod n = (r^e v)^{\,d} \bmod n = (r^{ed} v)^{\,d} \bmod n = r \cdot v^d \bmod n.$$

- Kadangi rinkėjas žino reikšmę r, jis gali ją pašalinti iš pasirašyto balso:

$$s = r^{-1} t = v^d \bmod n.$$

Galiausiai gaunama reikšmė s, pasirašyta institucijos privačiuoju raktu.

Elektroninio balsavimo sistemoje aklieji parašai gali būti panaudojami įvairiai. Pavyzdžiui, prieš balsavimą rinkėjas pirmiausia turi būti identifikuojamas registracijos institucijos. Po sėkmingo patvirtinimo, rinkėjas išsiunčia savo akląjį balsą registracijos institucijai, kuri jį pasirašo. Taigi rinkėjo balsas yra pasirašomas institucijos naudojant aklųjų parašų metodą, patvirtinant, kad rinkėjas turi teisę balsuoti. Dėl aklųjų parašų balso turinys lieka neatskleistas registracijos institucijai. Po to slaptas balsas pasiekia balsavimo instituciją, kuri taip pat turi galimybę jį patikrinti aklųjų parašų pagalba. Tokiu atveju registracijos institucija negali sužinoti už ką rinkėjas balsavo, o balsavimo – kuriam rinkėjui balsas priklauso. [11]

1.5. Estijos elektroninio balsavimo sistema

Estija pirmoji išbandė nuotolinį elektroninį balsavimą ir vienintelė jį naudoja visoje šalyje visiems rinkimams. Valstybėje internetu balsuojama nuo 2005 m., o paskutiniuose rinkimuose šiuo būdu savo balsus atidavė daugiau nei 30 % rinkėjų. Kadangi Estijos sistema naudojama plačiai ir ne vienerius metus, galima ją panagrinėti plačiau.

Svarbi elektroninių balsavimų dalis Estijoje yra lustinės tapatybės kortelės (ID kortelės). Jos turi galimybę vykdyti kriptografines funkcijas. Internetinio balsavimo metu, naudodami ID korteles, rinkėjai yra autentifikuojami bei pasirašo savo balsus. Kiekvienoje kortelėje yra dvi RSA raktų poros. Viena rinkėjo patvirtinimui, kita skaitmeninių parašų sudarymui. Viešieji raktai laikomi ne tik kortelėje, tačiau ir atvirai prieinamoje duomenų bazėje. Kortelė neleidžia atskleisti privačiųjų raktų, taigi visos kriptografinės operacijos atliekamos viduje. Kaip papildoma apsauga, kiekvienas raktas susietas su PIN kodu, kuris įvedamas atliekant visas operacijas. Taip pat rinkėjai autentifikavimui ir pasirašymui gali naudoti telefonus su specialiomis SIM kortelėmis [14, 17].

Elektroniniams balsavimams Estijoje naudojami keturi serveriai: balso nukreipimo, balso saugojimo, prisijungimo ir balsų apskaičiavimo. Balso nukreipimo serveris prieinamas viešai. Jame patikrinama rinkėjo teisė balsuoti. Taip pat šis serveris panaudojamas kaip balsų saugojimo serverio tarpininkas. Balsų saugojimo serveryje talpinami pasirašyti ir užšifruoti balsai internetinio balsavimo metu. Prisijungimo serveryje saugoma prieš tai aprašytų serverių įvykių informacija ir statistika. Balsų apskaičiavimo serveris niekada neprijungiamas prie tinklo ir naudojamas tik paskutiniame balsavimo etape [14, 18].

Internetinę Estijos balsavimo sistemą sudaro trys pagrindiniai protokolai: balsavimo, balso patvirtinimo, balso apskaičiavimo [14, 16, 18].

- Balsavimo procesas. Rinkėjas naudodamas savo ID kortelę ar specialią SIM kortelę prisijungia prie rinkimų serverių autentifikavimui. Serveriuose patikrinama, ar rinkėjas turi teisę balsuoti. Jeigu balsavimo teisę rinkėjas turi, jam išsiunčiama kandidatų aibė C. Rinkėjas pasirenka kandidatą $c \in C$ ir suranda ar sugeneruoja reikalingas reikšmes. Gautą užšifruotą ir pasirašytą reikšmę v rinkėjas išsiunčia į rinkimų serverius. Juose ši reikšmė susiejama su atsitiktiniu skaičiumi x ir siunčiama atgal rinkėjui. Galiausiai rinkėjui grąžinamas QR kodas sudarytas iš x ir r reikšmių.

- Balso patvirtinimo procesas. Patvirtinimo programa nuskenuojamas QR kodas ir reikšmė x siunčiama į rinkimų serverius. Juose surandamas atitinkantis užšifruotas balsas b ir sudaromas visų galimų kandidatų sąrašas C. Ši informacija siunčiama programai, kuri panaudoja r užšifruodama sumodeliuotą balsą už kiekvieną kandidatą ir palygina rezultatus su gautu rezultatu iš serverių. Jeigu atitikmuo randamas, programa parodo atitinkamą kandidatą ir rinkėjas gali pasitikrinti ar jo balsas užskaitytas tikrai tam kandidatui, už kurį jis balsavo. Serveriai leidžia vykdyti patikrinimą tris kartus 30 minučių po balsavimo.

- Balsų apskaičiavimas ir tabuliacija. Pasibaigus internetiniam balsavimo procesui, balsų saugojimo serveris iš naujo patvirtina parašus ir pašalina visus atšauktus ir neteisingus

balsus. Tinkami anonimiški užšifruoti balsai perkeliami į DVD diską, o tada į apskaičiavimo serverį. Jis iš aparatūros saugumo modulio gauna privatųjį raktą, iššifruoja kiekvieną balsą ir susumuoja rezultatus. Šie vėl perkeliami į DVD diską ir sudedami su visų balsavimo stočių rezultatais.

Visų pirma, Estijos elektroninės balsavimo sistemos saugumą pakankamai stipriai pažeidžia jos kūrėjai ir prižiūrėtojai. Jie naudoja kompiuterius, kuriuose yra įvairios programinės įrangos, per kurią galima nesunkiai užkrėsti kompiuterį virusu. Kūrėjai naudoja nesaugius kanalus parsiųsdami reikiamą programinę įrangą, kas taip pat leistų nesunkiai įsibrovėliams pakenkti kompiuteriui. Rezultatų perkėlimui naudojamas USB su rinkimams visiškai nereikalinga informacija. Taip pat egzistuoja ir kiti neatitikimai neapsaugantys rinkėjo informacijos nuo elektroninius rinkimus prižiūrinčių darbuotojų [14].

Tam tikros serverio operacijos yra apsaugotos kriptografijos elementais (pvz., balsas negali būti iššifruotas frontalinio žiniatinklio serverio, nes jis neturi reikiamo privataus rakto). Kitais atvejais serveriais tenka visiškai pasitikėti ir laikyti, jog jie veikia tinkamai. Dėl šios priežasties internetinės balsavimo sistemos yra labai patrauklus taikinys įsibrovėliams. Jie gali pažeisti du pagrindinius Estijos elektroninės balsavimo sistemos komponentus: rinkėjo balsavimo procesą ir serverius. Išpuolius šiose dalyse galima apibūdinti plačiau [14]:

- Išpuolis rinkėjo balsavimo proceso metu. Rinkėjo įrenginys laikomas patikima elektroninio balsavimo dalimi, todėl egzistuoja keli būdai, kaip įsibrovėliai gali pažeisti pakankamą kiekį Estijos rinkėjų kompiuterių, jog tai padarytų įtaką rinkimų rezultatams. Pirmiausia, kenksminga programa gali pakeisti rinkėjo balsą. Ji įsimena PIN kodą, suvedamą rinkėjui balsuojant. Šį procesą iki galo rinkėjas gali atlikti be jokių trikdžių ir tada, kai rinkėjui uždaroma patvirtinimo prieiga (po 30 minučių) ar jis pilnai atlieka visą balsavimo procesą, kenksminga programa patikrina, ar rinkėjo ID kortelė vis dar yra kompiuteryje. Jeigu taip, ji atidaro rinkėjo duomenų kopiją ir naudodama klavišų paspaudimo simuliaciją, pateikia pakeistą balsą. Jeigu kortelės nebėra, programa laukia, kol ji vėl bus įdėta. Kadangi Estijoje ID kortelės naudojamos gana plačiai, didelė tikimybė, jog ji bus panaudota dar nepasibaigus rinkimams ir programa vis tiek galės pakeisti balsą.

Taip pat kenksminga programa gali tiesiogiai paveikti patvirtinimo programą. Vartotojai dažnai susieja savo telefonus su kompiuteriais. Kenksminga programa, veikdama rinkėjo kompiuteryje ir telefone, nustato, už kurį kandidatą rinkėjas balsuoja ir pakeičia jį taip, jog kompiuteryje QR kodas parodytų teisingą variantą. Telefone kenksminga programa veikia taip, jog gaunamas bet kuris variantas, išskyrus teisingą. Kadangi patvirtinimo programa veiksmus

tiek kompiuteryje, tiek telefone priima kaip teisingus, tai leidžia kenksmingai programai kompiuteryje pakeisti balsą nesukeliant įtarimų.

- Išpuolis serveriuose. Nepaisant nustatytų saugos procedūrų, įsibrovėliai, kurie veikia pakankamai anksti gali pažeisti skaičiavimo serverio programinį kodą, kuris įtakoja konfigūracijos procesą, o vėliau ir balsų apskaičiavimą. Steigimo fazės metu, rinkimų darbuotojai patikrina ISO rinkmenos santraukos funkciją parsiųstą FTP pagalba. Jeigu FTP neatitinka kriptografinio vientisumo patikrinimo, galima laikyti, jog ISO yra pažeistas kenksmingos programos. Vis dėl to, santraukos funkciją steigimo proceso metu galima stebėti vaizdo įrašuose, o tai gali sukelti įtarimų, jog įmanoma ją nesunkiai pažeisti. Taip pat įvairiomis kenksmingomis programos nesunkiai galima pakenkti naudojamoms USB bei CD laikmenoms, kurios virusus galėtų perduoti serveriams.

Taigi, galima daryti išvadą, jog Estijos rinkimų sistemoje yra pakankamai daug spragų, kurios neužtikrina visiško sistemos bei rinkėjo saugumo ir jas būtina taisyti. Tačiau nacionalinis estų rinkimų komitetas teigia, jog visi galimi aprašyti pažeidimai yra numatyti projektuojant sistemą ir yra neįmanoma veiksmingai atlikti atakų pakeičiant balsavimo rezultatus. Taip pat komitetas užtikrina, jog naudoja daugybę saugių priemonių ir įrenginių, kurie aptinka įsibrovėlius ar suklastotus rezultatus.

1.6. Elektroninio balsavimo sistema, paremta saugiu daugiašaliu skaičiavimu

Elektroninio balsavimo sistema, paremta saugiu daugiašaliu skaičiavimu paskelbta Cao Gang straipsnyje „An electronic voting sheme based on secure multi-party computation“ [6]. Šią sistemą sudaro trys etapai: rinkėjo registracijos, balsavimo bei balsų surinkimo. Balsavimo procese dalyvauja du atstovai:

- $V{:}V$ yra rinkėjas. Rinkėjas yra pagrindinis įgyvendinamos elektroninio balsavimo sistemos narys. Pasirinktoje sistemoje nustatoma, jog rinkėjų yra n. Taigi visa rinkėjų aibė gali būti nusakoma: $\{V_1, V_2, \dots, V_n\}$.
- $S{:}S$ yra trečios dalies aptarnaujanti institucija. S funkcija – generuoti atsitiktinius skaičius, maskuoti balsus, ir padalinti balsą X kiekvienam rinkėjui. Taip pat S yra atsakingas už rinkėjo autorizaciją balsavimo metu

Šie subjektai gali būti nepatikimi. Jei balsavimo etape jie bandytų sukčiauti, schema negarantuotų, jog sukčiavimas būtų atskleistas.

1.6.1. Rinkėjo registracijos etapas

Kadangi S prisiima atsakomybę už kiekvieno rinkėjo $\{V_1, V_2, \dots, V_n\}$ autentifikavimą, tai kiekvienas rinkėjas V_i privalo užsiregistruoti S. Po V_i registracijos S sugeneruoja atsitiktinį skaičių R_i. Tada aptarnaujanti institucija S apskaičiuoja R pagal formulę:

$$R = n\sum_{i=1}^{n} R_i.$$

Galiausiai, S siunčia R_i ir neužpildytą balsavimo kortelę rinkėjui V_i.

1.6.2. Balsavimo procesas

Rinkėjas gauna balsavimo kortelę su dviem pasirinkimais, kuriems atsitiktinai priskirtos reikšmės $\{x_1, x_2\}$. Kiekvienas rinkėjas V_i pasirenka už ką balsuoja ir savo pasirinktą balsą $X_i \in \{x_1, x_2\}$ atsitiktinai padalija į tiek privačių dalių, kiek yra rinkėjų. Tai gali būti išreikšta:

$$X_i = \sum_{j=1}^{n} X_{ij}.$$

Padalijęs balsą, rinkėjas V_i prideda savo n privačių dalių X_{ij} prie R_i, kurią V_i gauna užsiregistravęs:

$$A_{ij} = X_{ij} + R_i.$$

Tada V_i siunčia A_j kiekvienam kitam rinkėjui V_j ir A_i reikšmę pasilieka neišsiųstą. Po to kiekvienas rinkėjas, gavęs visą informaciją iš likusių rinkėjų (gavęs $n-1$ reikšmę) apskaičiuoja savo turimos (neišsiųstos) ir gautų reikšmių sumą:

$$B_i = \sum_{j=1}^{n} A_{ji} = \sum_{j=1}^{n} \left(X_{ji} + R_j\right).$$

Tuomet V_i siunčia B_i visiems kitiems dalyviams. Galiausiai kiekvienas rinkėjas apskaičiuoja sumą visų gautų duomenų:

$$C = \sum_{j=1}^{n} B_j = \sum_{j=1}^{n}\sum_{i=1}^{n} \left(X_{ij} + R_i\right) = \sum_{j=1}^{n}\sum_{i=1}^{n} X_{ij} + n\sum_{i=1}^{n} R_i = \sum_{j=1}^{n}\sum_{i=1}^{n} X_{ij} + R.$$

Pažymėjus $\sum_{j=1}^{n}\sum_{i=1}^{n} X_{ij} = D$, gauname reiškinį $C = D + R$.

1.6.3. Balsų surinkimo procesas

Rezultatas randamas panaudojus milijonieriaus problemą [20]. S prieš balsavimą turi sugeneruoti ribinę konstantą M. Šią konstantą S siunčia kiekvienam rinkėjui. Kiekvienas rinkėjas V_i ir S gali gauti balsavimo rezultatą. Rinkėjas V_i apskaičiuoja:

$$T = D + R - M.$$

Tada rinkėjas V_i ir aptarnaujanti institucija S veikia kartu. Kiekvienas rinkėjas turi tokią pačią reikšmę T, o aptarnaujanti institucija S turi R. Vykdydamos milijonieriaus problemos protokolą abi šalys nustato, ar $T > R$. Jeigu taip, tai $D - M \geq 0$, ir galima padaryti išvadą, kad $D \geq M$. Kitu atveju, $D \leq M$.

Galiausiai kiekvienas rinkėjas turi gauti tokį patį balsavimo rezultatą.

1.7. Pasirinktos temos aktualumas

Įvairūs elektroniniai balsavimai taikomi gana plačiai. Daugeliui rinkėjų jie gerokai patogesni už įprastus rinkimus, todėl vis dažniau svarstoma galimybė apie jų panaudojimą dar gausiau. Pasirinkta elektroninio balsavimo sistema, paremta saugiu daugiašaliu skaičiavimu, leidžia rinktis tik iš dviejų galimų pasirinkimų. Tačiau realybėje dažniausiai būtina galimybė rinktis bent iš kelių variantų. Taigi šiame darbe siekiama modifikuoti šią elektroninio balsavimo sistemą į kelių pasirinkimų sistemą.

Viena svarbiausių elektroninių balsavimų savybė yra saugumo užtikrinimas, taigi būtina išanalizuoti, ar sistema, paremta saugiu daugiašaliu skaičiavimu, šį reikalavimą tenkina. Taip pat svarbu, jog būtų tenkinami ir kiti pagrindiniai reikalavimai, keliami elektroninėms balsavimo sistemoms.

2. ALGORITMAI IR TYRIMŲ METODAI

Šioje dalyje apibūdinama kaip pasirinkta sistema pasikeičia modifikavus ją į kelių pasirinkimų sistemą. Nustatoma, kurie metodai ar sąlygos originalioje sistemoje nebetinka pakeistojoje. Aprašoma metodika, kurios pagalba buvo ieškoma, kad modifikuota sistema galėtų veikti tinkamai ir efektyviai.

2.1. Pasirinktos sistemos modifikavimas

Modifikuojant dviejų pasirinkimų sistemą į kelių galimų variantų pasikeičia tam tikros sistemos dalys. Pavyzdžiui, pasirinktoje sistemoje, kuris iš dviejų galimų variantų gavo daugiausia balsų galima sužinoti iš reikšmės D. Tarkime, kad pirma reikšmė už kurią balsuojama yra a_1, o antroji lygi a_2. Tuomet

$$D = n_1 \cdot a_1 + n_2 \cdot a_2,$$

čia $n_1 + n_2 = n$. Taigi pirmas variantas surinko n_1 balsų, o antrasis n_2. Šiuo atveju suma visuomet nusakoma vienareikšmiškai, nes vieną iš rinkėjo surinktų balsų skaičių galime išreikšti panaudojus kitą. Pavyzdžiui:

$$n_1 = n - n_2.$$

Tokiu atveju $D = (n - n_2) \cdot a_1 + n_2 \cdot a_2 = n \cdot a_1 - n_2(a_1 - a_2)$. Kadangi kandidatams priskiriamas reikšmes a_1 ir a_2 bei visą rinkėjų skaičių n galime laikyti konstantomis, gauname, jog suma D priklauso tik nuo vieno kintamojo. O keičiantis tik vienai reikšmei sumą visuomet gausime unikalią.

Pritaikant pasirinktą sistemą keliems galimiems pasirinkimams paliekame tą patį sumos principą, tik padidiname galimų pasirinkimų aibę. Taigi tokiu atveju gauname:

$$D = n_1 \cdot a_1 + n_2 \cdot a_2 + \cdots + n_k \cdot a_k$$

(k – kandidatų (variantų) skaičius). Ši suma ne visuomet gali būti nusakoma vienareikšmiškai. Pavyzdžiui, jeigu yra šeši rinkėjai ir trys kandidatai turime sumą:

$$D = n_1 \cdot a_1 + n_2 \cdot a_2 + n_3 \cdot a_3.$$

Tarkime, kad kandidatams priskiriamos reikšmės $a_1 = 8, a_2 = 4, a_3 = 12$. O vienas rinkėjas balsuoja už pirmąjį kandidatą, trys rinkėjai – už antrąjį, o už trečiąjį – du. Tokiu atveju

$$D = 1 \cdot 8 + 3 \cdot 4 + 2 \cdot 12 = 44.$$

Tačiau tokią pačią D reikšmę galima gauti ir, kai $n_1 = 3, n_2 = 2, n_3 = 1$:

$$D = 3 \cdot 8 + 2 \cdot 4 + 1 \cdot 12 = 44.$$

Vadinasi suma nusakoma nevienareikšmiškai. Kadangi rinkėjų pasirinkimai yra nepriklausomi, tai galime tik nustatyti papildomas sąlygas kandidatams priskiriamoms generuojamoms reikšmėms, kad jos užtikrintų rezultato sumos D unikalumą.

Taip pat, modifikavus sistemą į kelių pasirinkimų pasikeičia rezultato radimas. Jo paieškai milijonieriaus problemos metodo naudoti nebegalime, nes nebeįmanoma nustatyti ribų (konstantų), kurios vienareikšmiškai nurodytų laimėtoją. Egzistuojant tik dviem pasirinkimams galime nustatyti tam tikrą ribą, kuri vienareikšmiškai nurodytų, kuris variantas surinko daugiau balsų. Jeigu gautas rezultatas mažesnis už nustatytą ribą, laimi pirmasis (mažesnę priskirtą reikšmę turintis pasirinkimas), jeigu rezultatas viršija šią ribą, laimi antrasis pasirinkimas. Tokia riba galime laikyti minimalaus balsų kiekio, kad laimėtų kandidatas su mažiausia priskirta reikšme, bei tos reikšmės sandaugą.

Išaugus pasirinkimų skaičiui tokių tikslių rėžių nustatyti nebeįmanoma, nes galime gauti didelį skirtumą tarp mažiausios ir didžiausios kandidatui priskirtos reikšmės. Tokiu atveju, jeigu daugiausiai balsų surinktų kandidatas su mažiausia priskirta reikšme, o bent vienas rinkėjas balsuotų už kandidatą su didžiausia reikšme, šis balsas galėtų padidinti rezultato sumą taip, kad ji viršytų minimalaus balsų kiekio už kandidatą su mažiausia priskirta reikšme ribą.

2.2. Modifikuota elektroninio balsavimo sistema

Pasirinktoje elektroninio balsavimo sistemoje pritaikytoje keliems pasirinkimams balsavimo dalyviai nepasikeičia. Kaip ir pradinėje sistemoje rinkimuose dalyvauja n rinkėjų V bei S (aptarnaujanti institucija). Modifikavus sistemą galimas pasirinkimų skaičius lygus k.

Koreguotą sistemą taip pat sudaro rinkėjo registracijos, balsavimo bei balsų apskaičiavimo etapas.

2.2.1. Rinkėjo registracijos etapas

Kiekvienas rinkėjas V_i užsiregistruoja S. Po V_i registracijos S sugeneruoja atsitiktinį skaičių R_i. Tada aptarnaujanti institucija S apskaičiuoja R pagal formulę:

$$R = n\sum_{i=1}^{n} R_i.$$

Galiausiai, S siunčia R_i ir neužpildytą balsavimo kortelę su kandidatams priskirtomis reikšmėmis $\{x_1, x_2, \dots, x_k\}$ rinkėjui V_i.

2.2.2. Balsavimo procesas

Rinkėjas gauna balsavimo kortelę su galimais pasirinkimais. Kiekvienas rinkėjas V_i pasirenka už ką balsuoja ir savo pasirinktą balsą $X_i \in \{x_1, x_2, \dots, x_k\}$ atsitiktinai padalija į tiek privačių dalių, kiek yra rinkėjų:

$$X_i = \sum_{j=1}^{n} X_{ij}.$$

Padalijęs balsą, rinkėjas V_i prideda savo n privačių dalių X_{ij} prie R_i:

$$A_{ij} = X_{ij} + R_i.$$

Tada V_i siunčia A_j kiekvienam kitam rinkėjui V_j ir A_i reikšmę pasilieka neišsiųstą. Po to kiekvienas rinkėjas, gavęs visą informaciją iš likusių rinkėjų apskaičiuoja:

$$B_i = \sum_{j=1}^{n} A_{ji} = \sum_{j=1}^{n} (X_{ji} + R_j).$$

Tuomet V_i siunčia B_i visiems kitiems dalyviams. Galiausiai kiekvienas rinkėjas apskaičiuoja:

$$C = \sum_{j=1}^{n} B_j.$$

2.2.3. Balsų surinkimo procesas

Rinkėjas V_i ir aptarnaujanti institucija S veikia kartu. S su turima reikšme R iš rinkėjo turimos reikšmės C suranda kandidatų surinktų balsų sumą:

$$D = C - R.$$

Ši suma unikali ir sudaryta iš kandidatų priskirtų reikšmių ir gautų balsų kiekio:

$$D = n_1 \cdot a_1 + n_2 \cdot a_2 + \cdots + n_k \cdot a_k.$$

Taigi S suranda visas n_i, $i = \overline{1,k}$ ir paskelbia rezultatus.

2.3. Kandidatams priskiriamoms reikšmėms reikalavimų paieška

Kokios turi būti kandidatams priskiriamos reikšmės, užtikrinančios rezultato sumos unikalumą, buvo ieškoma sukūrus programinę priemonę aprašytą 2.5 skyrelyje.

Pasirinkta sistema paremta rinkėjų bendru padalintu skaičiavimu ir kiekvienas rinkėjas tam tikras balso dalis ar apskaičiuotas reikšmes siunčia kitiems. Taigi tokią sistemą įgyvendinti tarp

labai didelio rinkėjų skaičiaus būtų itin sudėtinga. Dėl to, pasirinktą sistemą optimalu panaudoti būtų ten, kur nėra didelio rinkėjų skaičiaus. Pavyzdžiui, Seime, Europos Parlamente ar kitoje institucijoje, turinčioje ribotą balsuotojų skaičių. Europos Parlamentą sudaro vienas iš didžiausių narių skaičių lyginant su kitomis valstybinėmis institucijomis. Dėl to maksimalų rinkėjų skaičių apribojame iki kiek didesnio nei šioje įstaigoje, iki 800.

Kandidatams priskiriamų reikšmių reikalavimų paieška vykdoma priskiriant jiems tik pirminius skaičius arba bet kokius (tiek pirminius, tiek sudėtinius, pagal apskaičiuojamas sąlygas). Tiek vieniems, tiek kitiems skaičiams reikalavimų paieška vykdoma tokiu būdu:

1. Pasirenkamas kandidatų skaičius.
   Galimas kandidatų skaičius $3, 5$ ir 7.
2. Pasirenkamos sąlygos kandidatų reikšmėms.
3. Pirmajam kandidatui pirmoji priskiriama reikšmė lygi kandidatų skaičiui. Jeigu ieškoma tik pirminių skaičių, pirmoji reikšmė yra pirmasis pirminis skaičius didesnis už kandidatų skaičių. Likusiems kandidatams reikšmės priskiriamos pagal nurodytas sąlygas.
4. Nustatomas rinkėjų skaičius lygus $5, 10, 50, 100, 200, 400$ ar 800. Su vienu kandidatams priskirtu skaičių rinkiniu patikrinami visi įmanomi variantai, kiek kiekvienas kandidatas gali surinkti balsų su nustatytu rinkėjų skaičiumi. Jeigu visos sumos gaunamos unikalios pirmojo kandidato reikšmė padidinama vienetu (jeigu tikrinami pirminiai skaičiai, randamas pirmas pirminis skaičius didesnis nei ši reikšmė) ir vėl tikrinami visi įmanomi balsavimo variantai. Toks tikrinimas su kiekvienu rinkėjų skaičiumi vykdomas ne daugiau kaip tris valandas. Jeigu nustatoma dar bent viena tokia pati suma, kokia jau buvo rasta prieš tai su tokiu pačiu kandidatams priskirtų skaičių rinkiniu bei rinkėjų skaičiumi tikrinimas nutraukiamas.
5. Jeigu atlikus visus tikrinimus po tris valandas su visais rinkėjų skaičiais kiekviena suma gaunama unikali, užfiksuojamas patikrintų variantų skaičius ir kartojama nuo 1 žingsnio, kol pasiekiamas maksimalus kandidatų skaičius.

Sąlygos priimamos kaip tinkamos, jeigu patikrinus visus variantus su kiekvienu kandidatų skaičiumi negauname nevienareikšmiškų sumų. Sąlygas siekiame sudaryti tokias, jog kandidatams priskiriamos reikšmės būtų kuo mažesnės.

2.4. Rezultato paieškos metodai

Modifikuotos elektroninio balsavimo sistemos rezultatą sudaro kandidatų priskirtų reikšmių ir gautų balsų kiekio sandaugų sumos:

$$D = n_1 \cdot a_1 + n_2 \cdot a_2 + \cdots + n_k \cdot a_k \quad (n_1 + n_2 + \cdots + n_k = n),$$

$k -$ kandidatų kiekis. Iš šios diofantinės lygties reikia išskirti kiek balsų surinko kiekvienas kandidatas. Tikslaus kandidatų surinktų balsų kiekio buvo ieškoma trimis būdais.

Vienas iš rezultato paieškos būdų yra pilnas perrinkimas. Šiuo atveju rezultato ieškoma būdu, aprašytu 1 algoritme.

1 algoritmas. Pilno perrinkimo metodas

Įvedimas: rinkėjų skaičius n, rezultato suma D, kandidatams priskirtos reikšmės $a_i, i = \overline{1,k}$.

---

kol i_k: 0 iki n **vykdyti**
 kol i_{k-1}: 0 iki n **vykdyti**
 ...
  kol i_2: 0 iki n **vykdyti**
   $i_1 = n - i_2 - \cdots - i_{k-1} - i_k;$
   jeigu $i_1 \geq n$
    jeigu $a_1 i_1 + a_2 i_2 + \cdots + a_{k-1} i_{k-1} + a_k i_k = D$
     tada $\{i_1, i_2, \ldots i_{k-1}, i_k\}$ ir **stop.**

Išvedimas: kandidatų surinktų balsų aibė $\{i_1, i_2, \ldots i_{k-1}, i_k\}$.

Antrasis rezultato radimo metodas – tiesinės diofantinės lygties sprendimas. Jos sprendinio paieškai panaudotas dr. Florentin Smarandache darbe „Integer algorithms to solve diophantine linear equations and systems“ aprašytas algoritmas [15]. Šio metodo aprašymas ir eiga:

Įvedimo parametrai. Tiesinė lygtis $a_1 x_1 + \cdots + a_n x_n = b$, $a_i, b \in \mathbb{Z}$, x_i – sveikieji nežinomieji skaičiai, $i = \overline{1,n}$ – ir ne visi $a_i = 0$.

Grąžinamas rezultatas. Jeigu lygtis turi sveikųjų skaičių sprendinį, surandami visi bendrieji lygties sprendiniai.

Rezultato paieška:

1. Apskaičiuojamas $d = DBD(a_1, \ldots, a_n)$ (didžiausias bendras daliklis tarp visų a_i).

2. Jeigu $d|b$ (b dalija d), tuomet egzistuoja lygties sveikieji sprendiniai. Priešingu atveju – ne, ir algoritmas toliau nebevykdomas.

3. Nustatomas $h = 1$. Jeigu $|d| \neq 1$, lygtis padalijama iš d. Taigi koeficientai pasikeičia:

$$a_i = \frac{a_i}{d}, i = \overline{1,n}, \qquad b = \frac{b}{d}.$$

4. Randamas $a = \min_{a_s \neq 0} |a_s|$ ir nustatomas i, kurio $a_i = a$.
5. Jeigu $a \neq 1$ vykdomas 7 žingsnis.
6. Jeigu $a = 1$:

A. $x_i = -(a_1 x_1 + \cdots + a_{i-1} x_{i-1} + a_{i+1} x_{i+1} + \cdots + a_n x_n - b) \cdot a_i$.

B. Pakeičiama x_i reikšmė visuose prieš tai nustatytose nežinomuosiuose (x_i įrašomas į visas lygtis, gautas ar perskaičiuotas 8 žingsnyje).

C. Visi gauti nežinomi parametrai pakeičiami atitinkamais kintamaisiais $k_1, k_2, \ldots, k_{n-1}, k_{n-2}$.

D. Aprašomas gautas galutinis bendrasis sprendinys ir paieška baigiama.

7. Visi $a_j, j \neq i$ perskaičiuojami tokiu būdu:

$$a_j = a_i q_j + r_j, \qquad q_j = \left[\frac{a_j}{a_i}\right],$$

$$b = a_i q + r, \qquad q = \left[\frac{b}{a_i}\right].$$

8. Aprašomas $x_i = -q_1 x_1 - \cdots - q_{i-1} x_{i-1} - q_{i+1} x_{i+1} - \cdots - q_n x_n + q - t_h$. Ši x_i reikšmė pakeičiama visuose prieš tai nustatytuose nežinomuosiuose (jei 8 žingsnis vykdomas ne pirmą kartą x_i įrašomas į visas lygtis, gautas ar perskaičiuotas šiame žingsnyje).
9. Sudaroma nauja lygtis, kurios koeficientai:

$$\begin{cases} a_1 = r_1, \\ \vdots \\ a_{i-1} = r_{i-1}, \\ a_{i+1} = r_{i+1}, \\ \vdots \\ a_n = r_n. \end{cases} \text{ ir } \begin{cases} a_i = -a_i, \\ b = r, \\ x_i = t_h, \\ h = h + 1. \end{cases}$$

Grįžtama į žingsnį 4.

Tiesinės diofantinės lygties sprendinio algoritmas suranda visus galimus sveikuosius sprendinius su priklausomais ir nepriklausomais nežinomaisiais x_i. Kadangi, mūsų atveju tinkamas tik vienas sprendinys, jis surandamas iš gauto bendrojo sprendinio.

Taip pat rezultato buvo ieškoma naudojant sveikųjų skaičių dalybos operaciją. Šio būdo rezultato paieškos eiga:

1. Surandamos reikšmės q_i ir $r_i, i = \overline{1,k}$:

$$\left[\frac{D}{a_k}\right] = q_k, r_k = D - a_k q_k,$$

...

$$\left[\frac{r_i}{a_{i-1}}\right] = q_{i-1}, r_{i-1} = r_i - a_{i-1} q_{i-1},$$

...

$$\left[\frac{r_2}{a_1}\right] = q_1, r_1 = r_2 - a_1 q_1.$$

2. Jeigu $q_k + q_{k-1} + \cdots + q_1 \neq n,$ vykdomas trečias žingsnis. Priešingu atveju paieška stabdoma ir grąžinamas rezultatas, jog už $k-$ ąjį kandidatą balsavo q_k rinkėjų, už $k-1$ kandidatą – q_{k-1} rinkėjų ir t. t.

3. Randamas $q_i = 1$, jis sumažinamas vienetu, o q_j ir r_j $(j < i)$ perskaičiuojami:

$$r_{i-1} = r_i,$$

...

$$\left[\frac{r_j}{a_{j-1}}\right] = q_{j-1}, r_{j-1} = r_j - a_{j-1} q_{j-1},$$

...

$$\left[\frac{r_2}{a_1}\right] = q_1, r_1 = r_2 - a_1 q_1.$$

2.5. Rezultatų radimo efektyvumo tyrimo metodika

Rezultatų paieškos efektyvumas tiriamas su sukurta programine priemone aprašyta 2.5 skyrelyje. Bandoma nustatyti priklausomybė tarp skaičiavimo laiko ir rinkėjų bei tarp laiko ir kandidatų. Ieškant priklausomybės tarp rinkėjų skaičiaus ir skaičiavimo laiko pasirenkamas kandidatų skaičius lygus 3. O rinkėjų skaičius lygus $20, 50, 100, 200, 400, 600$ ir 800. Su kiekviena iš šių septynių reikšmių atliekama po 15 skaičiavimų ir randamas visų laikų vidurkis. Nustatant priklausomybę tarp skaičiavimo laiko ir kandidatų skaičiaus pasirenkamas rinkėjų skaičius lygus 20. Su kiekviena kandidatų skaičiaus reikšme, $3 - 20,$ atliekama po 15 skaičiavimų ir randamas visų laikų vidurkis. Nustatytas ryšys tarp kintamųjų pateiktas grafikuose.

Visais atvejais pirmajam kandidatui priskiriamas atsitiktinis 16 bitų eilės pirminis skaičius. Likusiems kandidatams pirminis skaičius pagal nurodytas sąlygas $a_i \geq (a_{i-1} - n - 3)n.$

2.6. Sukurtų programinių priemonių aprašas

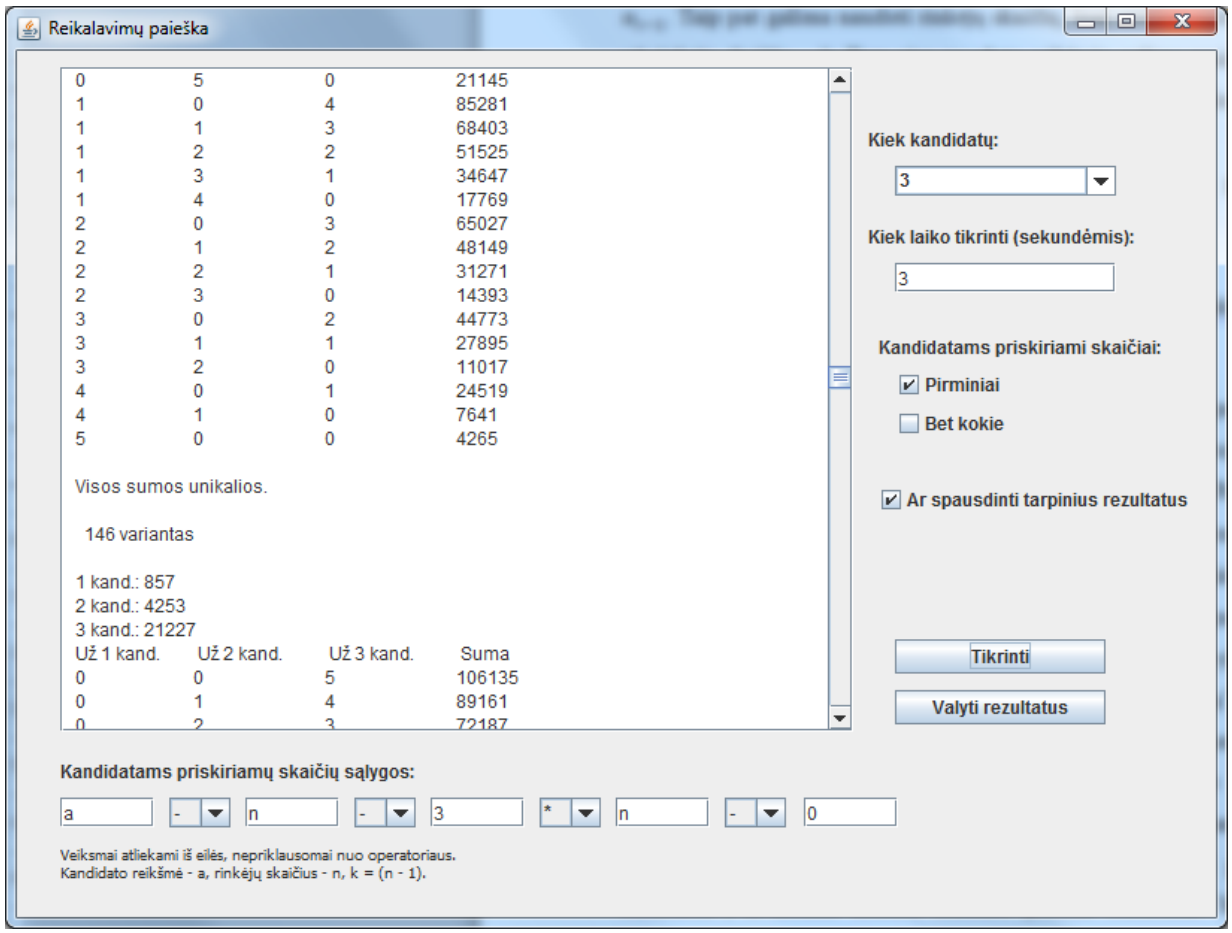
Viena iš sukurtų programinių priemonių skirta reikalavimų kandidatams priskiriamoms reikšmėms paieškai, kita – rezultato paieškos efektyvumo tyrimui. Abi jos sukurtos JAVA programavimo kalba „Eclipse Mars.2“ aplinkoje.

Pirmosios, skirtos kandidatams priskiriamų reikšmių reikalavimų paieškai, programinės priemonės langas pavaizduotas 1 pav. Laukelyje „Kiek kandidatų“ galima pasirinkti norimą

kandidatų skaičių, nuo 3 iki 9. Laukelyje „Kiek laiko tikrinti (sekundėmis)“ nurodoma, kiek sekundžių tikrinti kiekvieną variantą (2.2.skyrelis 3 žingsnis). Taip pat galima pasirinkti ir pažymėti varnelę ties norimu variantu, kokius skaičius naudoti reikalavimų paieškai. Pažymėjus varnelę ties „Ar spausdinti tarpinius rezultatus“ bus parodoma, kokie skaičiai priskiriami kandidatams, visi galimi variantai bei jų suma. Nepažymėjus šio laukelio grąžinamas tik patikrintų variantų skaičius.

Programos lango apačioje galima pasirikti, kokias penkias reikšmes norime priskirti kandidatams. Šioms sąlygoms nurodyti galima naudoti visus sveikuosius teigiamus skaičius. Įrašius raidę a, i −ajam kandidatui bus naudojama $(i-1)$ −ajam kandidatui priskirta reikšmė a_{i-1}. Taip pat galima naudoti rinkėjų skaičių, kurį atitiktų raidė n, ar reikšmę, vienetu mažesnę už rinkėjų skaičių – k. Tarp visų nuodytų reikšmių galima pasirinkti vieną iš keturių pagrindinių aritmetikos operatorių. O visi veiksmai vykdomi iš eilės, nepriklausomai nuo parinkto operatoriaus.

Paspaudus mygtuką „Tikrinti“ pradedama vykdyti programa, paspaudus – „Valyti rezultatus“, išvalomas laukas, kuriame įrašomi rezultatai.

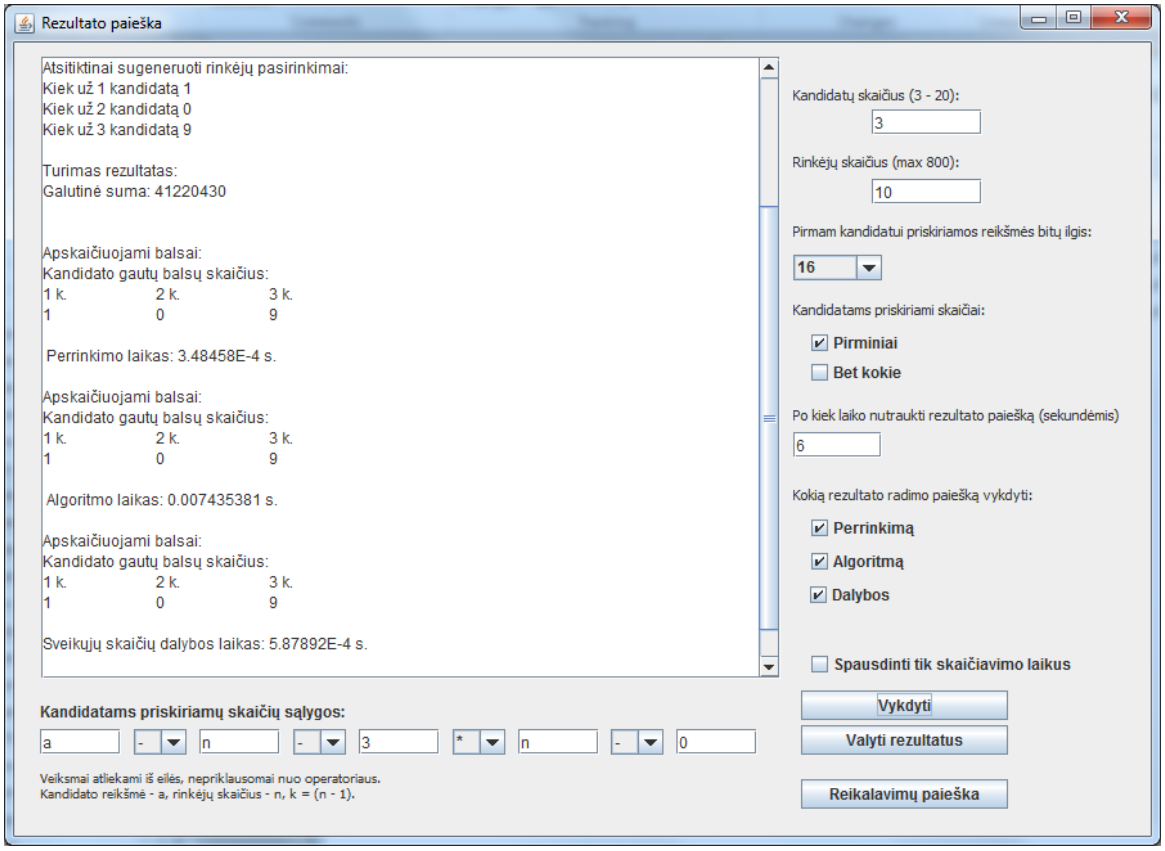


1 pav. Programinės priemonės, skirtos kandidatams priskiriamoms reikšmėms reikalavimų paieškai, langas

Antrosios programinės priemonės, skirtos rezultato paieškos efektyvumui tirti, langas pavaizduotas 2 pav. Joje, taip pat, kaip ir prieš tai aprašytoje reikia pasirinkti sąlygas, pagal kurias apskaičiuojamos kandidatams priskiriamos reikšmės. Pasirinkti kandidatų skaičių nuo 3 iki 20, rinkėjų skaičių nuo 0 iki 800, pažymėti kokius, pirminius ar sudėtinius kandidatams priskiriamus skaičius naudoti. Pirmajam kandidatui priskiriamą skaičiaus bitų ilgį galima pasirinkti lygų $2, 4, 8, 16$ arba 32. Taip pat prieš programos vykdymą įrašomas laikas sekundėmis, po kurio rezultato paieška nutraukiama. Norint vykdyti rezultato paiešką perrinkimo metodu reikia pažymėti varnelę ties laukeliu „Perrinkimą“. Norint, kad vykdytų tiesinės diofantinės lygties sprendimo algoritmą pažymima ties „Algoritmą“. Rezultato paieška sveikųjų skaičių dalybos būdu vykdoma pažymėjus varnelę ties laukeliu „Dalybos“. Pažymėjus „Spausdinti tik skaičiavimo laikus“ rezultato lauke atspausdinami tik metodų skaičiavimo laikai.

Paspaudus mygtuką „Vykdyti“ pradedama vykdyti programa, paspaudus – „Valyti rezultatus“, išvalomas laukas, kuriame įrašomi rezultatai. Mygtukas „Reikalavimų paieška“ atidaro programinės priemonės, skirtos reikalavimų kandidatams priskiriamoms reikšmėms paieškai.

Prieš atlikdama rezultatų paiešką programa atsitiktinai sugeneruoja atitinkamos eilės pirmajam kandidatui priskiriamą reikšmę, taip pat atsitiktinai sugeneruojama, kiek kiekvienas kandidatas surenka balsų.



2 pav. Programinės priemonės, skirtos rezultato paieškos efektyvumui tirti, langas

3. TYRIMŲ REZULTATAI IR JŲ APTARIMAS

Šiame skyriuje pateikiama reikalavimų kandidatams priskiriamų reikšmių paieškos eiga. Apibūdinama kokius reikalavimus pasirinkta sistema užtikrina bei įvertinama, kokiais būdais jie garantuojami. Plačiai aprašoma neužtikrinamų savybių analizė bei rekomendacijos, kokiu būdu būtų galima šiuos reikalavimus išpildyti. Taip pat įvertinamas ir modifikuotos sistemos saugumas.

3.1. Sistemos pritaikymas keliems pasirinkimams

Pagal 2.2 skyrelyje aprašytą metodiką buvo ieškoma, kokie turi būti kandidatams priskiriami skaičiai a_i, kad gauta galutinė suma būtų unikali. Pirmiausia ieškoma kokie turi būti reikalavimai, kai kandidatams priskiriami pirminiai skaičiai, po to, kai gali priskiriami ir sudėtiniai.

Reikalavimų pirminiams skaičiams paieška:

- Kandidatams priskiriami iš eilės einantys pirminiai skaičiai grąžina neunikalias sumas.
- Kai kandidatui priskiriama reikšmė dauginama iš tam tikros konstantos gaunamos nevienareikšmiškos sumos. Tačiau, pavyzdžiui, jeigu yra penki rinkėjai ir dauginame iš skaičiaus lygaus 5, sumas gauname unikalias. Taigi galima daryti prielaidą, jog kandidato reikšmė turi priklausyti nuo rinkėjų skaičiaus n.
- Kandidatams priskiriamos reikšmės, tenkinančios sąlygas $a_i = a_{i-1}n$, iš viso patikrinus 11701834 variantus, grąžina tik unikalias sumas. Tačiau, jeigu kandidatų skaičius būtų itin didelis, tai paskutiniajam kandidatui priskiriama reikšmė pasidarytų labai didelė. Taigi, bandome šią sąlygą pakeisti taip, jog kandidatų reikšmės taptų mažesnės.
- Mažiname reikšmę a_{i-1}. Sumažinus iki $a_i = (a_{i-1} - n - 4)n$ susidaro neunikali suma, tačiau visos unikalios sumos gaunamos, kai $a_i = (a_{i-1} - n - 3)n$. Šiuo atveju iš viso patikrinta 39511905 variantų. Nors tikėtina, kad galima surasti sąlygas, pagal kurias gautume dar mažesnes kandidatams priskiriamas reikšmes, tačiau šią sąlygą priimame kaip galutinę ir toliau paieškos nebetęsiame.

Reikalavimų bet kokiems sveikiesiems skaičiams paieška:

- Kandidatams priskiriami iš eilės einantys skaičiai grąžina neunikalias sumas.

- Vėl darome prielaidą, jog kandidato reikšmė turi priklausyti nuo rinkėjų skaičiaus n. Kaip ir su pirminiais skaičiais, taip ir šiuo atveju kandidatams priskiriamos reikšmės, tenkinančios sąlygas $a_i = a_{i-1}n$, iš viso patikrinus 11710243 variantus grąžina tik unikalias sumas.
- Galutinė sąlyga, gauta priminiams skaičiams sudėtinių reikšmių atveju nebetinka, ji sudaro neunikalias sumas. Tačiau randame, jog unikalios sumos gaunamos, kai $a_i = (a_{i-1} - n - 2)n$. Su šia sąlyga iš viso patikrinta 43848331 variantų.

3.2. Pradinės ir modifikuotos sistemos saugumas

Pasirinktos sistemos autoriai teigia, jog jų sukurta sistema užtikrina visus pagrindinius reikalavimus taikomus elektroninio balsavimo sistemoms. Išanalizavus sistemą detaliau galime pastebėti, kad nors dalis reikalavimų tenkinami ar išpildomi iš dalies, tačiau dalis jų gali būti pažeidžiami.

Prieš balsavimą kiekvienas rinkėjas V_i turi užsiregistruoti aptarnaujančiai institucija S, o ši rinkėją patikrina, taip atrinkdama tik turinčius teisę balsuoti rinkėjus. Tuo pačiu būdu užtikrinamas unikalumas. Jeigu rinkėjas bandytų balsuoti antrą kartą, jis nebegalėtų užsiregistruoti ir negautų rinkimų biuletenio.

Taip pat sistema garantuoja, kad kiekvieno rinkėjo balsas yra įskaičiuojamas tinkamai. Jeigu kuris nors rinkėjas V_i pabandys pakeisti gautus iš kitų rinkėjų privačius duomenis, jo apskaičiuotas rezultatas skirsis nuo kitų. Taigi balsas negali būti pakeistas ir įskaičiuojamas tik toks, kokį jį pateikia kiekvienas rinkėjas. Galima teigti, jog tokiu būdu iš dalies užtikrintas patikrinamumo reikalavimas. Kadangi rinkėjai negali pakeisti kitų balsų, užtikrinama, jog kiekvieno rinkėjo balsas įskaičiuotas ir įrašytas būtent toks, kokį jis pateikė. Nors tiesiogiai pasitikrinti, ar jo balsas įskaitytas tinkamai, rinkėjas galimybės neturi. Vis dėl to, patikrinamumo savybė yra tiesiogiai susijusi su tikslumu, o ši savybė sistemoje, esant nesąžiningiems rinkimų dalyviams, gali būti pažeidžiama, taigi ir patikrinamumas nėra tenkinamas pilnai.

Balsavimas vykdomas tik dalyvaujant visiems rinkėjams, o persiunčiamos reikšmės yra tik dalis balso ar galutinės sumos, taigi iš šių reikšmių atskirti galimą balsavimo rezultatą yra neįmanoma. Saugūs daugiašaliai skaičiavimai apsaugo balsavimo procesą nuo tarpinių rezultatų atskleidimo.

Privatumo savybė garantuojama tik tada, kai įvedame papildomus reikalavimus nepaminėtus originalioje sistemoje. Likusieji reikalavimai, tikslumas bei vientisumas, nėra

visiškai užtikrinami. Galimi šių savybių pažeidimai nagrinėjami plačiau tolimesniuose skyreliuose.

3.2.1. Rinkėjo privatumas

Pasirinktoje sistemoje nėra apibrėžti jokie reikalavimai generuojamiems skaičiams. Tokiu atveju gali susidaryti situacijos, kai įmanoma nustatyti, kaip rinkėjai balsavo.

Pavyzdžiui, yra du kandidatai ir penki rinkėjai. Pirmajam kandidatui priskiriama reikšmė $a_1 = 7$, o antrajam – $a_2 = 1000$. S sugeneruoja ribinę konstantą $M = 21$ bei reikšmes R_i:

$$R = 5\sum_{i=1}^{5} R_i = 5(R_1 + R_2 + R_3 + R_4 + R_5) = 5(3 + 4 + 3 + 7 + 2) = 95.$$

Taigi, rinkėjui V_1 išsiunčiama reikšmė $R_1 = 3$, rinkėjas V_2 gauna $R_2 = 4$, V_3 turi $R_3 = 3$, V_4 – $R_4 = 7, V_5 - R_5 = 2$. Taip pat kiekvienam rinkėjui išsiunčiama ribinė konstanta. Tarkime, jog pirmasis rinkėjas balsuoja už antrąjį kandidatą, antrasis – už pirmąjį, trečiasis – už antrąjį, ketvirtasis – už antrąjį, penktasis – už pirmąjį.

Rinkėjai V_i išskaido savo balsą X_i į penkias dalis:

$$X_1 = \sum_{j=1}^{5} X_{1j} = X_{11} + X_{12} + X_{13} + X_{14} + X_{15} = 251 + 143 + 92 + 216 + 298 = 1000.$$

$$X_2 = \sum_{j=1}^{5} X_{2j} = X_{21} + X_{22} + X_{23} + X_{24} + X_{25} = 2 + 2 + 1 + 0 + 2 = 7.$$

$$X_3 = \sum_{j=1}^{5} X_{3j} = X_{31} + X_{32} + X_{33} + X_{34} + X_{35} = 2 + 413 + 195 + 97 + 293 = 1000.$$

$$X_4 = \sum_{j=1}^{5} X_{4j} = X_{41} + X_{42} + X_{43} + X_{44} + X_{45} = 87 + 217 + 154 + 472 + 70 = 1000.$$

$$X_5 = \sum_{j=1}^{5} X_{5j} = X_{51} + X_{52} + X_{53} + X_{54} + X_{55} = 1 + 0 + 3 + 1 + 2 = 7.$$

Kiekvienas rinkėjas V_i prie kiekvienos savo padalinto balso dalies X_{ij} prideda reikšmę R_i, gautą iš aptarnaujančios institucijos S ir savo balso dalį persiunčia kitam rinkėjui V_j:

$$V_1\colon\ X_{12} + R_1 = 143 + 3 = 146 \text{ (išsiunčia } V_2),$$
$$X_{13} + R_1 = 92 + 3 = 95 \text{ (išsiunčia } V_3),$$
$$X_{14} + R_1 = 216 + 3 = 219 \text{ (išsiunčia } V_4),$$
$$X_{15} + R_1 = 298 + 3 = 301 \text{ (išsiunčia } V_5),$$

$X_{11} + R_1 = 251 + 3 = 254$ (pasilieka sau).

V_2: $X_{21} + R_2 = 2 + 4 = 6$ (išsiunčia V_1),

$X_{23} + R_2 = 1 + 4 = 5$ (išsiunčia V_3),

$X_{24} + R_2 = 0 + 4 = 4$ (išsiunčia V_4),

$X_{25} + R_2 = 2 + 4 = 6$ (išsiunčia V_5),

$X_{22} + R_2 = 2 + 4 = 6$ (pasilieka sau).

V_3: $X_{31} + R_3 = 2 + 3 = 5$ (išsiunčia V_1),

$X_{32} + R_3 = 413 + 3 = 416$ (išsiunčia V_2),

$X_{34} + R_3 = 97 + 3 = 100$ (išsiunčia V_4),

$X_{35} + R_3 = 293 + 3 = 296$ (išsiunčia V_5),

$X_{33} + R_3 = 195 + 3 = 298$ (pasilieka sau).

V_4: $X_{41} + R_4 = 87 + 7 = 94$ (išsiunčia V_1),

$X_{42} + R_4 = 217 + 7 = 224$ (išsiunčia V_2),

$X_{43} + R_4 = 154 + 7 = 161$ (išsiunčia V_3),

$X_{45} + R_4 = 70 + 7 = 77$ (išsiunčia V_5),

$X_{44} + R_4 = 472 + 7 = 479$ (pasilieka sau).

V_5: $X_{51} + R_5 = 1 + 2 = 3$ (išsiunčia V_1),

$X_{52} + R_5 = 0 + 2 = 2$ (išsiunčia V_2),

$X_{53} + R_5 = 3 + 2 = 5$ (išsiunčia V_3),

$X_{54} + R_5 = 1 + 2 = 3$ (išsiunčia V_4),

$X_{55} + R_5 = 2 + 2 = 4$ (pasilieka sau).

Gavęs visas reikšmes kiekvienas rinkėjas jas prideda prie turimos (neišsiųstos) ir siunčia vėl kiekvienam kitam rinkėjui:

V_1: $254 + 6 + 5 + 94 + 3 = 362$ (išsiunčia V_2, V_3, V_4 ir V_5),

V_2: $6 + 146 + 416 + 224 + 2 = 794$ (išsiunčia V_1, V_3, V_4 ir V_5),

V_3: $198 + 95 + 5 + 161 + 5 = 464$ (išsiunčia V_1, V_2, V_4 ir V_5),

V_4: $479 + 219 + 4 + 100 + 3 = 805$ (išsiunčia V_1, V_2, V_3 ir V_5),

V_5: $4 + 301 + 6 + 296 + 77 = 684$ (išsiunčia V_1, V_2, V_3 ir V_4).

Taigi rinkėjai su gautomis ir turimomis reikšmėmis apskaičiuoja galutinę sumą:

V_1: turi: 362,

gauna: $794, 464, 805, 684$,

$362 + 794 + 464 + 805 + 684 = 3109$.

V_2: turi: 794,

gauna: $362, 464, 805, 684$,

$$794 + 362 + 464 + 805 + 684 = 3109.$$

V_3: turi: 464,

gauna: $362, 794, 805, 684$,

$$464 + 362 + 794 + 805 + 684 = 3109.$$

V_4: turi: 805,

gauna: $362, 794, 464, 684$,

$$805 + 362 + 794 + 464 + 684 = 3109.$$

V_5: turi: 684,

gauna: $362, 794, 464, 805$,

$$684 + 362 + 794 + 464 + 805 = 3109.$$

Po šių apskaičiavimų kiekvienas rinkėjas randa reikšmę

$$T = 3109 - 21 = 3088.$$

Tada rinkėjai bendradarbiaudami su S nustato, ar $T > R$ bei suranda rezultatą. Kadangi

$$T = 3088 > 95 = R,$$

galima daryti išvadą, jog daugiau balsų surinko antrasis kandidatas.

Po rezultato suradimo rinkėjai nesunkiai gali atsekti kai kurių rinkėjų pasirinkimą. Kadangi žinomas rinkėjų skaičius ir rinkėjai žino kandidatams priskirtus skaičius, tai lengva nustatyti, kuris kandidatas, kiek balsų surinko. Minimalus balsų skaičius, kad kandidatas laimėtų yra lygus 3, tokiu atveju galutinę sumą rinkėjai gautų didesnę už 3000. Jeigu kandidatas surinktų 4 balsus, galutinė suma būtų didesnė už 4000, bet, kaip matome, gauta galutinė suma yra mažesnė. Vadinasi, antrasis kandidatas surinko tris balsus, o už pirmąjį – balsavo du rinkėjai. Taigi kiekvienas rinkėjas gali surasti reikšmę R:

$$R = 3109 - 2 \cdot 7 - 3 \cdot 1000 = 95.$$

Kadangi $R = 5\sum_{i=1}^{5} R_i$, tai V_i rinkėjo gauta R_i reikšmė negali būti didesnė už 19. Taigi, maksimali įmanoma reikšmė balsuojant už pirmąjį kandidatą yra $7 + 19 = 26$. Jeigu ji didesnė, reiškia, jog rinkėjas tikrai pasirinko antrąjį kandidatą.

Žinome, kad po pirmojo balsų persiuntimo rinkėjai turi tokias reikšmes:

V_1: turi: 254,

gauna iš V_2: 6, V_3: 5, V_4: 94, V_5: 3.

V_2: turi: 6,

gauna iš V_1: 146, V_3: 416, V_4: 224, V_5: 2.

V_3: turi: 198,

gauna iš V_1: 95, V_2: 5, V_4: 161, V_5: 5.

V_4: turi: 479,

gauna iš V_1: 219, V_2: 4, V_3: 100, V_5: 3.

V_5: turi: 4,

gauna iš V_1: 301, V_2: 6, V_3: 296, V_4: 77.

Iš šių, gautų ir turimų reikšmių, V_2 rinkėjas gali išskirti, jog V_1, V_3ir V_4 rinkėjai tikrai balsavo už antrąjį kandidatą, taigi V_5 pasirinko antrąjį. Likusieji rinkėjai taip pat gali lengvai atpažinti rinkėjo pasirinkimą, išskyrus V_1 rinkėją. Jis tiksliai nustatyti gali tik vieno rinkėjo (V_4) pasirinkimą.

Tokia situacija nesusidarytų, jeigu būtų įvesti reikalavimai reikšmėms R_i. Šie skaičiai privalo būti nemažesni už didžiausią kandidatui priskiriamą reikšmę. Mūsų pateiktame pavyzdyje – $R_i \geq 1000$. Tarkime, jog

$$R = 5\sum_{i=1}^{5} R_i = 5(R_1 + R_2 + R_3 + R_4 + R_5) = 5(1211 + 6321 + 2356 + 1212 + 2012)$$
$$= 65560.$$

Tada po pirmojo balsų persiuntimo rinkėjai turi tokias reikšmes:

V_1: turi: 1462,

gauna iš V_2: 6323, V_3: 2358, V_4: 1299, V_5: 2013 .

V_2: turi: 6323,

gauna iš V_1: 1354, V_3: 2769, V_4: 1429, V_5: 2012.

V_3: turi: 2551,

gauna iš V_1: 1303, V_2: 6322, V_4: 1366, V_5: 2015 .

V_4: turi: 1684,

gauna iš V_1: 1427, V_2: 6321, V_3: 2453, V_5: 2013 .

V_5: turi: 2014,

gauna iš V_1: 1509, V_2: 6323, V_3: 2649, V_4: 1282 .

Po antrojo balsų persiuntimo kiekvienas rinkėjas gautų sumą lygią 68574. Tada jie apskaičiuotų

$$T = 68574 - 21 = 68553.$$

Rinkėjai bendradarbiaudami su S nustato, ar $T > R$ bei suranda rezultatą.

$$T = 68553 > 65560 = R.$$

Rezultatą kaip ir prieš tai gauname teisingą. Daugiau balsų surinko antrasis kandidatas, tačiau šiuo atveju rinkėjai nebegali nustatyti, kiek tiksliai balsų surinko kiekvienas kandidatas. O bandant nustatyti galimą minimalią R reikšmę gauname, jog R_i reikšmė būtų ne didesnė už 12714. Šis skaičius yra žymiai didesnis už bet kurią turimą ar gautą rinkėjo reikšmę, taigi išskirti rinkėjo balso tampa nebeįmanoma.

Taip pat balsų nebebūtų įmanoma išskirti, jeigu kandidatams priskirtume panašaus dydžio reikšmes. Pavyzdžiui, $a_1 = 7$, o $a_2 = 8$. Tada pridėjus net ir mažesnius R_i už kandidato didžiausią reikšmę, gautųsi panašaus dydžio siunčiamos reikšmės, kas neleistų atsekti rinkėjo pasirinkimo.

Taigi originali pasirinkta elektroninio balsavimo sistema veiktų tinkamai tik nustačius aukščiau aprašytus reikalavimus aptarnaujančios institucijos generuojamam skaičiui R arba kandidatams priskiriamoms reikšmėms.

Modifikavus sistemą į kelių pasirinkimų sistemą ieškant rezultato būtina žinoti tikslų surinktą kiekvieno kandidato balsų skaičių. Tokiu atveju, kiekvienas rinkėjas lengvai apskaičiuotų R reikšmę. Dėl to garantuojant patikimą balso užmaskavimą geriausia naudoti R_i, kuris n kartų didesnis už didžiausią kandidato priskiriamą skaičių. Kitu atveju galėtų susidaryti prieš tai aprašyta situacija, kai rinkėjams labai nesudėtinga atrinkti kiekvieno kito rinkimų dalyvio pasirinkimą.

3.2.2. Sistemos tikslumas ir vientisumas

Pasirinktoje sistemoje nesąžiningi rinkėjai gali pakeisti savo balso reikšmę taip, jog tai pakoreguotų galutinį rezultatą. Tarkime, kad yra du kandidatai ir penki rinkėjai. Pirmajam kandidatui priskiriama reikšmė $a_1 = 10$, o antrajam – $a_2 = 20$. S sugeneruoja ribinę konstantą $M = 30$ bei reikšmes R_i:

$$R = \sum_{i=1}^{5} R_i = 5(R_1 + R_2 + R_3 + R_4 + R_5) = 5(24 + 31 + 24 + 32 + 23) = 670.$$

Tarkime, jog pirmasis rinkėjas balsuoja už antrąjį kandidatą, antrasis, ketvirtasis ir penktasis – už pirmąjį. O trečiasis rinkėjas nori, jog laimėtų antrasis kandidatas, todėl sukčiauja ir vietoj galimų kandidatų reikšmių pasirenka skaičių, lygų 50.

Rinkėjai V_i išskaido savo balsą X_i, o trečiasis rinkėjas savo sugalvotą reikšmę į penkias dalis, sau pasilikdamas didžiausią:

$$X_1 = \sum_{j=1}^{5} X_{1j} = X_{11} + X_{12} + X_{13} + X_{14} + X_{15} = 4 + 5 + 4 + 3 + 4 = 20.$$

$$X_2 = \sum_{j=1}^{5} X_{2j} = X_{21} + X_{22} + X_{23} + X_{24} + X_{25} = 2 + 2 + 1 + 3 + 2 = 10.$$

$$X_3 = \sum_{j=1}^{5} X_{3j} = X_{31} + X_{32} + X_{33} + X_{34} + X_{35} = 3 + 6 + 35 + 5 + 1 = 50.$$

$$X_4 = \sum_{j=1}^{5} X_{4j} = X_{41} + X_{42} + X_{43} + X_{44} + X_{45} = 6 + 0 + 1 + 1 + 2 = 10.$$

$$X_5 = \sum_{j=1}^{5} X_{5j} = X_{51} + X_{52} + X_{53} + X_{54} + X_{55} = 1 + 3 + 3 + 1 + 2 = 10.$$

Kiekvienas rinkėjas V_i prie kiekvienos savo padalinto balso dalies X_{ij} prideda reikšmę R_i, gautą iš aptarnaujančios institucijos S ir savo balso dalį persiunčia kitam rinkėjui V_j. Po pirmojo persiuntimo rinkėjai turi tokias reikšmes:

V_1: turi: 28,
gauna iš V_2: 33, V_3: 27 , V_4: 38, V_5: 24.

V_2: turi: 33,
gauna iš V_1: 29, V_3: 30 , V_4: 32, V_5: 26.

V_3: turi: 59,
gauna iš V_1: 28, V_2: 32 , V_4: 33, V_5: 26.

V_4: turi: 33,
gauna iš V_1: 27, V_2: 34, V_3: 29, V_5: 24.

V_5: turi: 25,
gauna iš V_1: 28, V_2: 33, V_3: 25, V_4: 34.

Gavę visas reikšmes kiekvienas rinkėjas jas prideda prie turimos (neišsiųstos) ir siunčia vėl kiekvienam kitam rinkėjui:

V_1: $28 + 33 + 27 + 38 + 24 = 150$ (išsiunčia V_2, V_3, V_4 ir V_5),
V_2: $29 + 33 + 30 + 32 + 26 = 150$ (išsiunčia V_1, V_3, V_4 ir V_5),
V_3: $28 + 32 + 59 + 33 + 26 = 178$ (išsiunčia V_1, V_2, V_4 ir V_5),
V_4: $27 + 34 + 29 + 33 + 24 = 147$ (išsiunčia V_1, V_2, V_3 ir V_5),
V_5: $28 + 33 + 25 + 34 + 25 = 145$ (išsiunčia V_1, V_2, V_3 ir V_4).

Visi rinkėjai su gautomis ir turimomis reikšmėmis apskaičiuoja galutinę sumą:

$$150 + 150 + 178 + 147 + 145 = 770.$$

Po šių apskaičiavimų kiekvienas rinkėjas randa reikšmę

$$T = 770 - 30 = 740.$$

Tada V_i bendradarbiaudami su S nustato, ar $T > R$ bei suranda rezultatą. Kadangi

$$T = 740 > 670 = R,$$

galima daryti išvadą, jog daugiau balsų surinko antrasis kandidatas. Tačiau žinome, jog daugiau rinkėjų balsavo už pirmąjį kandidatą. Taigi trečiasis rinkėjas pakeitė savo balsą taip, jog nulėmė visiškai kitą rinkimų baigtį. Išskirti, kad jo balsas nėra tinkamas neįmanoma nė viename

siuntimo etape. Aptarnaujanti institucija žinodama R reikšmę gali bandyti nustatyti ar rinkimų rezultatas teisingas, jei jis neviršija galimų rėžių. Pavyzdžiui, mūsų nagrinėtu atveju T negali būti didesnė nei $R + 5a_2 - M = 740$. Gautas T lygus šiai reikšmei, vadinasi rezultatas butų priimamas kaip teisingas. Tačiau, jei trečiasis rinkėjas pasirinktų didesnį skaičių, rezultatas viršytų galimas ribas ir rezultatą tektų atmesti. Nors tokiu atveju, neįmanoma atrinkti, kuris rinkėjas balsuoja netinkamai, tačiau bent jau klaidingas rezultatas nebūtų įskaitomas kaip teisingas.

Taip pat įmanoma, jog aptarnaujanti institucija nėra sąžininga balsavimo proceso dalyvė. Tokiu atveju ji galėtų bendradarbiauti su norinčiu pakenkti ar pakeisti rezultatus dalyviu, ir toks sukčiavimas jokiais būdais nebūtų atskleistas.

Taigi, matome, jog nesąžiningiems rinkėjams pakeisti galutinį rezultatą nėra sudėtinga, o neteisingas balsas visuomet bus priimamas kaip tinkamas. Pakeisti balsą taip, jog laimėtų konkretus kandidatas, kuris norime, kad laimėtų šiek tiek sunkiau, nes nežinome ar pasirinktas skaičius neviršys galimų rėžių. Tačiau esant nesąžiningai ir S tą padaryti būtų labai paprasta. O sužlugdyti visus rinkimus net ir vieninteliam rinkėjui visiškai nesudėtinga.

Modifikuotoje sistemoje nesąžiningas rinkėjas taip pat gali pakeisti savo balso reikšmę į bet kokį kitą skaičių, tačiau jam tai padarius galutinė suma gautųsi tokia, jog nebūtų įmanoma rasti kandidatų gautų balsų kiekio. Pavyzdžiui, jeigu yra penki rinkėjai, trys kandidatai ir $a_1 = 13, a_2 = 29, a_3 = 107$. S sugeneruoja reikšmes R_i:

$$R = \sum_{i=1}^{5} R_i = 5(R_1 + R_2 + R_3 + R_4 + R_5) = 5(132 + 107 + 213 + 145 + 202) = 3995.$$

Tarkime, jog pirmasis ir penktasis rinkėjas balsuoja už pirmąjį kandidatą, antrasis – už antrąjį, ketvirtasis – už trečiąjį. O trečiasis rinkėjas sukčiauja ir vietoj galimų kandidatų reikšmių pasirenka skaičių, lygų 87.

Rinkėjai V_i išskaido savo balsą X_i, o trečiasis rinkėjas savo sugalvotą reikšmę į penkias dalis:

$$X_1 = \sum_{j=1}^{5} X_{1j} = X_{11} + X_{12} + X_{13} + X_{14} + X_{15} = 4 + 5 + 1 + 1 + 2 = 13.$$

$$X_2 = \sum_{j=1}^{5} X_{2j} = X_{21} + X_{22} + X_{23} + X_{24} + X_{25} = 8 + 7 + 5 + 4 + 5 = 29.$$

$$X_3 = \sum_{j=1}^{5} X_{3j} = X_{31} + X_{32} + X_{33} + X_{34} + X_{35} = 13 + 22 + 16 + 15 + 21 = 87.$$

$$X_4 = \sum_{j=1}^{5} X_{4j} = X_{41} + X_{42} + X_{43} + X_{44} + X_{45} = 16 + 30 + 21 + 28 + 12 = 107.$$

$$X_5 = \sum_{j=1}^{5} X_{5j} = X_{51} + X_{52} + X_{53} + X_{54} + X_{55} = 2 + 3 + 3 + 3 + 2 = 13.$$

Pridėję prie kiekvienos savo padalinto balso dalies X_{ij} reikšmę R_i ir persiuntę jas kiekvienam kitam rinkėjai turi tokias reikšmes:

V_1: turi: 136,
gauna iš V_2: 115, V_3: 226, V_4: 161, V_5: 204.

V_2: turi: 114,
gauna iš V_1: 137, V_3: 235, V_4: 175, V_5: 205.

V_3: turi: 229,
gauna iš V_1: 133, V_2: 112, V_4: 166, V_5: 205.

V_4: turi: 173,
gauna iš V_1: 133, V_2: 111, V_3: 228, V_5: 205.

V_5: turi: 204,
gauna iš V_1: 134, V_2: 112, V_3: 234, V_4: 157.

Gavę visas reikšmes kiekvienas rinkėjai jas prideda prie turimos (neišsiųstos) ir siunčia vėl kiekvienam kitam rinkėjui:

V_1: $136 + 115 + 226 + 161 + 204 = 842$ (išsiunčia V_2, V_3, V_4 ir V_5),
V_2: $137 + 114 + 235 + 175 + 205 = 866$ (išsiunčia V_1, V_3, V_4 ir V_5),
V_3: $133 + 112 + 229 + 166 + 205 = 845$ (išsiunčia V_1, V_2, V_4 ir V_5),
V_4: $133 + 111 + 228 + 173 + 205 = 850$ (išsiunčia V_1, V_2, V_3 ir V_5),
V_5: $134 + 112 + 234 + 157 + 204 = 841$ (išsiunčia V_1, V_2, V_3 ir V_4).

Kiekvienas rinkėjas su gautomis ir turimomis reikšmėmis apskaičiuoja galutinę sumą:

$$842 + 866 + 845 + 850 + 841 = 4244.$$

Tada rinkėjai bendradarbiaudami su S nustato visų balsų sumą:

$$C - R = 4244 - 3995 = 249.$$

O pagal tokį rezultatą nebūtų įmanoma nustatyti, kiek balsų kuris kandidatas surinko. Galima peržiūrėti visus įmanomus variantus:

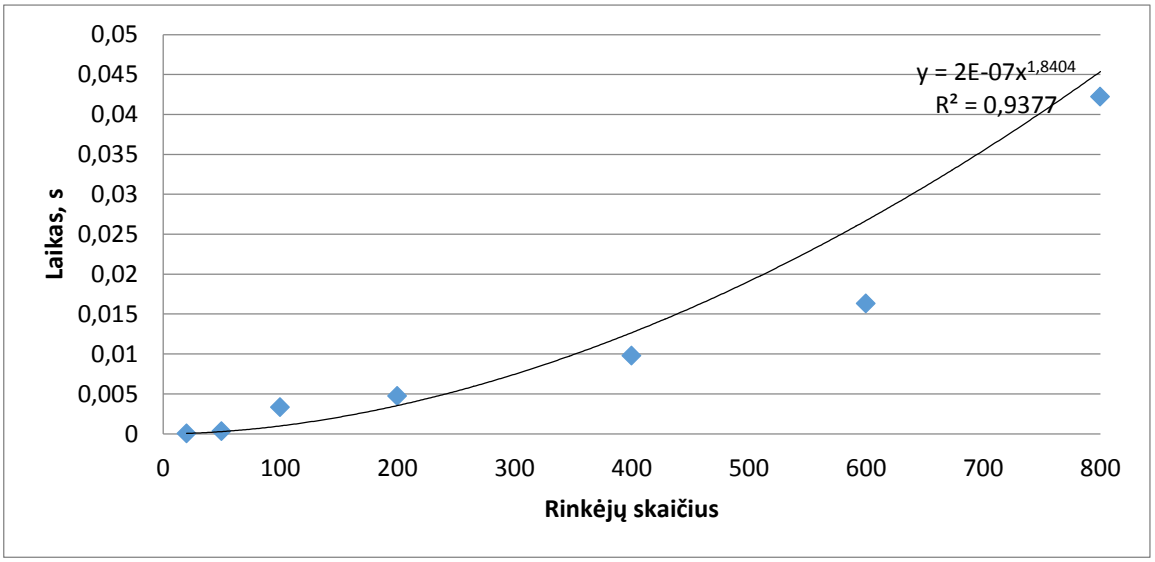
1 lentelė. Visi įmanomi balsavimo variantai, kai yra penki rinkėjai

Už 1 kand.	Už 2 kand.	Už 3 kand.	Suma
0	0	5	535
0	1	4	457
0	2	3	379
0	3	2	301
0	4	1	223
0	5	0	145
1	0	4	441
1	1	3	363
1	2	2	285
1	3	1	207
1	4	0	129
2	0	3	347
2	1	2	269
2	2	1	191
2	3	0	113
3	0	2	253
3	1	1	175
3	2	0	97
4	0	1	159
4	1	0	81
5	0	0	65

Kaip matome, joks galimas variantas nesudaro gautos sumos – 249. Tokiu atveju rinkimai turėtų būti paskelbti negaliojančiais. Taigi nesąžiningiems rinkėjams pakenkti rinkimams nėra sudėtinga. Tačiau rinkėjui pakeisti savo reikšmę taip, kad laimėtų konkretus kandidatas pakankamai sudėtinga. Nežinant kitų kandidatų pasirinkimų, itin sunku nustatyti skaičių, kuris garantuotų, jog galutinė gauta suma atitiks reikiamą.

3.3. Rezultato radimo efektyvumo tyrimas

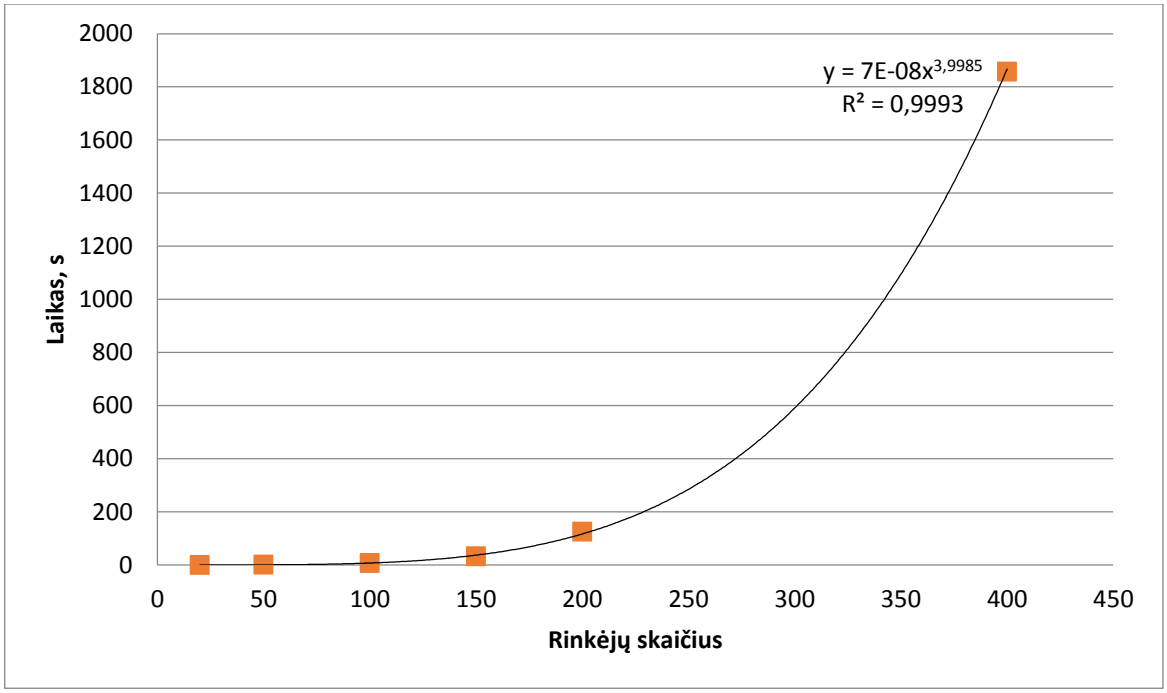
Pirmiausia ieškome rezultato paieškos laiko priklausomybės nuo rinkėjų skaičiaus. Perrinkimo būdu gauti rezultatai pavaizduoti 3 pav.



3 pav. Rezultato radimo pilno perrinkimo metodu laiko priklausomybės nuo rinkėjų skaičiaus grafikas

Šiuo atveju laikus gauname pakankamai mažus. Ryšį tarp skaičiavimo laiko ir rinkimų dalyvių skaičiaus geriausiai atspindi laipsninė funkcija.

Matuojant diofantinės lygties algoritmo rezultato paieškos laiką jį gauname didesnį nei perrinkimo būdu (4 pav.).

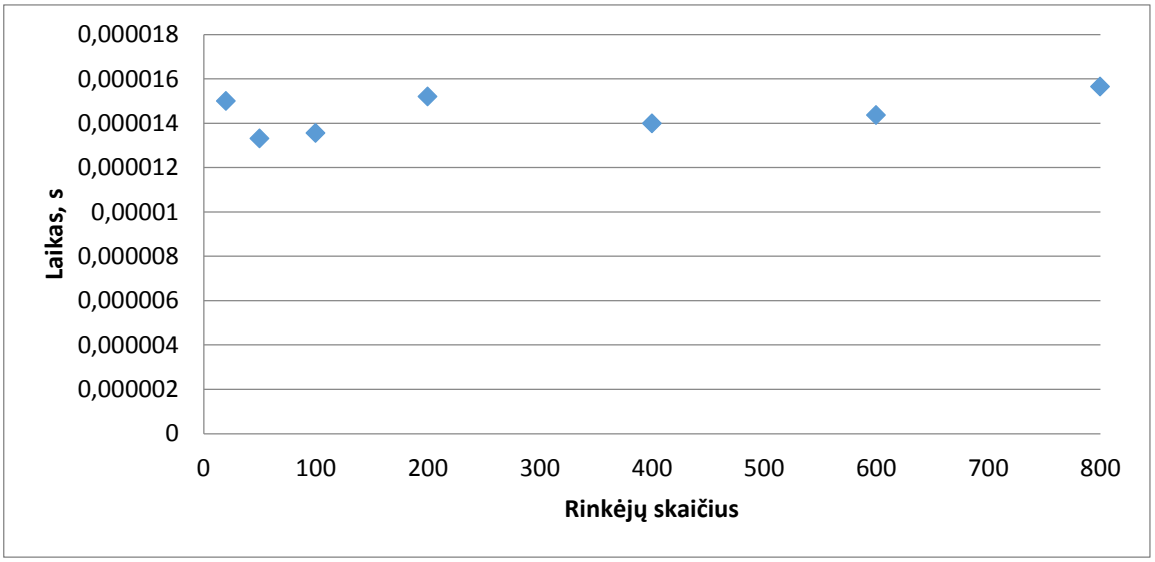


4 pav. Rezultato radimo panaudojus tiesinės diofantinės lygties algoritmą laiko priklausomybės nuo rinkėjų skaičiaus grafikas

Kadangi jau esant 400 rinkėjų skaičiui skaičiavimo laikas viršija 30 minučių ir yra žymiai ilgesnis nei perrinkimo būdu gautas laikas galime teigti, jog algoritmo, skirto tiesinių diofantinių

lygčių sprendimui naudoti balsavimo rezultatų paieškai yra visiškai neoptimalu ir šio algoritmo efektyvumo tyrimo toliau nebetęsiame.

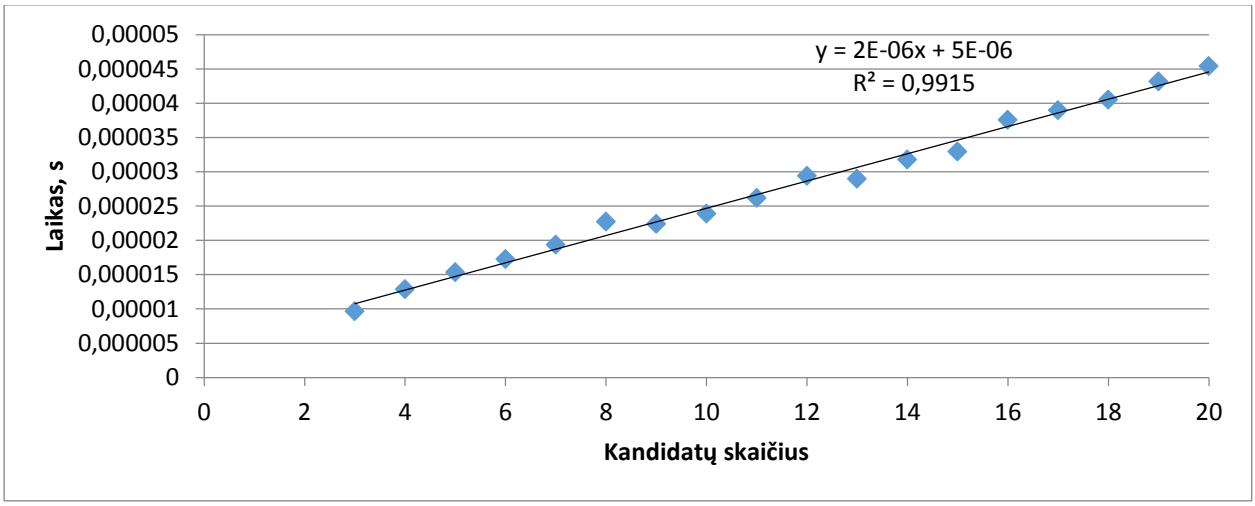
Toliau nustatome rezultato paieškos sveikųjų skaičių metodu skaičiavimo laiko priklausomybę nuo rinkėjų skaičiaus. Šie rezultatai pateikti 5 pav.



5 pav. Rezultato radimo panaudojus sveikųjų skaičių dalybą laiko priklausomybės nuo rinkėjų skaičiaus grafikas

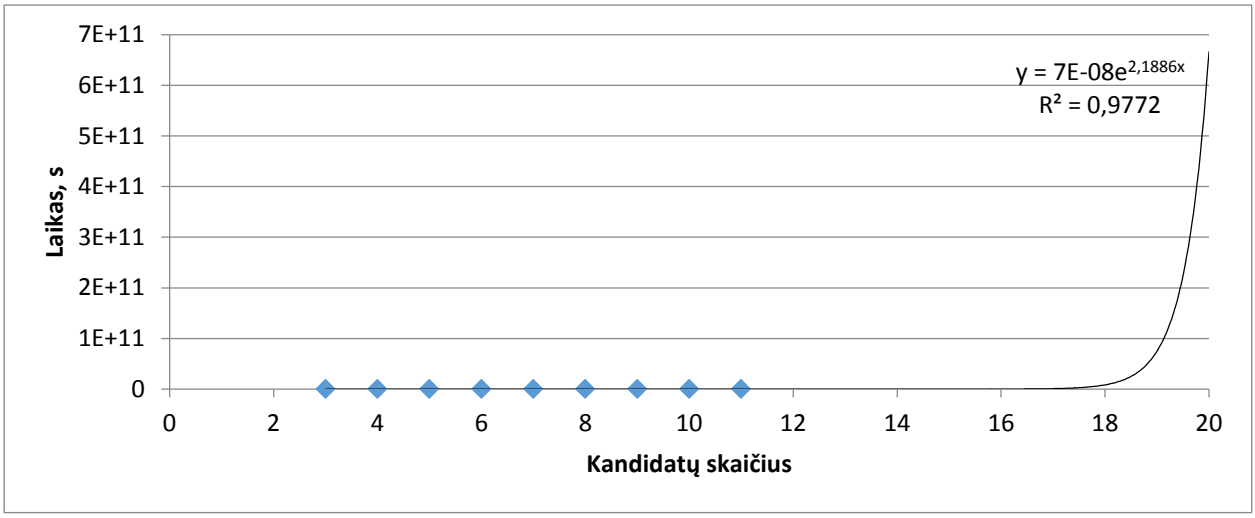
Galime pastebėti, jog rezultato paieškos laikas, naudojant sveikųjų skaičių dalybą, nepriklauso nuo rinkėjų skaičiaus. Šiuo būdu rezultato radimas užtrunka žymiai mažesnį laiko tarpą nei perrinkimo metodu.

Taip pat nustatome rezultato paieškos laiko priklausomybę nuo kandidatų skaičiaus.



6 pav. Rezultato radimo panaudojus sveikųjų skaičių dalybą laiko priklausomybės nuo kandidatų skaičiaus grafikas

Kaip galime pastebėti iš 6 pav. rezultato radimo laikas sveikųjų skaičių dalybos metodu tiesiogiai priklauso nuo kandidatų skaičiaus.



7 pav. Rezultato radimo pilno perrinkimo metodu laiko priklausomybės nuo kandidatų skaičiaus grafikas

Didėjant kandidatų skaičiui pilno perrinkimo būdu rezultato paieškos laikas stipriai auga. Todėl atliekame skaičiavimus tik su devyniomis kandidatų reikšmėmis, o likusias bandome prognozuoti pagal gautųjų priklausomybę. Šiuo metodo laiko priklausomybę nuo kandidatų skaičiaus geriausiai atspindi eksponentinė funkcija. Pagal ją galime nustatyti, jog esant 20 kandidatų rezultato paieška užtruktų ne vienerius metus. O tai žymiai ilgiau nei sveikųjų skaičių dalybos metodu.

IŠVADOS IR REZULTATAI

- Modifikuojant dviejų pasirinkimų elektroninio balsavimo sistemą į kelių pasirinkimų sistemą nuspręsta išlaikyti pradinės sistemos struktūrą ir padidinti galimų balsavimo pasirinkimų (kandidatų) aibę.
- Keičiant pasirinktą elektroninio balsavimo sistemą, nustatyta, jog norint užtikrinti galutinės sumos $D = \sum_{i=1}^{k} n_i \cdot a_i$, $(k -$ kandidatų skaičius) unikalumą būtina įvesti sąlygas kandidatams priskiriamiems skaičiams. Empiriniu būdu nustatytos tokios kandidatams priskiriamų reikšmių sąlygos:
    - Kai naudojami tik pirminiai skaičiai:

$$a_i = (a_{i-1} - n - 3)n.$$

    - Kai naudojami ir pirminiai, ir sudėtiniai skaičiai:

$$a_i = (a_{i-1} - n - 2)n.$$

a_i −kandidatui priskiriama reikšmė, n −rinkėjų skaičius.

- Atlikus saugumo analizę, nustatyta, jog anonimiškumo reikalavimas būtų tenkinamas, jei atsitiktiniai skaičiai išduodami rinkėjams balso maskavimui būtų nemažesni nei didžiausia kandidatams priskiriama reikšmė. Siekiant saugumo rezervo rekomenduojama skaičius, skirtus rinkėjo balso maskavimui, generuoti n kartų didesnius nei didžiausia kandidatams priskiriama reikšmė.
- Tiek originali, tiek modifikuota sistema netenkina dviejų reikalavimų, tikslumo ir vientisumo. Patikrinamumo savybė užtikrinama tik iš dalies. Likusieji reikalavimai (tinkamumas, unikalumas, teisingumas) tenkinami pilnai.
- Atlikus efektyvumo analizę ir atsižvelgus į rezultato paieškos laiko priklausomybę nuo kandidatų bei rinkėjų skaičiaus, nustatyta, jog žymiai efektyvesnis yra rezultato radimas naudojant sveikųjų skaičių dalybą.
- Dėl labai didelio skaičiavimo laiko tiesinės diofantinės lygties sprendimo algoritmas nėra tinkamas balsavimo rezultato paieškai.

LITERATŪRA

1. Eligijus Sakalauskas, Narimantas Listopadskis, Gediminas Simonas Dosinas ir kt. / Kriptografinės sistemos // Kauno technologijos univ. – Kaunas: Vitae Litera, 2008. – 166 p.
2. Shubhangi S. Shinde, Sonali Shukla, prof. D. K. Chitre / Secure e-voting using homomorphic technology // International journal of emerging technology and advanced engineering .- ISSN 2250-2459 .- 2013, t. 6, nr. 8, p. 203-206.
3. Andrea Huszti. A homomorphic encryption-based secure electronic voting scheme // Publ. Math. Debrecen .-2011, t. 79, nr. 3-4, p. 479-496.
4. Suryakanta Panda, Santosh Kumar Sahu, Jagannath Mohapatra, Ramesh Kumar Mohapatra / An application of time stamped proxy blind signature in e-voting // International journal on computer science and engineering .-2013, t. 5, nr. 6, p. 547-552.
5. Mary Bellis. The history of voting machines. Iš *About* [interaktyvus]. [žiūrėta 2016-04-26]. Prieiga per internetą: http://inventors.about.com/library/weekly/aa111300b.htm
6. Ron Rivest, Adi Shamir, Leonard Adleman / A method for obtaining Digital signatures and public-key cryptosystems // Communications of the ACM .-1978.
7. Lelia Barlow. An introduction to electronic voting // CiteSeerX .-2003.
8. Jordi Barrat i Esteve, Ben Goldsmith, John Turner / International experience with e-voting. Norwegian e-vote project // International Foundation for Electoral Systems .-2012.
9. Peter Haynes. Online voting: rewards and risks // Atlantic Council .-2014.
10. Thomas Rossler. E-voting. A survey and introduction // Austria Secure Information Technology Center .-2004.
11. Mahmood Khalel Ibrahem, Nada Mahdi Kiatan / Homomorphic encryption protocol for secure electronic voting system // Al Nahrain University
12. Josh Cohen, Moti Yung / Distributing the power of government to enhance the privacy of voters // In Proceedings of 5th ACM Symposium on Principles of Distributed Computing (PODC) .- 1986, p. 52-62.
13. Krishna Sampigethaya, Radha Poovendran / A framework and taxonomy for comparison of electronic voting schemes // Computers & Security .-2006, t. 25, nr. 2, p. 137-153.
14. Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, J. Alex Halderman / Security analysis of the Estonian internet voting system // ACM Conference on computer and communications security .-2014, p. 703-715.
15. Florentin Smarandache. Integer algorithms to solve diophantine linear equations and systems // University of New Mexico .-2000, 57 p.

16. *Republic of Estonia. Information system authority* [interaktyvus]. [žiūrėta 2016-04-22]. Prieiga per internetą: https://www.ria.ee/en/

17. *ID* [interaktyvus]. [žiūrėta 2016-04-22]. Prieiga per internetą: http://id.ee/?lang=en&id=36881

18. *Evalimine* [interaktyvus]. [žiūrėta 2016-04-23]. Prieiga per internetą: https://github.com/vvk-ehk/evalimine

19. *Comment on the article published in The Guardian* [interaktyvus]. [žiūrėta 2016-04-22]. Prieiga per internetą: http://vvk.ee/valimiste-korraldamine/vvk-uudised/vabariigi-valimiskomisjoni-vastulause-the-guardianis-ilmunud-artiklile/

20. Andrew C. Yao. Protocols for secure computations // IEEE .- ISSN 0272-5428 .- 1982, p. 160-164.

21. David Chaum. Untraceable electronic mail, return addresses, and Digital pseudonyms // Communications of the ACM .-1981, t. 24, nr. 2, p. 84-88.

22. Choonsik Park, Kazutomo Itoh, Kaoru Kurosawa / Efficient anonymous channel and all/nothing election scheme // Eurocrypt ′93 .-1994, t. 765, p. 248-259.

Priedas 1. Sukurtų priemonių programinis kodas.

```
import java.awt.EventQueue;      import java.lang.*;      import java.awt.Font;      import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;      import java.math.*;      import java.util.Random;      import javax.swing.*;

public class allElVotingSystem {

   private JFrame frmRezultatoPaieka;      private JTextField voterNum;      private JTextField candidateNum;
   private JScrollPane scrollPane;      private JTextArea scrollText;      private JCheckBox candPrimes;
   private JCheckBox candAll;      private JComboBox bitNum;      private JTextField timeNum;
   private JLabel lblKadaNutrauktiPaiek;      private JTextField num1;      private JTextField num2;
   private JTextField num3;      private JTextField num4;      private JComboBox op1;
   private JComboBox op2;      private JComboBox op3;      private JCheckBox forceSearch;
   private JCheckBox searchAlg;      private JCheckBox printCount;      private JTextField num5;
   private JComboBox op4;      private JCheckBox searchDiv;

   public static void main(String[] args) {
      EventQueue.invokeLater(new Runnable() {
         public void run() {
            try {
               allElVotingSystem window = new allElVotingSystem();
               window.frmRezultatoPaieka.setVisible(true);
            } catch (Exception e) {
               e.printStackTrace();
            } } }); }

   public allElVotingSystem() { initialize();   }

   private void initialize() {

   //Laukai ----------------------------------------------------------------
      frmRezultatoPaieka = new JFrame();
      frmRezultatoPaieka.setTitle("Rezultato paie\u0161ka");
      frmRezultatoPaieka.setBounds(100, 100, 900, 655);

   frmRezultatoPaieka.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      frmRezultatoPaieka.getContentPane().setLayout(null);
      candidateNum = new JTextField();
      candidateNum.setText("3");
      candidateNum.setBounds(667, 52, 86, 20);
      frmRezultatoPaieka.getContentPane().add(candidateNum);
      candidateNum.setColumns(10);
      voterNum = new JTextField();
      voterNum.setText("100");
      voterNum.setBounds(667, 106, 86, 20);
      frmRezultatoPaieka.getContentPane().add(voterNum);
      voterNum.setColumns(10);
      JLabel lblCandidateNumber = new JLabel("Kandidat\u0173 skai\u010Dius (3 - 20):");
      lblCandidateNumber.setFont(new Font("Tahoma", Font.PLAIN, 11));
      lblCandidateNumber.setBounds(606, 31, 203, 20);
      frmRezultatoPaieka.getContentPane().add(lblCandidateNumber);
      JLabel lblVoterNumber = new JLabel("Rink\u0117j\u0173 skai\u010Dius (max 800):");
      lblVoterNumber.setFont(new Font("Tahoma", Font.PLAIN, 11));
      lblVoterNumber.setBounds(606, 83, 186, 20);
      frmRezultatoPaieka.getContentPane().add(lblVoterNumber);
      scrollPane = new JScrollPane();
      scrollPane.setBounds(21, 11, 575, 484);
      frmRezultatoPaieka.getContentPane().add(scrollPane);
      scrollText = new JTextArea();
      scrollPane.setViewportView(scrollText);
      scrollText.setLineWrap(true);
      candPrimes = new JCheckBox("Pirminiai");
      candPrimes.setSelected(true);
      candPrimes.setBounds(616, 222, 97, 23);
      frmRezultatoPaieka.getContentPane().add(candPrimes);
      candAll = new JCheckBox("Bet kokie");
      candAll.setBounds(616, 245, 97, 23);
      frmRezultatoPaieka.getContentPane().add(candAll);
      ButtonGroup candGroup = new ButtonGroup();
      candGroup.add(candPrimes);
      candGroup.add(candAll);
      JLabel lblKandidatamsPriskiriamiSkaiiai = new JLabel("Kandidatams priskiriami skai\u010Diai:");
      lblKandidatamsPriskiriamiSkaiiai.setFont(new Font("Tahoma", Font.PLAIN, 11));
      lblKandidatamsPriskiriamiSkaiiai.setBounds(606, 201, 223, 14);

   frmRezultatoPaieka.getContentPane().add(lblKandidatamsPriskiriamiSkaiiai);
      String[] bitL = { "2", "4", "8", "16", "32", "64" };
      bitNum = new JComboBox(bitL);
      bitNum.setSelectedIndex(1);
      bitNum.setBounds(606, 165, 69, 20);
      frmRezultatoPaieka.getContentPane().add(bitNum);
      JLabel lblPirmamKandidatuiPriskiriamos = new JLabel("Pirmam kandidatui priskiriamos reik\u0161m\u0117s bit\u0173 ilgis:");
      lblPirmamKandidatuiPriskiriamos.setFont(new Font("Tahoma", Font.PLAIN, 11));
      lblPirmamKandidatuiPriskiriamos.setBounds(606, 140, 306, 14);

   frmRezultatoPaieka.getContentPane().add(lblPirmamKandidatuiPriskiriamos);
      timeNum = new JTextField();
      timeNum.setText("600");
      timeNum.setBounds(606, 303, 69, 20);
      frmRezultatoPaieka.getContentPane().add(timeNum);
      timeNum.setColumns(10);
      lblKadaNutrauktiPaiek = new JLabel("Po kiek laiko nutraukti rezultato paie\u0161k\u0105 (sekund\u0117mis)");
      lblKadaNutrauktiPaiek.setFont(new Font("Tahoma", Font.PLAIN, 11));
      lblKadaNutrauktiPaiek.setBounds(606, 283, 306, 14);
      frmRezultatoPaieka.getContentPane().add(lblKadaNutrauktiPaiek);
      String[] oerator = { "+", "-", "*", "/" };
      op1 = new JComboBox(oerator);
      op1.setSelectedIndex(2);
      op1.setBounds(93, 534, 41, 20);
      frmRezultatoPaieka.getContentPane().add(op1);
      op2 = new JComboBox(oerator);
      op2.setSelectedIndex(1);
      op2.setBounds(217, 534, 41, 20);
      frmRezultatoPaieka.getContentPane().add(op2);
      op3 = new JComboBox(oerator);
      op3.setSelectedIndex(1);
      op3.setBounds(341, 534, 41, 20);
      frmRezultatoPaieka.getContentPane().add(op3);
      op4 = new JComboBox(oerator);
      op4.setSelectedIndex(1);
      op4.setBounds(464, 534, 41, 20);
      frmRezultatoPaieka.getContentPane().add(op4);
      num1 = new JTextField("a");
      num1.setColumns(10);
      num1.setBounds(20, 534, 63, 20);
      frmRezultatoPaieka.getContentPane().add(num1);
      num2 = new JTextField("n");
      num2.setColumns(10);
      num2.setBounds(144, 534, 63, 20);
      frmRezultatoPaieka.getContentPane().add(num2);
      num3 = new JTextField("0");
      num3.setColumns(10);
      num3.setBounds(268, 534, 63, 20);
      frmRezultatoPaieka.getContentPane().add(num3);
      num4 = new JTextField("0");
      num4.setColumns(10);
      num4.setBounds(392, 534, 63, 20);
      frmRezultatoPaieka.getContentPane().add(num4);
      num5 = new JTextField("0");
      num5.setColumns(10);
      num5.setBounds(515, 534, 63, 20);
      frmRezultatoPaieka.getContentPane().add(num5);
```

```
        JLabel label = new JLabel("Kandidato reik\u0161m\u0117 - a,
rink\u0117j\u0173 skai\u010Dius - n, k = (n - 1).");
        label.setFont(new Font("Tahoma", Font.PLAIN, 10));
        label.setBounds(20, 577, 306, 14);
        frmRezultatoPaieka.getContentPane().add(label);
        JLabel label_1 = new JLabel("Veiksmai atliekami i\u0161 eil\u0117s,
nepriklausomai nuo operatoriaus.");
        label_1.setFont(new Font("Tahoma", Font.PLAIN, 10));
        label_1.setBounds(20, 565, 306, 14);
        frmRezultatoPaieka.getContentPane().add(label_1);
        JLabel label_2 = new JLabel("Kandidatams priskiriam\u0173
skai\u010Di\u0173 s\u0105lygos:");
        label_2.setBounds(20, 513, 255, 14);
        frmRezultatoPaieka.getContentPane().add(label_2);
        JLabel lblKokiResultatRadimo = new JLabel("Koki\u0105 rezultato
radimo paie\u0161k\u0105 vykdyti:");
        lblKokiResultatRadimo.setFont(new Font("Tahoma", Font.PLAIN, 11));
        lblKokiResultatRadimo.setBounds(606, 345, 273, 14);
        frmRezultatoPaieka.getContentPane().add(lblKokiResultatRadimo);
        forceSearch = new JCheckBox("Perrinkim\u0105");
        forceSearch.setSelected(true);
        forceSearch.setBounds(616, 366, 97, 23);
        frmRezultatoPaieka.getContentPane().add(forceSearch);
        searchAlg = new JCheckBox("Algoritm\u0105");
        searchAlg.setSelected(true);
        searchAlg.setBounds(616, 392, 97, 23);
        frmRezultatoPaieka.getContentPane().add(searchAlg);
        printCount = new JCheckBox("Spausdinti tik skai\u010Diavimo laikus");
        printCount.setBounds(616, 472, 247, 23);
        frmRezultatoPaieka.getContentPane().add(printCount);
        searchDiv = new JCheckBox("Dalybos");
        searchDiv.setHorizontalAlignment(SwingConstants.CENTER);
        searchDiv.setSelected(true);
        searchDiv.setBounds(602, 418, 97, 23);
        frmRezultatoPaieka.getContentPane().add(searchDiv);

    //Programa ------------------------------------------------------------

        JButton btnSpausti = new JButton("Vykdyti");
        btnSpausti.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    String n1, n2, n3, n4, n5;
                    int cNum, vNum;
                    cNum = takeCandidate();
                    vNum = takeVoter();
                    n1 = takeNum1(); n2 = takeNum2(); n3 = takeNum3(); n4 = takeNum4(); n5 = takeNum5();
                    if (cNum > 2 && cNum <= 20 && vNum <= 800 && (n1.equals("n") || n1.equals("k") || n1.equals("a") || Integer.parseInt(n1) >= 0) && (n2.equals("n") ||
n2.equals("k") || n2.equals("a") || Integer.parseInt(n2) >= 0) && (n3.equals("n") || n3.equals("a") || n3.equals("k") || Integer.parseInt(n3) >= 0) && (n4.equals("n") ||
n4.equals("k") || n4.equals("a") || Integer.parseInt(n4) >= 0) && (n5.equals("n") || n5.equals("k") || n5.equals("a") || Integer.parseInt(n5) >= 0)) {
                printAllResult();
                    }
                    else JOptionPane.showMessageDialog(frmRezultatoPaieka, "Nurodyta(-os) reikšmė(-s) per didelės.");
                }
                catch (Exception exc) {
                    JOptionPane.showMessageDialog(frmRezultatoPaieka, "Neteisingai nurodyta(-os) reikšmė(-s).");
                    return;
                } } });
        btnSpausti.setBounds(612, 504, 161, 23);
        frmRezultatoPaieka.getContentPane().add(btnSpausti);

        JButton btnClear = new JButton("Valyti rezultatus");
        btnClear.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                scrollText.setText("");
            }
        });
        btnClear.setBounds(612, 531, 161, 23);
        frmRezultatoPaieka.getContentPane().add(btnClear);

        JButton btnNewWindow = new JButton("Reikalavim\u0173 paie\u0161ka");
        btnNewWindow.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) { checking ch = new checking(); ch.newWindow();
            }
        });
        btnNewWindow.setBounds(612, 573, 161, 23);
        frmRezultatoPaieka.getContentPane().add(btnNewWindow);

    }

    public static int randInt(int min, int max) { Random rnd = new Random(); int randomNum = rnd.nextInt(max - min + 1) + min; return randomNum; }

    public int takeCandidate() { int candNum; candNum = Integer.parseInt(candidateNum.getText()); return candNum; }

    public int takeVoter() { int voteNum; voteNum = Integer.parseInt(voterNum.getText()); return voteNum; }

    public int takeTime() { int t; t = Integer.parseInt(timeNum.getText()); return t; }

    public String takeNum1() { String c = num1.getText(); return c; }

    public String takeNum2() { String c = num2.getText(); return c; }

    public String takeNum3() { String c = num3.getText(); return c; }

    public String takeNum4() { String c = num4.getText(); return c; }

    public String takeNum5() { String c = num5.getText(); return c; }

    public String takeOperator1() { String c = op1.getSelectedItem().toString(); return c; }

    public String takeOperator2() { String c = op2.getSelectedItem().toString(); return c; }

    public String takeOperator3() { String c = op3.getSelectedItem().toString(); return c; }
```

```
    public String takeOperator4() { String c = op4.getSelectedItem().toString(); return c; }

    public String takeBits() { String c = bitNum.getSelectedItem().toString(); return c; }

    public void printAllResult() {
        int c = takeCandidate();
        int v = takeVoter();
        BigInteger[] rc = candidateValues();
        if (printCount.isSelected() == false) {
            scrollText.append("Pasirinktos reikšmės:\n");
            scrollText.append("Kandidatų skaičius: " + c + "\n");
            scrollText.append("Rinkėjų skaičius: " + v + "\n\n");
            scrollText.append("Atsitiktinai sugeneruotos kandidatams priskirtos reikšmės:\n");
            for (int i = 0; i < c; i++) {
                scrollText.append(BigInteger.valueOf(i + 1) + " kand.: " + rc[i] + "\n");
            }
            scrollText.append("\n");
            scrollText.append("Atsitiktinai sugeneruoti rinkėjų pasirinkimai:\n");
        }
        BigInteger[] hmv = howManyVotes();
        if (printCount.isSelected() == false)
            scrollText.append("Turimas rezultatas:\n");
        BigInteger finRez = result(rc, hmv);
        if (printCount.isSelected() == false)
            scrollText.append("\n");
        if (forceSearch.isSelected() == true) {
            if (printCount.isSelected() == false)
                scrollText.append("Apskaičiuojami balsai:\n");
            searchAll(rc, finRez);
            scrollText.append("\n");
        }
        if (searchAlg.isSelected() == true) {
            if (printCount.isSelected() == false)
                scrollText.append("Apskaičiuojami balsai:\n");
            findOther(rc, finRez, v, c);
        }
        if (searchDiv.isSelected() == true) {
            if (printCount.isSelected() == false)
                scrollText.append("Apskaičiuojami balsai:\n");
            findRes3(finRez, rc, v, c);
        }
    }

    // Kandidatų reikšmės
    public BigInteger[] candidateValues() {
        String n1, n2, n3, n4, n5, o1, o2, o3, o4;
        boolean in1 = false, in2 = false, in3 = false, in4 = false, in5 = false;
        boolean isP = false;
        int bitLength = Integer.parseInt(takeBits());
        int c = takeCandidate();
        int v = takeVoter();
        BigInteger sv = BigInteger.ONE;
        BigInteger[] rc = new BigInteger[c];
        Random rnd = new Random();
        n1 = takeNum1(); n2 = takeNum2(); n3 = takeNum3(); n4 = takeNum4();
n5 = takeNum5(); o1 = takeOperator1(); o2 = takeOperator2(); o3 =
takeOperator3(); o4 = takeOperator4();
        BigInteger var1 = BigInteger.ZERO, var2 = BigInteger.ZERO, var3 =
BigInteger.ZERO, var4 = BigInteger.ZERO, var5 = BigInteger.ZERO, var =
BigInteger.ZERO, vari = BigInteger.ZERO, vari2 = BigInteger.ZERO;
        rc[0] = BigInteger.probablePrime(bitLength, rnd);
        if (candPrimes.isSelected() == true) {
            while (isP != true) {
                if (rc[0].isProbablePrime(100) == true)
                    isP = true;
                else
                    rc[0] = rc[0].add(sv);
            }
        }
        if (n1.equals("n")) var1 = BigInteger.valueOf(v);
        else if (n1.equals("a")) { var1 = rc[0]; in1 = true; }
            else if (n1.equals("k")) var1 = BigInteger.valueOf(v-1);
                else var1 = new BigInteger(n1);
        if (n2.equals("n")) var2 = BigInteger.valueOf(v);
        else if (n2.equals("a")) { var2 = rc[0]; in2 = true; }
            else if (n2.equals("k")) var2 = BigInteger.valueOf(v-1);
                else var2 = new BigInteger(n2);
        if (n3.equals("n")) var3 = BigInteger.valueOf(v);
        else if (n3.equals("a")) { var3 = rc[0]; in3 = true; }
            else if (n3.equals("k")) var3 = BigInteger.valueOf(v-1);
                else var3 = new BigInteger(n3);
        if (n4.equals("n")) var4 = BigInteger.valueOf(v);
        else if (n4.equals("a")) { var4 = rc[0]; in4 = true; }
            else if (n4.equals("k")) var4 = BigInteger.valueOf(v-1);
                else var4 = new BigInteger(n4);
        if (n5.equals("n")) var5 = BigInteger.valueOf(v);
        else if (n5.equals("a")) { var5 = rc[0]; }
            else if (n5.equals("k")) var5 = BigInteger.valueOf(v-1);
                else var5 = new BigInteger(n5);

        // jeigu pirminiai
        if (candPrimes.isSelected() == true) {
            for (int i = 1; i < c; i++) {
                isP = false;
                if (in1 == false && in2 == false) {
                    if (o1.equals("*")) var = var1.multiply(var2);
                    else if (o1.equals("+")) var = var1.add(var2);
                        else if (o1.equals("-")) var = var1.subtract(var2);
                            else var = var1.divide(var2);
                }
                if (in1 == true && in2 == false) {
                    if (o1.equals("*")) var = rc[i-1].multiply(var2);
                    else if (o1.equals("+")) var = rc[i-1].add(var2);
                        else if (o1.equals("-")) var = rc[i-1].subtract(var2);
                            else var = rc[i-1].divide(var2);
                }
                if (in1 == false && in2 == true) {
                    if (o1.equals("*")) var = var1.multiply(rc[i-1]);
                    else if (o1.equals("+")) var = var1.add(rc[i-1]);
                        else if (o1.equals("-")) var = var1.subtract(rc[i-1]);
                            else var = var1.divide(rc[i-1]);
                }
                if (in3 == true) {
                    if (o2.equals("*")) vari = var.multiply(rc[i-1]);
                    else if (o2.equals("+")) vari = var.add(rc[i-1]);
                        else if (o2.equals("-")) vari = var.subtract(rc[i-1]);
                            else vari = var.divide(rc[i-1]);
                }
                if (in3 == false) {
                    if (o2.equals("*")) vari = var.multiply(var3);
                    else if (o2.equals("+")) vari = var.add(var3);
                        else if (o2.equals("-")) vari = var.subtract(var3);
                            else vari = var.divide(var3);
                }
                if (in4 == true) {
                    if (o3.equals("*")) vari2 = vari.multiply(rc[i-1]);
                    else if (o3.equals("+")) vari2 = vari.add(rc[i-1]);
                        else if (o3.equals("-")) vari2 = vari.subtract(rc[i-1]);
                            else vari2 = vari.divide(rc[i-1]);
                }
                if (in4 == false) {
                    if (o3.equals("*")) vari2 = vari.multiply(var4);
                    else if (o3.equals("+")) vari2 = vari.add(var4);
                        else if (o3.equals("-")) vari2 = vari.subtract(var4);
                            else vari2 = vari.divide(var4);
```

```
            }
            if (in5 == false) {
               if (o4.equals("*")) rc[i] = vari2.multiply(var5);
               else if (o4.equals("+")) rc[i] = vari2.add(var5);
                  else if (o4.equals("-")) rc[i] = vari2.subtract(var5);
                     else rc[i] = vari2.divide(var5);
            }
            while (isP != true) {
               if (rc[i].isProbablePrime(100) == true)
                  isP = true;
               else
                  rc[i] = rc[i].add(sv);
            }
         }
      }
      // jeigu bet kokie
      else {
         for (int i = 1; i < c; i++) {
            if (in1 == false && in2 == false) {
               if (o1.equals("*")) var = var1.multiply(var2);
               else if (o1.equals("+")) var = var1.add(var2);
                  else if (o1.equals("-")) var = var1.subtract(var2);
                     else var = var1.divide(var2);
            }
            if (in1 == true && in2 == false) {
               if (o1.equals("*")) var = rc[i-1].multiply(var2);
               else if (o1.equals("+")) var = rc[i-1].add(var2);
                  else if (o1.equals("-")) var = rc[i-1].subtract(var2);
                     else var = rc[i-1].divide(var2);
            }
            if (in1 == false && in2 == true) {
               if (o1.equals("*")) var = var1.multiply(rc[i-1]);
               else if (o1.equals("+")) var = var1.add(rc[i-1]);
                  else if (o1.equals("-")) var = var1.subtract(rc[i-1]);
                     else var = var1.divide(rc[i-1]);
            }
            if (in3 == true) {
               if (o2.equals("*")) vari = var.multiply(rc[i-1]);
               else if (o2.equals("+")) vari = var.add(rc[i-1]);
                  else if (o2.equals("-")) vari = var.subtract(rc[i-1]);
                     else vari = var.divide(rc[i-1]);
            }
            if (in3 == false) {
               if (o2.equals("*")) vari = var.multiply(var3);
               else if (o2.equals("+")) vari = var.add(var3);
                  else if (o2.equals("-")) vari = var.subtract(var3);
                     else vari = var.divide(var3);
            }
            if (in4 == true) {
               if (o3.equals("*")) vari2 = vari.multiply(rc[i-1]);
               else if (o3.equals("+")) vari2 = vari.add(rc[i-1]);
                  else if (o3.equals("-")) vari2 = vari.subtract(rc[i-1]);
                     else vari2 = vari.divide(rc[i-1]);
            }
            if (in4 == false) {
               if (o3.equals("*")) vari2 = vari.multiply(var4);
               else if (o3.equals("+")) vari2 = vari.add(var4);
                  else if (o3.equals("-")) vari2 = vari.subtract(var4);
                     else vari2 = vari.divide(var4);
            }
            if (in5 == false) {
               if (o4.equals("*")) rc[i] = vari2.multiply(var5);
               else if (o4.equals("+")) rc[i] = vari2.add(var5);
                  else if (o4.equals("-")) rc[i] = vari2.subtract(var5);
                     else rc[i] = vari2.divide(var5);
            }
         }
      }
      return rc;
   }

   public BigInteger[] howManyVotes() {
      int c = takeCandidate();
      int v = takeVoter();
      BigInteger[] hmv = new BigInteger[c];
      int rnd, r;
      BigInteger sum = new BigInteger("0");
      if (v <= 5)
         r = v;
      else if (v > 5 && v <= 15)
         r = v / 2;
         else
            r = v / 3;
      rnd = randInt(0, r);
      hmv[0] = BigInteger.valueOf(rnd);
      r = v - rnd;
      if (printCount.isSelected() == false)
         scrollText.append("Kiek už 1 kandidatą " + hmv[0] + "\n");
      for (int i = 1; i < c - 1; i++) {
         if (rnd == v) {
            hmv[i] = BigInteger.valueOf(0);
            return hmv;
         }
         r = r - rnd;
         rnd = randInt(0, r);
         hmv[i] = BigInteger.valueOf(rnd);
         if (printCount.isSelected() == false)
            scrollText.append("Kiek už " + (i + 1) + " kandidatą " + hmv[i] + "\n");
      }
      for (int i = 0; i < c - 1; i++) {
         sum = sum.add(hmv[i]);
      }
      hmv[c-1] = BigInteger.valueOf(v).subtract(sum);
      if (printCount.isSelected() == false)
         scrollText.append("Kiek už " + c + " kandidatą " + hmv[c-1] + "\n\n");
      return hmv;
   }

   public BigInteger result(BigInteger[] rc, BigInteger[] cv) {
      int c = takeCandidate();
      BigInteger finRez = new BigInteger("0");
      BigInteger[] rez = new BigInteger[c];

      for (int i = 0; i < c; i++) {
         rez[i] = rc[i].multiply(cv[i]);
         finRez = finRez.add(rez[i]);
         /*if (printCount.isSelected() == false)
            scrollText.append("Rezultatas " + (i + 1) + ":  " + rez[i] + "\n");*/
      }
      if (printCount.isSelected() == false)
         scrollText.append("Galutinė suma: " + finRez + "\n\n");
      return finRez;
   }

   public void searchAll(BigInteger[] rc, BigInteger finRez) {
      int c = takeCandidate();
      int v = takeVoter();
      if (c == 3) searchAll3(finRez, rc, v);
      else if (c == 4) searchAll4(finRez, rc, v);
         else if (c == 5) searchAll5(finRez, rc, v);
            else if (c == 6) searchAll6(finRez, rc, v);
               else if (c == 7) searchAll7(finRez, rc, v);
                  else if (c == 8) searchAll8(finRez, rc, v);
                     else if (c == 9) searchAll9(finRez, rc, v);
                        else if (c == 10) searchAll10(finRez, rc, v);
                           else if (c == 11) searchAll11(finRez, rc, v);
                              else if (c == 12) searchAll12(finRez, rc, v);
                                 else if (c == 13) searchAll13(finRez, rc, v);
                                    else if (c == 14) searchAll14(finRez, rc, v);
                                       else if (c == 15) searchAll15(finRez, rc, v);
                                          else if (c == 16) searchAll16(finRez, rc, v);
                                             else if (c == 17) searchAll17(finRez, rc, v);
```

```
                                else if (c == 18) searchAll18(finRez, rc, v);
                                  else if (c == 19) searchAll19(finRez, rc, v);
                                    else searchAll20(finRez, rc, v);
}

public void findOther(BigInteger[] rc, BigInteger res, int k, int max) {
   long startTime = System.nanoTime();
   BigInteger ai; int ii = 0;
   BigInteger[] min = new BigInteger[2];
   BigInteger[] q = new BigInteger[rc.length*k];
   BigInteger[] r = new BigInteger[rc.length*k];
   BigInteger[] bb = new BigInteger[2];
   BigInteger b = res;
   BigInteger[][] X = new BigInteger[rc.length*2][rc.length*k];
   BigInteger[][] L = new BigInteger[rc.length*2][rc.length*k];
   BigInteger m = BigInteger.ZERO;
   int maxx = max + 1;

   //pradine lygtis
   L[0][0] = res.negate();
   for (int i = 1; i <= max; i++)
      L[0][i] = rc[i-1];
   L[0][maxx] = BigInteger.ZERO;

   ai = BigInteger.ZERO;
   int li = 0, iin = 0;
   int[] in = new int[rc.length*k];

   while (!ai.equals(BigInteger.ONE)) {
      min = findMin(L[li], max);
      ai = min[0];
      ii = min[1].intValue();
      if (ai.equals(BigInteger.ONE) || ai.equals(BigInteger.ZERO))
         break;
      iin = iin + 1;
      in[li] = ii;
      li = li + 1;
      bb = findQR(b, ai);
      X[ii][0] = bb[0];
      L[li][0] = bb[1].negate();
      q[ii-1] = BigInteger.ZERO;
      r[ii-1] = BigInteger.ZERO;
      X[ii][ii] = BigInteger.ONE.negate();
      L[li][ii] = BigInteger.ZERO;
      if (iin > 1)
         maxx = maxx + iin - 1;
      for (int j = 1; j <= maxx; j++) {
         if (j != ii) {
            if (iin > 1) {
               if (j == maxx) {
                  bb = findQR(L[li-1][maxx-1], ai);
                  X[ii][maxx-1] = bb[0].negate();
                  L[li][maxx-1] = bb[1];
                  X[ii][maxx] = BigInteger.ONE.negate();
                  L[li][maxx] = ai.negate();
               }
               if (ii != j && j != maxx) {
                  if (L[li-1][j] == null)
                     L[li-1][j] = BigInteger.ZERO;
                  bb = findQR(L[li-1][j], ai);
                  X[ii][j] = bb[0].negate();
                  L[li][j] = bb[1];
               }
               if (ii == j && j != maxx)
                  X[ii][j] = BigInteger.ONE.negate();
            }
            else {
               bb = findQR(L[li-1][j], ai);
               X[ii][j] = bb[0].negate();
               L[li][j] = bb[1];
               X[ii][maxx] = BigInteger.ONE.negate();
               L[li][maxx] = ai.negate();
            }
         }
      }
      //jei reikia perskaiciuoti kintamuju lygtis
      if (iin > 1) {
         for (int j = 0; j < iin - 1; j++) {
            if (in[j] != 0) {
               m = X[in[j]][ii];
               for (int i = 0; i <= maxx; i++) {
                  if (X[in[j]][i] == null)
                     X[in[j]][i] = BigInteger.ZERO;
                  X[in[j]][i] = m.multiply(X[ii][i]).add(X[in[j]][i]);
               }
            }
         }
      }
      b = L[li][0].negate();
   }

   if (ai.equals(BigInteger.ONE))
      for (int i = 0; i <= maxx; i++)
         X[ii][i] = L[li][i].negate();

   //jei reikia perskaiciuoti kintamuju lygtis
   if (iin > 1) {
   for (int j = 0; j < in.length; j++) {
         if (in[j] != 0) {
            m = X[in[j]][ii];
            for (int i = 0; i <= maxx; i++) {
               if (X[in[j]][i] == null)
                  X[in[j]][i] = BigInteger.ZERO;
               X[in[j]][i] = m.multiply(X[ii][i]).add(X[in[j]][i]);
            }
         }
      }
   }

   //randa x (indeksus), nepriklausancius nuo kitu kintamuju
   int[] cx = new int[maxx+1];   //nepriklauso
   int[] Xno = new int[maxx+1]; //priklauso
   int isC, ci = 0;
   int xni = 0, isZ, g;
   for (int j = 1; j <= max; j++)  {
      isC = 0; g = 0;
      for (int i = 0; i <= maxx; i++) {
         if (X[j][i] == null) {
            X[j][i] = BigInteger.ZERO;
            g = g + 1;
         }
         if (!X[j][i].equals(BigInteger.ZERO)) {
            isC = isC + 1;
            break;
         }
         if (isC == 0 && i == maxx) {
            cx[ci] = j;
            ci = ci + 1;
            break;
         }
      }
   }
   //suranda, jei yra nuliniu ar nepriklausanciu
   int cC = 0;
   for (int i = 1; i <= maxx; i++) {
      isZ = 0;
      for (int j = 1; j <= max; j++) {
         if (i != j) {
            if (X[j][i] == null)
               X[j][i] = BigInteger.ZERO;
            if (i <= max) {
               for (int ij = 0; ij < ci; ij++) {
                  if (i == cx[ij] && isZ == 0) {
                     Xno[xni] = i;
                     xni = xni + 1;
                     if (i <= max)
                        cC = cC + 1;
                     isZ = isZ + 1;
                     break;
                  }
               }
               if (!X[j][i].equals(BigInteger.ZERO) && isZ == 0) {
                  Xno[xni] = i;
                  xni = xni + 1;
                  if (i <= max)
                     cC = cC + 1;
                  isZ = isZ + 1;
                  break;
               }
            }
            else {
               if (!X[j][i].equals(BigInteger.ZERO)) {
                  Xno[xni] = i;
                  xni = xni + 1;
                  if (i <= max)
                     cC = cC + 1;
                  isZ = isZ + 1;
                  break;
               }
            }
         }
      }
   }
}
```

```
    //perraso X
    if (xni > 0) {
        int s;
        for (int j = 1; j <= max; j++)  {
            s = 1;
            for (int ix = 0; ix < xni; ix++) {
                X[j][s] = X[j][Xno[ix]];
                if (s == xni)
                    break;
                s = s + 1;
            }
        }
    }

    maxx = xni;

    //kandidatu skaiciaus masyvas
    int[] kiekC = new int[max+1];
    int kcix = 0;
    for (int i = 0; i < cC; i++)  {
            kiekC[i] = Xno[kcix];
            kcix = kcix + 1;
            if (kcix == cC + 1)
                break;
    }
    //max iki kiek kintamuju tikrinti
    int[] ciMax = new int[maxx+1];
    int iki = k - k / 5;
    for (int i = 0; i < maxx; i++) {
        if (i < cC)
            ciMax[i] = k;
        else
            ciMax[i] = iki;
    }

    //tikrinimui
    int[] snm = new int[maxx];
    for (int i = 0; i < maxx; i++)
        snm[i] = 0;

    if (maxx == 2)
        algIf2 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 3)
        algIf3 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 4)
        algIf4 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 5)
        algIf5 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 6)
        algIf6 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 7)
        algIf7 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 8)
        algIf8 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 9)
        algIf9 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 10)
        algIf10 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 11)
        algIf11 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 12)
        algIf12 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 13)
        algIf13 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 14)
        algIf14 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);
    else if (maxx == 15)
        algIf15 (snm, ciMax, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno);

    int t = takeTime();
    long estimatedTime = System.nanoTime() - startTime;
    double seconds = (double)estimatedTime / 1000000000.0;
    if (seconds > t)
        scrollText.append("\nAlgoritmas sustabdytas. \n");
}

public BigInteger[] findMin(BigInteger[] x, int max) {
    BigInteger[] minim = new BigInteger[2];
    BigInteger mini = x[1];
    minim[0] = mini.abs(); minim[1] = BigInteger.ONE;
    if (mini.equals(BigInteger.ZERO))
        for (int i = 2; i <= max; i++)
            if (!x[i].equals(BigInteger.ZERO)) {
                mini = x[i].abs();
                minim[0] = mini;
                minim[1] = BigInteger.valueOf(i);
            }
    for (int i = 2; i < max; i++) {
        if (!x[i].equals(BigInteger.ZERO)) {
            mini = mini.min(x[i]);
            if (!minim[0].equals(mini)) {
                minim[0] = mini.abs();
                minim[1] = BigInteger.valueOf(i);
            }
        }
    }
    return minim;
}

public BigInteger[] findQR (BigInteger val1, BigInteger val2) {
    BigInteger[] qr = new BigInteger[2];
    if (val1.compareTo(BigInteger.ZERO) >= 0 && val2.compareTo(BigInteger.ZERO) >= 0)
        qr = val1.divideAndRemainder(val2);
    else {
        qr = val1.divideAndRemainder(val2);
        qr[0] = qr[0].subtract(BigInteger.ONE);
        qr[1] = val1.add(qr[0].multiply(val2.negate()));
    }
    return qr;
}

public void findRes3 (BigInteger res, BigInteger[] rc, int v, int c) {
    long startTime = System.nanoTime();
    BigInteger s = BigInteger.ZERO;
    BigInteger[] qr = new BigInteger[2];
    BigInteger[] q = new BigInteger[c];
    BigInteger[] r = new BigInteger[c];
    qr = findQR(res, rc[c-1]);
    q[c-1] = qr[0]; r[c-1] = qr[1];
    for (int i = c - 1; i > 0; i--) {
        qr = findQR(r[i], rc[i-1]);
        q[i-1] = qr[0]; r[i-1] = qr[1];
    }
    if (printCount.isSelected() == false) {
        scrollText.append("Kandidato gautų balsų skaičius: \n");
        for (int i = 1; i <= c; i++)
            scrollText.append(i + " k.\t");
        scrollText.append("\n");
        for (int i = 0; i < c; i++)
            scrollText.append(q[i] + "\t");
        scrollText.append("\n");
    }

    long estimatedTime = System.nanoTime() - startTime;
    double seconds = (double)estimatedTime / 1000000000.0;
    scrollText.append("\nSveikųjų skaičių dalybos laikas: " + seconds + "
s.\n\n\n");
    }
```

```
    public boolean findAlgRes (int[] snm, int[] cx, BigInteger res, int max, int
maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[]
Xno) {
        BigInteger morRez = BigInteger.ZERO;
        BigInteger morR = BigInteger.ZERO;
        int stop = 0;
        BigInteger[] resu = new BigInteger[max+1];
        BigInteger resKiek = BigInteger.ZERO;
        for (int j = 1; j <= max; j++) {
            resu[j] = BigInteger.ZERO;
            if (cC > 0) {
                stop = 0;
                int sumsnm = 0;
                if (snm[cC] > 0)
                    for (int i = 0; i < cC; i++)
                        sumsnm = sumsnm + snm[i];
                if (sumsnm > k)
                    break;
                part: for (int kci = 0; kci < max; kci++) {
                    for (int cxi = 0; cxi < ci; cxi++) {
                        if (cx[cxi] == kiekC[kci] && j == kiekC[kci]) {
                            for (int i = 0; i < cC; i++)
                                if (j == Xno[i]) {
                                    resu[j] = BigInteger.valueOf(snm[i]);
                                    stop = 1;
                                    break part;
                                }
                        }
                    }
                }
                if (stop == 0) {
                    if (cC == max)
                        for (int oj = 1; oj <= maxx; oj++) {
                            if (j != oj)
                                resu[j] =
resu[j].add(X[j][oj].multiply(BigInteger.valueOf(snm[oj-1])));
                        }
                    else
                        for (int i = 0; i < maxx; i++)
                            if (j != Xno[i])
                                resu[j] =
resu[j].add(X[j][i+1].multiply(BigInteger.valueOf(snm[i])));
                    resu[j] = resu[j].add(X[j][0]);
                }
            }
            else {
                for (int i = 0; i < maxx; i++) {
                    resu[j] = resu[j].add(X[j][i+1].multiply(BigInteger.valueOf(snm[i])));
                }
                resu[j] = resu[j].add(X[j][0]);
            }
            resKiek = resKiek.add(resu[j]);
            if (resKiek.compareTo(BigInteger.valueOf(k)) > 0 ||
resKiek.compareTo(BigInteger.ZERO) < 0)
                break;
        }
        if (resKiek.equals(BigInteger.valueOf(k))) {
            for (int rk = 1; rk <= max; rk++) {
                morR = rc[rk-1].multiply(resu[rk]);
                morRez = morRez.add(morR);
            }
            if (morRez.equals(res)) {
                if (printCount.isSelected() == false) {
                    scrollText.append("Kandidato gautų balsų skaičius: \n");
                    for (int rk = 1; rk <= max; rk++)
                        scrollText.append(rk + " k.\t");
                    scrollText.append("\n");
                    for (int rk = 1; rk <= max; rk++)
                        scrollText.append(resu[rk] + "\t");
                    scrollText.append("\n");
                }
                return true;
            }
        }
        return false;
    }
```

```
    public void algIf2 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
        int t = takeTime();
        long startTime = System.nanoTime();
        all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
            for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
                if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                    long estimatedTime = System.nanoTime() - startTime;
                    double seconds = (double)estimatedTime / 1000000000.0;
                    scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                    break all;
                }
                else {
                    long estimatedTime = System.nanoTime() - startTime;
                    double seconds = (double)estimatedTime / 1000000000.0;
                    if (seconds > t)
                        break all;
                } } } }

    public void algIf3 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
        int t = takeTime();
        long startTime = System.nanoTime();
        all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
            for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
                for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
                    if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                        long estimatedTime = System.nanoTime() - startTime;
                        double seconds = (double)estimatedTime / 1000000000.0;
                        scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                        break all;
                    }
                    else {
                        long estimatedTime = System.nanoTime() - startTime;
                        double seconds = (double)estimatedTime / 1000000000.0;
                        if (seconds > t)
                            break all;
                    } } } } }

    public void algIf4 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
        int t = takeTime();
        long startTime = System.nanoTime();
        all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
            for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
                for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
                    for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
                        if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                            long estimatedTime = System.nanoTime() - startTime;
                            double seconds = (double)estimatedTime / 1000000000.0;
                            scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                            break all;
```

```
            }
            else {
               long estimatedTime = System.nanoTime() - startTime;
               double seconds = (double)estimatedTime / 1000000000.0;
               if (seconds > t)
                  break all;
            }}}}}

public void algIf5 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
   int t = takeTime();
   long startTime = System.nanoTime();
   all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
      for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
         for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
            for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
               for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                  if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                     long estimatedTime = System.nanoTime() - startTime;
                     double seconds = (double)estimatedTime / 1000000000.0;
                     scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                     break all;
                  }
                  else {
                     long estimatedTime = System.nanoTime() - startTime;
                     double seconds = (double)estimatedTime / 1000000000.0;
                     if (seconds > t)
                        break all;
                  }}}}}}}

public void algIf6 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
   int t = takeTime();
   long startTime = System.nanoTime();
   all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
      for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
         for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
            for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
               for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                  for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                     if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                        long estimatedTime = System.nanoTime() - startTime;
                        double seconds = (double)estimatedTime / 1000000000.0;
                        scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                        break all;
                     }
                     else {
                        long estimatedTime = System.nanoTime() - startTime;
                        double seconds = (double)estimatedTime / 1000000000.0;
                        if (seconds > t)
                           break all;
                     }}}}}}}}

public void algIf7 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
   int t = takeTime();
   long startTime = System.nanoTime();
      all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
         for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
            for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
               for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
                  for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                     for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                        for (snm[6] = 0; snm[6] <= ciMax[6]; snm[6]++) {
                           if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                              long estimatedTime = System.nanoTime() - startTime;
                              double seconds = (double)estimatedTime / 1000000000.0;
                              scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                              break all;
                           }
                           else {
                              long estimatedTime = System.nanoTime() - startTime;
                              double seconds = (double)estimatedTime / 1000000000.0;
                              if (seconds > t)
                                 break all;
                           }}}}}}}}}

public void algIf8 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
   int t = takeTime();
   long startTime = System.nanoTime();
   all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
      for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
         for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
            for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
               for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                  for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                     for (snm[6] = 0; snm[6] <= ciMax[6]; snm[6]++) {
                        for (snm[7] = 0; snm[7] <= ciMax[7]; snm[7]++) {
                           if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                              long estimatedTime = System.nanoTime() - startTime;
                              double seconds = (double)estimatedTime / 1000000000.0;
                              scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
```

```
                        break all;
                    }
                    else {
                        long estimatedTime = System.nanoTime() - startTime;
                        double seconds = (double)estimatedTime / 1000000000.0;
                        if (seconds > t)
                            break all;
                    }}}}}}}}}}

public void algIf9 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
    int t = takeTime();
    long startTime = System.nanoTime();
    all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
        for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
            for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
                for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
                    for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                        for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                            for (snm[6] = 0; snm[6] <= ciMax[6]; snm[6]++) {
                                for (snm[7] = 0; snm[7] <= ciMax[7]; snm[7]++) {
                                    for (snm[8] = 0; snm[8] <= ciMax[8]; snm[8]++) {
                                        if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                                            long estimatedTime = System.nanoTime() - startTime;
                                            double seconds = (double)estimatedTime / 1000000000.0;
                                            scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                                            break all;
                                        }
                                        else {
                                            long estimatedTime = System.nanoTime() - startTime;
                                            double seconds = (double)estimatedTime / 1000000000.0;
                                            if (seconds > t)
                                                break all;
                                        }}}}}}}}}}}

public void algIf10 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
    int t = takeTime();
    long startTime = System.nanoTime();
    all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
        for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
            for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
                for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
                    for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                        for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                            for (snm[6] = 0; snm[6] <= ciMax[6]; snm[6]++) {
                                for (snm[7] = 0; snm[7] <= ciMax[7]; snm[7]++) {
                                    for (snm[8] = 0; snm[8] <= ciMax[8]; snm[8]++) {
                                        for (snm[9] = 0; snm[9] <= ciMax[9]; snm[9]++) {
                                            if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                                                long estimatedTime = System.nanoTime() - startTime;
                                                double seconds = (double)estimatedTime / 1000000000.0;
                                                scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                                                break all;
                                            }
                                            else {
                                                long estimatedTime = System.nanoTime() - startTime;
                                                double seconds = (double)estimatedTime / 1000000000.0;
                                                if (seconds > t)
                                                    break all;
                                            }}}}}}}}}}}}

public void algIf11 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
    int t = takeTime();
    long startTime = System.nanoTime();
    all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
        for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
            for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
                for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
                    for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                        for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                            for (snm[6] = 0; snm[6] <= ciMax[6]; snm[6]++) {
                                for (snm[7] = 0; snm[7] <= ciMax[7]; snm[7]++) {
                                    for (snm[8] = 0; snm[8] <= ciMax[8]; snm[8]++) {
                                        for (snm[9] = 0; snm[9] <= ciMax[9]; snm[9]++) {
                                            for (snm[10] = 0; snm[10] <= ciMax[10]; snm[10]++) {
                                                if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                                                    long estimatedTime = System.nanoTime() - startTime;
                                                    double seconds = (double)estimatedTime / 1000000000.0;
                                                    scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                                                    break all;
                                                }
                                                else {
                                                    long estimatedTime = System.nanoTime() - startTime;
                                                    double seconds = (double)estimatedTime / 1000000000.0;
                                                    if (seconds > t)
                                                        break all;
                                                }}}}}}}}}}}}}

public void algIf12 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
    int t = takeTime();
```

```
    long startTime = System.nanoTime();
    all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
       for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
          for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
             for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
                for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                   for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                      for (snm[6] = 0; snm[6] <= ciMax[6]; snm[6]++) {
                         for (snm[7] = 0; snm[7] <= ciMax[7]; snm[7]++) {
                            for (snm[8] = 0; snm[8] <= ciMax[8]; snm[8]++) {
                               for (snm[9] = 0; snm[9] <= ciMax[9]; snm[9]++) {
                                  for (snm[10] = 0; snm[10] <= ciMax[10]; snm[10]++) {
                                     for (snm[11] = 0; snm[11] <= ciMax[11]; snm[11]++) {
                                        if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                                           long estimatedTime = System.nanoTime() - startTime;
                                           double seconds = (double)estimatedTime / 1000000000.0;
                                           scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                                           break all;
                                        }
                                        else {
                                           long estimatedTime = System.nanoTime() - startTime;
                                           double seconds = (double)estimatedTime / 1000000000.0;
                                           if (seconds > t)
                                              break all;
                                        }}}}}}}}}}}}}}

public void algIf13 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
    int t = takeTime();
    long startTime = System.nanoTime();
    all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
       for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
          for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
             for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
                for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                   for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                      for (snm[6] = 0; snm[6] <= ciMax[6]; snm[6]++) {
                         for (snm[7] = 0; snm[7] <= ciMax[7]; snm[7]++) {
                            for (snm[8] = 0; snm[8] <= ciMax[8]; snm[8]++) {
                               for (snm[9] = 0; snm[9] <= ciMax[9]; snm[9]++) {
                                  for (snm[10] = 0; snm[10] <= ciMax[10]; snm[10]++) {
                                     for (snm[11] = 0; snm[11] <= ciMax[11]; snm[11]++) {
                                        for (snm[12] = 0; snm[12] <= ciMax[12]; snm[12]++) {
                                           if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                                              long estimatedTime = System.nanoTime() - startTime;
                                              double seconds = (double)estimatedTime / 1000000000.0;
                                              scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                                              break all;
                                           }
                                           else {
                                              long estimatedTime = System.nanoTime() - startTime;
                                              double seconds = (double)estimatedTime / 1000000000.0;
                                              if (seconds > t)
                                                 break all;
                                           }}}}}}}}}}}}}}}

public void algIf14 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
    int t = takeTime();
    long startTime = System.nanoTime();
    all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
       for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
          for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
             for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
                for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                   for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                      for (snm[6] = 0; snm[6] <= ciMax[6]; snm[6]++) {
                         for (snm[7] = 0; snm[7] <= ciMax[7]; snm[7]++) {
                            for (snm[8] = 0; snm[8] <= ciMax[8]; snm[8]++) {
                               for (snm[9] = 0; snm[9] <= ciMax[9]; snm[9]++) {
                                  for (snm[10] = 0; snm[10] <= ciMax[10]; snm[10]++) {
                                     for (snm[11] = 0; snm[11] <= ciMax[11]; snm[11]++) {
                                        for (snm[12] = 0; snm[12] <= ciMax[12]; snm[12]++) {
                                           for (snm[13] = 0; snm[13] <= ciMax[13]; snm[13]++) {
                                              if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                                                 long estimatedTime = System.nanoTime() - startTime;
                                                 double seconds = (double)estimatedTime / 1000000000.0;
                                                 scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                                                 break all;
                                              }
                                              else {
                                                 long estimatedTime = System.nanoTime() - startTime;
                                                 double seconds = (double)estimatedTime / 1000000000.0;
                                                 if (seconds > t)
                                                    break all;
                                              }}}}}}}}}}}}}}}}

public void algIf15 (int[] snm, int[] ciMax, int[] cx, BigInteger res, int max, int maxx, int ci, int[] kiekC, BigInteger[][] X, BigInteger[] rc, int k, int cC, int[] Xno) {
    int t = takeTime();
    long startTime = System.nanoTime();
    all: for (snm[0] = 0; snm[0] <= ciMax[0]; snm[0]++) {
```

```
            for (snm[1] = 0; snm[1] <= ciMax[1]; snm[1]++) {
               for (snm[2] = 0; snm[2] <= ciMax[2]; snm[2]++) {
                  for (snm[3] = 0; snm[3] <= ciMax[3]; snm[3]++) {
                     for (snm[4] = 0; snm[4] <= ciMax[4]; snm[4]++) {
                        for (snm[5] = 0; snm[5] <= ciMax[5]; snm[5]++) {
                           for (snm[6] = 0; snm[6] <= ciMax[6]; snm[6]++) {
                              for (snm[7] = 0; snm[7] <= ciMax[7]; snm[7]++) {
                                 for (snm[8] = 0; snm[8] <= ciMax[8]; snm[8]++) {
                                    for (snm[9] = 0; snm[9] <= ciMax[9]; snm[9]++) {
                                       for (snm[10] = 0; snm[10] <= ciMax[10]; snm[10]++) {
                                          for (snm[11] = 0; snm[11] <= ciMax[11]; snm[11]++) {
                                             for (snm[12] = 0; snm[12] <= ciMax[12]; snm[12]++) {
                                                for (snm[13] = 0; snm[13] <= ciMax[13]; snm[13]++) {
                                                   for (snm[14] = 0; snm[14] <= ciMax[14]; snm[14]++) {
                                                      if (findAlgRes(snm, cx, res, max, maxx, ci, kiekC, X, rc, k, cC, Xno) == true) {
                                                         long estimatedTime = System.nanoTime() - startTime;
                                                         double seconds = (double)estimatedTime / 1000000000.0;
                                                         scrollText.append("\n Algoritmo laikas: " + seconds + " s.\n\n");
                                                         break all;
                                                      }
                                                      else {
                                                         long estimatedTime = System.nanoTime() - startTime;
                                                         double seconds = (double)estimatedTime / 1000000000.0;
                                                         if (seconds > t)
                                                            break all;
                                                      }}}}}}}}}}}}}}}}}}}}

   public void searchAll3 (BigInteger finRez, BigInteger[] rc, int v) {
      int t = takeTime();
      long startTime = System.nanoTime();
      BigInteger morRez = new BigInteger("0");
      all: for (int c1 = 0; c1 < v + 1; c1++) {
         for (int c2 = 0; c2 < v + 1; c2++) {
            int c3 = v - c1 - c2;
            if (c3 >= 0) {
               morRez = (rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3)));
               if (morRez.equals(finRez)) {
                  if (printCount.isSelected() == false) {
                     scrollText.append("Kandidato gautų balsų skaičius: \n");
                     scrollText.append("1 k.\t2 k.\t3 k.\n");
                     scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\n");
                  }
                  break all;
               }
               else {
                  long estimatedTime = System.nanoTime() - startTime;
                  double seconds = (double)estimatedTime / 1000000000.0;
                  if (seconds > t)
                     break all;
               }}}}
      long estimatedTime = System.nanoTime() - startTime;
      double seconds = (double)estimatedTime / 1000000000.0;
      scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
   }

   public void searchAll4 (BigInteger finRez, BigInteger[] rc, int v) {
      int t = takeTime();
      long startTime = System.nanoTime();
      BigInteger morRez = new BigInteger("0");
      all: for (int c1 = 0; c1 < v + 1; c1++) {
         for (int c2 = 0; c2 < v + 1; c2++) {
            for (int c3 = 0; c3 < v + 1; c3++) {
               int c4 = v - c1 - c2 - c3;
               if (c4 >= 0) {
                  morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4)));
                  if (morRez.equals(finRez)) {
                     if (printCount.isSelected() == false) {
                        scrollText.append("Kandidato gautų balsų skaičius: \n");
                        scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\n");
                        scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\n");
                     }
                     break all;
                  }
                  else {
                     long estimatedTime = System.nanoTime() - startTime;
                     double seconds = (double)estimatedTime / 1000000000.0;
                     if (seconds > t)
                        break all;
                  }}}}}
      long estimatedTime = System.nanoTime() - startTime;
      double seconds = (double)estimatedTime / 1000000000.0;
      scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
   }

   public void searchAll5 (BigInteger finRez, BigInteger[] rc, int v) {
      int t = takeTime();
      long startTime = System.nanoTime();
```

```
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        int c5 = v - c1 - c2 - c3 - c4;
                        if (c5 >= 0) {
                            morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5)));
                            if (morRez.equals(finRez)) {
                                if (printCount.isSelected() == false) {
                                    scrollText.append("Kandidato gautų balsų skaičius: \n");
                                    scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\n");
                                    scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\n");
                                }
                                break all;
                            }
                            else {
                                long estimatedTime = System.nanoTime() - startTime;
                                double seconds = (double)estimatedTime / 1000000000.0;
                                if (seconds > t)
                                    break all;
                            } } } } } }
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll6 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime();
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            int c6 = v - c1 - c2 - c3 - c4 - c5;
                            if (c6 >= 0) {
                                morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6)));
                                if (morRez.equals(finRez)) {
                                    if (printCount.isSelected() == false) {
                                        scrollText.append("Kandidato gautų balsų skaičius: \n");
                                        scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\n");
                                        scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\n");
                                    }
                                    break all;
                                }
                                else {
                                    long estimatedTime = System.nanoTime() - startTime;
                                    double seconds = (double)estimatedTime / 1000000000.0;
                                    if (seconds > t)
                                        break all;
                                } } } } } } }
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll7 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime();
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                int c7 = v - c1 - c2 - c3 - c4 - c5 - c6;
                                if (c7 >= 0) {
                                    morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7)));
                                    if (morRez.equals(finRez)) {
                                        if (printCount.isSelected() == false) {
                                            scrollText.append("Kandidato gautų balsų skaičius: \n");
                                            scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\n");
                                            scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\n");
                                        }
                                        break all;
                                    }
                                    else {
                                        long estimatedTime = System.nanoTime() - startTime;
                                        double seconds = (double)estimatedTime / 1000000000.0;
                                        if (seconds > t)
```

```
                                break all;
                            } } } } } } } }
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll8 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime();
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    int c8 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7;
                                    if (c8 >= 0) {
                                        morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8)));
                                        if (morRez.equals(finRez)) {
                                            if (printCount.isSelected() == false) {
                                                scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\n");
                                                scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\n");
                                            }
                                            break all;
                                        }
                                        else {
                                            long estimatedTime = System.nanoTime() - startTime;
                                            double seconds = (double)estimatedTime / 1000000000.0;
                                            if (seconds > t)
                                                break all;
                                        } } } } } } } } }
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll9 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime();
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        int c9 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8;
                                        if (c9 >= 0) {
                                            morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9)));
                                            if (morRez.equals(finRez)) {
                                                if (printCount.isSelected() == false) {
                                                    scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                    scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\n");
                                                    scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 + "\n");
                                                }
                                                break all;
                                            }
                                            else {
                                                long estimatedTime = System.nanoTime() - startTime;
                                                double seconds = (double)estimatedTime / 1000000000.0;
                                                if (seconds > t)
                                                    break all;
                                            } } } } } } } } } }
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll10 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime();
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
```

```
                    for (int c6 = 0; c6 < v + 1; c6++) {
                        for (int c7 = 0; c7 < v + 1; c7++) {
                            for (int c8 = 0; c8 < v + 1; c8++) {
                                for (int c9 = 0; c9 < v + 1; c9++) {
                                    int c10 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9;
                                    if (c10 >= 0) {
                                        morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10)));
                                        if (morRez.equals(finRez)) {
                                            if (printCount.isSelected() == false) {
                                                scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\n");
                                                scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 + "\t" + c10 + "\n");
                                            }
                                            break all;
                                        }
                                        else {
                                            long estimatedTime = System.nanoTime() - startTime;
                                            double seconds = (double)estimatedTime / 1000000000.0;
                                            if (seconds > t)
                                                break all;
                                        }}}}}}}}}}}
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll11 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime();
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        for (int c9 = 0; c9 < v + 1; c9++) {
                                            for (int c10 = 0; c10 < v + 1; c10++) {
                                                int c11 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10;
                                                if (c11 >= 0) {
                                                    morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11)));
                                                    if (morRez.equals(finRez)) {
                                                        if (printCount.isSelected() == false) {
                                                            scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                            scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\n");
                                                            scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 + "\t" + c10 + "\t" +
c11 + "\n");
                                                        }
                                                        break all;
                                                    }
                                                    else {
                                                        long estimatedTime = System.nanoTime() - startTime;
                                                        double seconds = (double)estimatedTime / 1000000000.0;
                                                        if (seconds > t)
                                                            break all;
                                                    }}}}}}}}}}}}
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll12 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime();
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        for (int c9 = 0; c9 < v + 1; c9++) {
                                            for (int c10 = 0; c10 < v + 1; c10++) {
                                                for (int c11 = 0; c11 < v + 1; c11++) {
                                                    int c12 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10 - c11;
                                                    if (c12 >= 0) {
                                                        morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
```

```
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11))).add(rc[11].m
ultiply(BigInteger.valueOf(c12)));
                                        if (morRez.equals(finRez)) {
                                            if (printCount.isSelected() == false) {
                                                scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\t12 k.\n");
                                                scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 + "\t" + c10 + "\t"
+ c11 + "\t" + c12 + "\n");
                                            }
                                            break all;
                                        }
                                        else {
                                            long estimatedTime = System.nanoTime() - startTime;
                                            double seconds = (double)estimatedTime / 1000000000.0;
                                            if (seconds > t)
                                                break all;
                                        } } } } } } } } } } }
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll13 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime(); int stop = 0;
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        for (int c9 = 0; c9 < v + 1; c9++) {
                                            for (int c10 = 0; c10 < v + 1; c10++) {
                                                for (int c11 = 0; c11 < v + 1; c11++) {
                                                    for (int c12 = 0; c12 < v + 1; c12++) {
                                                        int c13 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10 - c11 - c12;
                                                        if (c13 >= 0) {
                                                            morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11))).add(rc[11].m
ultiply(BigInteger.valueOf(c12))).add(rc[12].multiply(BigInteger.valueOf(c13)));
                                                            if (morRez.equals(finRez)) {
                                                                if (printCount.isSelected() == false) {
                                                                    scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                                    scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\t12 k.\t13 k.\n");
                                                                    scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 + "\t" + c10 +
"\t" + c11 + "\t" + c12 + "\t" + c13 + "\n");
                                                                }
                                                                break all;
                                                            }
                                                            else {
                                                                long estimatedTime = System.nanoTime() - startTime;
                                                                double seconds = (double)estimatedTime / 1000000000.0;
                                                                if (seconds > t)
                                                                    break all;
                                                            } } } } } } } } } } } } }
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll14 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime(); int stop = 0;
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        for (int c9 = 0; c9 < v + 1; c9++) {
                                            for (int c10 = 0; c10 < v + 1; c10++) {
                                                for (int c11 = 0; c11 < v + 1; c11++) {
                                                    for (int c12 = 0; c12 < v + 1; c12++) {
                                                        for (int c13 = 0; c13 < v + 1; c13++) {
                                                            int c14 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10 - c11 - c12 - c13;
                                                            if (c14 >= 0) {
                                                                morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11))).add(rc[11].m
ultiply(BigInteger.valueOf(c12))).add(rc[12].multiply(BigInteger.valueOf(c13))).add(rc[13].multiply(BigInteger.valueOf(c14)));
```

```
                                        if (morRez.equals(finRez)) {
                                            if (printCount.isSelected() == false) {
                                                scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\t12 k.\t13 k.\t14 k.\n");
                                                scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 + "\t" + c10
+ "\t" + c11 + "\t" + c12 + "\t" + c13 + "\t" + c14 + "\n");
                                            }
                                            break all;
                                        }
                                        else {
                                            long estimatedTime = System.nanoTime() - startTime;
                                            double seconds = (double)estimatedTime / 1000000000.0;
                                            if (seconds > t)
                                                break all;
                                        }}}}}}}}}}}}}}}
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll15 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime(); int stop = 0;
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        for (int c9 = 0; c9 < v + 1; c9++) {
                                            for (int c10 = 0; c10 < v + 1; c10++) {
                                                for (int c11 = 0; c11 < v + 1; c11++) {
                                                    for (int c12 = 0; c12 < v + 1; c12++) {
                                                        for (int c13 = 0; c13 < v + 1; c13++) {
                                                            for (int c14 = 0; c14 < v + 1; c14++) {
                                                                int c15 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10 - c11 - c12 - c13 - c14;
                                                                if (c15 >= 0) {
                                                                    morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11))).add(rc[11].m
ultiply(BigInteger.valueOf(c12))).add(rc[12].multiply(BigInteger.valueOf(c13))).add(rc[13].multiply(BigInteger.valueOf(c14))).add(rc[14].multiply(BigInteger.valueOf
(c15)));
                                                                    if (morRez.equals(finRez)) {
                                                                        if (printCount.isSelected() == false) {
                                                                            scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                                            scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\t12 k.\t13 k.\t14 k.\t15 k.\n");
                                                                            scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 + "\t" +
c10 + "\t" + c11 + "\t" + c12 + "\t" + c13 + "\t" + c14 + "\t" + c15 + "\n");
                                                                        }
                                                                        break all;
                                                                    }
                                                                    else {
                                                                        long estimatedTime = System.nanoTime() - startTime;
                                                                        double seconds = (double)estimatedTime / 1000000000.0;
                                                                        if (seconds > t)
                                                                            break all;
                                                                    }}}}}}}}}}}}}}}}}
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll16 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime(); int stop = 0;
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        for (int c9 = 0; c9 < v + 1; c9++) {
                                            for (int c10 = 0; c10 < v + 1; c10++) {
                                                for (int c11 = 0; c11 < v + 1; c11++) {
                                                    for (int c12 = 0; c12 < v + 1; c12++) {
                                                        for (int c13 = 0; c13 < v + 1; c13++) {
                                                            for (int c14 = 0; c14 < v + 1; c14++) {
                                                                for (int c15 = 0; c15 < v + 1; c15++) {
                                                                    int c16 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10 - c11 - c12 - c13 - c14 - c15;
                                                                    if (c16 >= 0) {
                                                                        morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
```

```
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteger.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11))).add(rc[11].multiply(BigInteger.valueOf(c12))).add(rc[12].multiply(BigInteger.valueOf(c13))).add(rc[13].multiply(BigInteger.valueOf(c14))).add(rc[14].multiply(BigInteger.valueOf(c15))).add(rc[15].multiply(BigInteger.valueOf(c16)));
                                                    if (morRez.equals(finRez)) {
                                                        if (printCount.isSelected() == false) {
                                                            scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                            scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\t12 k.\t13 k.\t14 k.\t15 k.\t16 k.\n");
                                                            scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 + "\t" + c10 + "\t" + c11 + "\t" + c12 + "\t" + c13 + "\t" + c14 + "\t" + c15 + "\t" + c16 + "\n");
                                                        }
                                                        break all;
                                                    }
                                                    else {
                                                        long estimatedTime = System.nanoTime() - startTime;
                                                        double seconds = (double)estimatedTime / 1000000000.0;
                                                        if (seconds > t)
                                                            break all;
                                                    }}}}}}}}}}}}}}}}}}}
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll17 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime(); int stop = 0;
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        for (int c9 = 0; c9 < v + 1; c9++) {
                                            for (int c10 = 0; c10 < v + 1; c10++) {
                                                for (int c11 = 0; c11 < v + 1; c11++) {
                                                    for (int c12 = 0; c12 < v + 1; c12++) {
                                                        for (int c13 = 0; c13 < v + 1; c13++) {
                                                            for (int c14 = 0; c14 < v + 1; c14++) {
                                                                for (int c15 = 0; c15 < v + 1; c15++) {
                                                                    for (int c16 = 0; c16 < v + 1; c16++) {
                                                                        int c17 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10 - c11 - c12 - c13 - c14 - c15 - c16;
                                                                        if (c17 >= 0) {
                                                                            morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteger.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11))).add(rc[11].multiply(BigInteger.valueOf(c12))).add(rc[12].multiply(BigInteger.valueOf(c13))).add(rc[13].multiply(BigInteger.valueOf(c14))).add(rc[14].multiply(BigInteger.valueOf(c15))).add(rc[15].multiply(BigInteger.valueOf(c16))).add(rc[16].multiply(BigInteger.valueOf(c17)));
                                                                            if (morRez.equals(finRez)) {
                                                                                if (printCount.isSelected() == false) {
                                                                                    scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                                                    scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\t12 k.\t13 k.\t14 k.\t15 k.\t16 k.\t17 k.\t\n");
                                                                                    scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 + "\t" + c10 + "\t" + c11 + "\t" + c12 + "\t" + c13 + "\t" + c14 + "\t" + c15 + "\t" + c16 + "\t" + c17 + "\n");
                                                                                }
                                                                                break all;
                                                                            }
                                                                            else {
                                                                                long estimatedTime = System.nanoTime() - startTime;
                                                                                double seconds = (double)estimatedTime / 1000000000.0;
                                                                                if (seconds > t)
                                                                                    break all;
                                                                            }}}}}}}}}}}}}}}}}}}}
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll18 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime(); int stop = 0;
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        for (int c9 = 0; c9 < v + 1; c9++) {
                                            for (int c10 = 0; c10 < v + 1; c10++) {
                                                for (int c11 = 0; c11 < v + 1; c11++) {
                                                    for (int c12 = 0; c12 < v + 1; c12++) {
```

```
                                    for (int c13 = 0; c13 < v + 1; c13++) {
                                        for (int c14 = 0; c14 < v + 1; c14++) {
                                            for (int c15 = 0; c15 < v + 1; c15++) {
                                                for (int c16 = 0; c16 < v + 1; c16++) {
                                                    for (int c17 = 0; c17 < v + 1; c17++) {
                                                        int c18 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10 - c11 - c12 - c13 - c14 - c15 - c16 - c17;
                                                        if (c18 >= 0) {
                                                            morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11))).add(rc[11].m
ultiply(BigInteger.valueOf(c12))).add(rc[12].multiply(BigInteger.valueOf(c13))).add(rc[13].multiply(BigInteger.valueOf(c14))).add(rc[14].multiply(BigInteger.valueOf
(c15))).add(rc[15].multiply(BigInteger.valueOf(c16))).add(rc[16].multiply(BigInteger.valueOf(c17))).add(rc[17].multiply(BigInteger.valueOf(c18)));
                                                            if (morRez.equals(finRez)) {
                                                                if (printCount.isSelected() == false) {
                                                                    scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                                    scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\t12 k.\t13 k.\t14 k.\t15 k.\t16
k.\t17 k.\t18 k.\n");
                                                                    scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9 +
"\t" + c10 + "\t" + c11 + "\t" + c12 + "\t" + c13 + "\t" + c14 + "\t" + c15 + "\t" + c16 + "\t" + c17 + "\t" + c18 + "\n");
                                                                }
                                                                break all;
                                                            }
                                                            else {
                                                                long estimatedTime = System.nanoTime() - startTime;
                                                                double seconds = (double)estimatedTime / 1000000000.0;
                                                                if (seconds > t)
                                                                    break all;
                                                            }}}}}}}}}}}}}}}}}}}
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll19 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime(); int stop = 0;
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
            for (int c2 = 0; c2 < v + 1; c2++) {
                for (int c3 = 0; c3 < v + 1; c3++) {
                    for (int c4 = 0; c4 < v + 1; c4++) {
                        for (int c5 = 0; c5 < v + 1; c5++) {
                            for (int c6 = 0; c6 < v + 1; c6++) {
                                for (int c7 = 0; c7 < v + 1; c7++) {
                                    for (int c8 = 0; c8 < v + 1; c8++) {
                                        for (int c9 = 0; c9 < v + 1; c9++) {
                                            for (int c10 = 0; c10 < v + 1; c10++) {
                                                for (int c11 = 0; c11 < v + 1; c11++) {
                                                    for (int c12 = 0; c12 < v + 1; c12++) {
                                                        for (int c13 = 0; c13 < v + 1; c13++) {
                                                            for (int c14 = 0; c14 < v + 1; c14++) {
                                                                for (int c15 = 0; c15 < v + 1; c15++) {
                                                                    for (int c16 = 0; c16 < v + 1; c16++) {
                                                                        for (int c17 = 0; c17 < v + 1; c17++) {
                                                                            for (int c18 = 0; c18 < v + 1; c18++) {
                                                                                int c19 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10 - c11 - c12 - c13 - c14 - c15 - c16 - c17 - c18;
                                                                                if (c19 >= 0) {
                                                                                    morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11))).add(rc[11].m
ultiply(BigInteger.valueOf(c12))).add(rc[12].multiply(BigInteger.valueOf(c13))).add(rc[13].multiply(BigInteger.valueOf(c14))).add(rc[14].multiply(BigInteger.valueOf
(c15))).add(rc[15].multiply(BigInteger.valueOf(c16))).add(rc[16].multiply(BigInteger.valueOf(c17))).add(rc[17].multiply(BigInteger.valueOf(c18))).add(rc[18].multipl
y(BigInteger.valueOf(c19)));
                                                                                    if (morRez.equals(finRez)) {
                                                                                        if (printCount.isSelected() == false) {
                                                                                            scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                                                            scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\t12 k.\t13 k.\t14 k.\t15 k.\t16
k.\t17 k.\t18 k.\t19 k.\n");
                                                                                            scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" + c9
+ "\t" + c10 + "\t" + c11 + "\t" + c12 + "\t" + c13 + "\t" + c14 + "\t" + c15 + "\t" + c16 + "\t" + c17 + "\t" + c18 + "\t" + c19 + "\n");
                                                                                        }
                                                                                        break all;
                                                                                    }
                                                                                    else {
                                                                                        long estimatedTime = System.nanoTime() - startTime;
                                                                                        double seconds = (double)estimatedTime / 1000000000.0;
                                                                                        if (seconds > t)
                                                                                            break all;
                                                                                    }}}}}}}}}}}}}}}}}}}}}
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
    }

    public void searchAll20 (BigInteger finRez, BigInteger[] rc, int v) {
        int t = takeTime();
        long startTime = System.nanoTime(); int stop = 0;
```

```
        BigInteger morRez = new BigInteger("0");
        all: for (int c1 = 0; c1 < v + 1; c1++) {
           for (int c2 = 0; c2 < v + 1; c2++) {
              for (int c3 = 0; c3 < v + 1; c3++) {
                 for (int c4 = 0; c4 < v + 1; c4++) {
                    for (int c5 = 0; c5 < v + 1; c5++) {
                       for (int c6 = 0; c6 < v + 1; c6++) {
                          for (int c7 = 0; c7 < v + 1; c7++) {
                             for (int c8 = 0; c8 < v + 1; c8++) {
                                for (int c9 = 0; c9 < v + 1; c9++) {
                                   for (int c10 = 0; c10 < v + 1; c10++) {
                                      for (int c11 = 0; c11 < v + 1; c11++) {
                                         for (int c12 = 0; c12 < v + 1; c12++) {
                                            for (int c13 = 0; c13 < v + 1; c13++) {
                                               for (int c14 = 0; c14 < v + 1; c14++) {
                                                  for (int c15 = 0; c15 < v + 1; c15++) {
                                                     for (int c16 = 0; c16 < v + 1; c16++) {
                                                        for (int c17 = 0; c17 < v + 1; c17++) {
                                                           for (int c18 = 0; c18 < v + 1; c18++) {
                                                              for (int c19 = 0; c19 < v + 1; c19++) {
                                                                 int c20 = v - c1 - c2 - c3 - c4 - c5 - c6 - c7 - c8 - c9 - c10 - c11 - c12 - c13 - c14 - c15 - c16 - c17 - c18 - c19;
                                                                 if (c20 >= 0) {
                                                                    morRez =
(rc[0].multiply(BigInteger.valueOf(c1))).add(rc[1].multiply(BigInteger.valueOf(c2))).add(rc[2].multiply(BigInteger.valueOf(c3))).add(rc[3].multiply(BigInteger.valueOf
(c4))).add(rc[4].multiply(BigInteger.valueOf(c5))).add(rc[5].multiply(BigInteger.valueOf(c6))).add(rc[6].multiply(BigInteger.valueOf(c7))).add(rc[7].multiply(BigInteg
er.valueOf(c8))).add(rc[8].multiply(BigInteger.valueOf(c9))).add(rc[9].multiply(BigInteger.valueOf(c10))).add(rc[10].multiply(BigInteger.valueOf(c11))).add(rc[11].m
ultiply(BigInteger.valueOf(c12))).add(rc[12].multiply(BigInteger.valueOf(c13))).add(rc[13].multiply(BigInteger.valueOf(c14))).add(rc[14].multiply(BigInteger.valueOf
(c15))).add(rc[15].multiply(BigInteger.valueOf(c16))).add(rc[16].multiply(BigInteger.valueOf(c17))).add(rc[17].multiply(BigInteger.valueOf(c18))).add(rc[18].multipl
y(BigInteger.valueOf(c19))).add(rc[19].multiply(BigInteger.valueOf(c20)));
                                                                 if (morRez.equals(finRez)) {
                                                                    if (printCount.isSelected() == false) {
                                                                       scrollText.append("Kandidato gautų balsų skaičius: \n");
                                                                       scrollText.append("1 k.\t2 k.\t3 k.\t4 k.\t5 k.\t6 k.\t7 k.\t8 k.\t9 k.\t10 k.\t11 k.\t12 k.\t13 k.\t14 k.\t15
k.\t16 k.\t17 k.\t18 k.\t19 k.\t20 k.\n");
                                                                       scrollText.append(c1 + "\t" + c2 + "\t" + c3 + "\t" + c4 + "\t" + c5 + "\t" + c6 + "\t" + c7 + "\t" + c8 + "\t" +
c9 + "\t" + c10 + "\t" + c11 + "\t" + c12 + "\t" + c13 + "\t" + c14 + "\t" + c15 + "\t" + c16 + "\t" + c17 + "\t" + c18 + "\t" + c19 + "\t" + c20 + "\n");
                                                                    }
                                                                    break all;
                                                                 }
                                                                 else {
                                                                    long estimatedTime = System.nanoTime() - startTime;
                                                                    double seconds = (double)estimatedTime / 1000000000.0;
                                                                    if (seconds > t)
                                                                       break all;
                                                                 }}}}}}}}}}}}}}}}}}}}}
        long estimatedTime = System.nanoTime() - startTime;
        double seconds = (double)estimatedTime / 1000000000.0;
        scrollText.append("\n Perrinkimo laikas: " + seconds + " s. \n");
     }
}

import java.awt.EventQueue;

public class checking {

   private JFrame checkWindow;
   private JLabel lblKiekKandidat;
   private JCheckBox candAll;
   private JTextField num2;
   private JLabel lblVeiksmaiAtliekamil;
   private JTextField num4;
   private JTextField num5;
   private JTextArea scrollText;
   private JComboBox candNum;
   private JTextField num1;
   JComboBox op2;
   private JLabel lblKandidatoReikm;
   private JComboBox op3;
   private JComboBox op4;
   private JButton button;
    JCheckBox candPrimes;
   JComboBox op1;
   private JTextField num3;
   private JCheckBox chckbxNewCheckBox;
    JTextField timeCheck;

   public static void newWindow() {
      EventQueue.invokeLater(new Runnable() {
         public void run() {
            try {
               checking window = new checking();
               window.checkWindow.setVisible(true);
            } catch (Exception e) {
               e.printStackTrace();
            }
         }
      });
   }

   public checking() {
      initialize();
   }

   private void initialize() {
      checkWindow = new JFrame();
      checkWindow.setTitle("Reikalavim\u0173 paie\u0161ka");
      checkWindow.setBounds(100, 100, 819, 618);
      checkWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

   checkWindow.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      checkWindow.getContentPane().setLayout(null);
      JScrollPane scrollPaneCh = new JScrollPane();
      scrollPaneCh.setBounds(29, 11, 531, 445);
      checkWindow.getContentPane().add(scrollPaneCh);
      scrollText = new JTextArea();
      scrollPaneCh.setViewportView(scrollText);
      String[] candBitNumber = { "10", "15", "20", "25", "30", "40", "50", "75",
"100" };
      button = new JButton("Valyti rezultatus");
      button.addActionListener(new ActionListener() {
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            scrollText.setText("");
        }
    });
    button.setBounds(588, 428, 141, 23);
    checkWindow.getContentPane().add(button);
    lblKiekKandidat = new JLabel("Kiek kandidat\u0173:");
    lblKiekKandidat.setBounds(570, 52, 288, 14);
    checkWindow.getContentPane().add(lblKiekKandidat);
    String[] candNumber = { "3", "4", "5", "6", "7", "8", "9" };
    candNum = new JComboBox(candNumber);
    candNum.setSelectedIndex(0);
    candNum.setBackground(Color.WHITE);
    candNum.setBounds(588, 77, 148, 20);
    checkWindow.getContentPane().add(candNum);
    JLabel label = new JLabel("Kandidatams priskiriami skai\u010Diai:");
    label.setBounds(577, 191, 289, 14);
    checkWindow.getContentPane().add(label);
    candPrimes = new JCheckBox("Pirminiai");
    candPrimes.setSelected(true);
    candPrimes.setBounds(587, 212, 97, 23);
    checkWindow.getContentPane().add(candPrimes);
    candAll = new JCheckBox("Bet kokie");
    candAll.setBounds(587, 238, 97, 23);
    checkWindow.getContentPane().add(candAll);
    ButtonGroup candGroup = new ButtonGroup();
    candGroup.add(candPrimes);
    candGroup.add(candAll);
    String[] oerator = { "+", "-", "*", "/" };
    op1 = new JComboBox(oerator);
    op1.setSelectedIndex(2);
    op1.setBounds(102, 500, 41, 20);
    checkWindow.getContentPane().add(op1);
    op2 = new JComboBox(oerator);
    op2.setSelectedIndex(1);
    op2.setBounds(226, 500, 41, 20);
    checkWindow.getContentPane().add(op2);
    op3 = new JComboBox(oerator);
    op3.setSelectedIndex(1);
    op3.setBounds(350, 500, 41, 20);
    checkWindow.getContentPane().add(op3);
    op4 = new JComboBox(oerator);
    op4.setSelectedIndex(1);
    op4.setBounds(474, 500, 41, 20);
    checkWindow.getContentPane().add(op4);
    num1 = new JTextField("a");
    num1.setColumns(10);
    num1.setBounds(29, 500, 63, 20);
    checkWindow.getContentPane().add(num1);

    num2 = new JTextField("n");
    num2.setColumns(10);
    num2.setBounds(153, 500, 63, 20);
    checkWindow.getContentPane().add(num2);
    num3 = new JTextField("0");
    num3.setColumns(10);
    num3.setBounds(277, 500, 63, 20);
    checkWindow.getContentPane().add(num3);
    num4 = new JTextField("0");
    num4.setColumns(10);
    num4.setBounds(401, 500, 63, 20);
    checkWindow.getContentPane().add(num4);
    num5 = new JTextField("0");
    num5.setColumns(10);
    num5.setBounds(527, 500, 63, 20);
    checkWindow.getContentPane().add(num5);
    lblVeiksmaiAtliekamil = new JLabel("Veiksmai atliekami i\u0161
eil\u0117s, nepriklausomai nuo operatoriaus.");
    lblVeiksmaiAtliekamil.setFont(new Font("Tahoma", Font.PLAIN, 10));
    lblVeiksmaiAtliekamil.setBounds(29, 531, 306, 14);
    checkWindow.getContentPane().add(lblVeiksmaiAtliekamil);
    lblKandidatoReikm = new JLabel("Kandidato reik\u0161m\u0117 - a,
rink\u0117j\u0173 skai\u010Dius - n, k = (n - 1).");
    lblKandidatoReikm.setFont(new Font("Tahoma", Font.PLAIN, 10));
    lblKandidatoReikm.setBounds(29, 543, 306, 14);
    checkWindow.getContentPane().add(lblKandidatoReikm);
    chckbxNewCheckBox = new JCheckBox("Ar spausdinti tarpinius
rezultatus");
    chckbxNewCheckBox.setBounds(575, 289, 229, 23);
    checkWindow.getContentPane().add(chckbxNewCheckBox);
    timeCheck = new JTextField("10800");
    timeCheck.setBounds(588, 142, 148, 20);
    checkWindow.getContentPane().add(timeCheck);
    timeCheck.setColumns(10);
    JLabel lblKiekLaikoTikrinti = new JLabel("Kiek laiko tikrinti
(sekund\u0117mis):");
    lblKiekLaikoTikrinti.setBounds(570, 117, 270, 14);
    checkWindow.getContentPane().add(lblKiekLaikoTikrinti);
    JLabel label_1 = new JLabel("Kandidatams priskiriam\u0173
skai\u010Di\u0173 s\u0105lygos:");
    label_1.setBounds(29, 475, 255, 14);
    checkWindow.getContentPane().add(label_1);

    JButton btnSpausti = new JButton("Tikrinti");
    btnSpausti.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
            try {
                int g = takeCandNumber();
                int t = takeTime();
                String n1, n2, n3, n4, n5;
                n1 = takeNum1(); n2 = takeNum2(); n3 = takeNum3(); n4 = takeNum4(); n5 = takeNum5();
                if (t > 0 && g > 0 && (n1.equals("n") || n1.equals("k") || n1.equals("a") || Integer.parseInt(n1) >= 0) && (n2.equals("n") || n2.equals("k") ||
n2.equals("a") || Integer.parseInt(n2) >= 0) && (n3.equals("n") || n3.equals("a") || n3.equals("k") || Integer.parseInt(n3) >= 0) && (n4.equals("n") || n4.equals("k") ||
n4.equals("a") || Integer.parseInt(n4) >= 0) && (n5.equals("n") || n5.equals("k") || n5.equals("a") || Integer.parseInt(n5) >= 0)) {
                    moreResult();
                }
            }
            catch (Exception exc) {
                JOptionPane.showMessageDialog(checkWindow, "Neteisingai nurodyta(-os) reikšmė(-s).");
                return;
            }
        }
    });
    btnSpausti.setBounds(588, 394, 141, 23);
    checkWindow.getContentPane().add(btnSpausti);

}

public int takeCandNumber() {
    int c = Integer.parseInt(candNum.getSelectedItem().toString());
    return c;
}

public int takeTime() { int c = Integer.parseInt(timeCheck.getText()); return c; }

public String takeNum1() { String c = num1.getText(); return c; }

public String takeNum2() { String c = num2.getText(); return c; }

public String takeNum3() { String c = num3.getText(); return c; }

public String takeNum4() { String c = num4.getText(); return c; }
```

```
public String takeNum5() { String c = num5.getText(); return c; }

public String takeOperator1() { String c = p1.getSelectedItem().toString();     return c; }

public String takeOperator2() { String c = op2.getSelectedItem().toString(); return c; }

public String takeOperator3() { String c = op3.getSelectedItem().toString(); return c; }

public String takeOperator4() { String c = op4.getSelectedItem().toString(); return c; }
```

```
public void moreResult () {
    BigInteger[] rc; BigInteger rc1, rc2 = BigInteger.valueOf(1);
    int v;
    int t = takeTime();
    int[] vv = {5, 10, 50, 100, 200, 400, 800};
    long startTime = System.nanoTime();
    for (int l = 0; l < vv.length; l++) {
       double seconds = 0;
       int j = 0;
       v = vv[l];
       scrollText.append("\n\n    " + v + " rinkėjai(-ų) \n");
       while (seconds <= t) {
          if (j == 0) {
             rc1 = BigInteger.valueOf(v);
             rc = candidateValues(rc1, v);
          }
          else {
             rc1 = rc2.add(BigInteger.valueOf(1));
             rc = candidateValues(rc1, v);
          }
          rc2 = rc[0];
          int g = takeCandNumber();
          if (chckbxNewCheckBox.isSelected() == true) {
             scrollText.append("\n     " + (j + 1) + " variantas \n\n");
             for (int i = 0; i < rc.length; i++) {
                scrollText.append("   " + BigInteger.valueOf(i + 1) + " kand.: " +
rc[i] + "\n");
             }
          }
          if (g == 3)
             if (thenCand3(rc, v) == false)
                return;
          if (g == 4)
             if (thenCand4(rc, v) == false)
                return;
          if (g == 5)
             if (thenCand5(rc, v) == false)
                return;
          if (g == 6)
             if (thenCand6(rc, v) == false)
                return;
          if (g == 7)
             if (thenCand7(rc, v) == false)
                return;
          if (g == 8)
             if (thenCand8(rc, v) == false)
                return;
          if (g == 9)
             if (thenCand9(rc, v) == false)
                return;
          j = j + 1;
          long estimatedTime = System.nanoTime() - startTime;
          seconds = (double)estimatedTime / 1000000000.0;
       }
       scrollText.append("\n     Kai " + v + " rinkėjų, patikrinta " + j + "
variantas(-ų) \n\n");
    }
 }
```

```
public BigInteger[] candidateValues(BigInteger rc1, int v) {
    String n1, n2, n3, n4, n5, o1, o2, o3, o4;
    n1 = takeNum1(); n2 = takeNum2(); n3 = takeNum3(); n4 = takeNum4();
n5 = takeNum5(); o1 = takeOperator1(); o2 = takeOperator2(); o3 =
takeOperator3(); o4 = takeOperator4();
    BigInteger var1 = new BigInteger("0"), var2 = new BigInteger("0"), var3 =
new BigInteger("0"), var4 = new BigInteger("0"), var5 = new BigInteger("0"),
var = new BigInteger("0"), vari = new BigInteger("0"), vari2 = new
BigInteger("0");
    boolean in1 = false, in2 = false, in3 = false, in4 = false, in5 = false;
    boolean isP = false;
    int c = takeCandNumber();
    BigInteger sv = new BigInteger("1");
    BigInteger[] rc = new BigInteger[c];
    rc[0] = rc1;
    if (candPrimes.isSelected() == true) {
       while (isP != true) {
          if (rc[0].isProbablePrime(100) == true)
             isP = true;
          else
             rc[0] = rc[0].add(sv);
       }
    }
    if (n1.equals("n")) var1 = BigInteger.valueOf(v);
    else if (n1.equals("a")) { var1 = rc[0]; in1 = true; }
       else if (n1.equals("k")) var1 = BigInteger.valueOf(v-1);
          else var1 = new BigInteger(n1);
    if (n2.equals("n")) var2 = BigInteger.valueOf(v);
    else if (n2.equals("a")) { var2 = rc[0]; in2 = true; }
       else if (n2.equals("k")) var2 = BigInteger.valueOf(v-1);
          else var2 = new BigInteger(n2);
    if (n3.equals("n")) var3 = BigInteger.valueOf(v);
    else if (n3.equals("a")) { var3 = rc[0]; in3 = true; }
       else if (n3.equals("k")) var3 = BigInteger.valueOf(v-1);
          else var3 = new BigInteger(n3);
    if (n4.equals("n")) var4 = BigInteger.valueOf(v);
    else if (n4.equals("a")) { var4 = rc[0]; in4 = true; }
       else if (n4.equals("k")) var4 = BigInteger.valueOf(v-1);
          else var4 = new BigInteger(n4);
    if (n5.equals("n")) var5 = BigInteger.valueOf(v);
    else if (n5.equals("a")) { var5 = rc[0]; }
       else if (n5.equals("k")) var5 = BigInteger.valueOf(v-1);
          else var5 = new BigInteger(n5);
    // jeigu pirminiai
    if (candPrimes.isSelected() == true) {
       for (int i = 1; i < c; i++) {
          isP = false;
          if (in1 == false && in2 == false) {
             if (o1.equals("*")) var = var1.multiply(var2);
                else if (o1.equals("+")) var = var1.add(var2);
                   else if (o1.equals("-")) var = var1.subtract(var2);
                      else var = var1.divide(var2);
          }
          if (in1 == true && in2 == false) {
             if (o1.equals("*")) var = rc[i-1].multiply(var2);
             else if (o1.equals("+")) var = rc[i-1].add(var2);
                else if (o1.equals("-")) var = rc[i-1].subtract(var2);
                   else var = rc[i-1].divide(var2);
          }
          if (in1 == false && in2 == true) {
             if (o1.equals("*")) var = var1.multiply(rc[i-1]);
             else if (o1.equals("+")) var = var1.add(rc[i-1]);
                else if (o1.equals("-")) var = var1.subtract(rc[i-1]);
                   else var = var1.divide(rc[i-1]);
          }
          if (in3 == true) {
             if (o2.equals("*")) vari = var.multiply(rc[i-1]);
             else if (o2.equals("+")) vari = var.add(rc[i-1]);
                else if (o2.equals("-")) vari = var.subtract(rc[i-1]);
                   else vari = var.divide(rc[i-1]);
          }
          if (in3 == false) {
             if (o2.equals("*")) vari = var.multiply(var3);
             else if (o2.equals("+")) vari = var.add(var3);
                else if (o2.equals("-")) vari = var.subtract(var3);
                   else vari = var.divide(var3);
          }
          if (in4 == true) {
             if (o3.equals("*")) vari2 = vari.multiply(rc[i-1]);
             else if (o3.equals("+")) vari2 = vari.add(rc[i-1]);
                else if (o3.equals("-")) vari2 = vari.subtract(rc[i-1]);
                   else vari2 = vari.divide(rc[i-1]);
          }
          if (in4 == false) {
             if (o3.equals("*")) vari2 = vari.multiply(var4);
             else if (o3.equals("+")) vari2 = vari.add(var4);
                else if (o3.equals("-")) vari2 = vari.subtract(var4);
                   else vari2 = vari.divide(var4);
          }
          if (in5 == false) {
             if (o4.equals("*")) rc[i] = vari2.multiply(var5);
             else if (o4.equals("+")) rc[i] = vari2.add(var5);
                else if (o4.equals("-")) rc[i] = vari2.subtract(var5);
                   else rc[i] = vari2.divide(var5);
          }
          while (isP != true) {
             if (rc[i].isProbablePrime(100) == true)
                isP = true;
```

```
            else
               rc[i] = rc[i].add(sv);
         }
      }
   }
   // jeigu bet kokie
   else {
      for (int i = 1; i < c; i++) {
         if (in1 == false && in2 == false) {
            if (o1.equals("*")) var = var1.multiply(var2);
            else if (o1.equals("+")) var = var1.add(var2);
               else if (o1.equals("-")) var = var1.subtract(var2);
                  else var = var1.divide(var2);
         }
         if (in1 == true && in2 == false) {
            if (o1.equals("*")) var = rc[i-1].multiply(var2);
            else if (o1.equals("+")) var = rc[i-1].add(var2);
               else if (o1.equals("-")) var = rc[i-1].subtract(var2);
                  else var = rc[i-1].divide(var2);
         }
         if (in1 == false && in2 == true) {
            if (o1.equals("*")) var = var1.multiply(rc[i-1]);
            else if (o1.equals("+")) var = var1.add(rc[i-1]);
               else if (o1.equals("-")) var = var1.subtract(rc[i-1]);
                  else var = var1.divide(rc[i-1]);
         }
         if (in3 == true) {
            if (o2.equals("*")) vari = var.multiply(rc[i-1]);
            else if (o2.equals("+")) vari = var.add(rc[i-1]);
               else if (o2.equals("-")) vari = var.subtract(rc[i-1]);
                  else vari = var.divide(rc[i-1]);
         }
         if (in3 == false) {
            if (o2.equals("*")) vari = var.multiply(var3);
            else if (o2.equals("+")) vari = var.add(var3);
               else if (o2.equals("-")) vari = var.subtract(var3);
                  else vari = var.divide(var3);
         }
         if (in4 == true) {
            if (o3.equals("*")) vari2 = vari.multiply(rc[i-1]);
            else if (o3.equals("+")) vari2 = vari.add(rc[i-1]);
               else if (o3.equals("-")) vari2 = vari.subtract(rc[i-1]);
                  else vari2 = vari.divide(rc[i-1]);
         }
         if (in4 == false) {
            if (o3.equals("*")) vari2 = vari.multiply(var4);
            else if (o3.equals("+")) vari2 = vari.add(var4);
               else if (o3.equals("-")) vari2 = vari.subtract(var4);
                  else vari2 = vari.divide(var4);
         }
         if (in5 == false) {
            if (o4.equals("*")) rc[i] = vari2.multiply(var5);
            else if (o4.equals("+")) rc[i] = vari2.add(var5);
               else if (o4.equals("-")) rc[i] = vari2.subtract(var5);
                  else rc[i] = vari2.divide(var5);
         }
      }
   }
   return rc;
}

public boolean checkSum(BigInteger[] sum, String[] index, int j) {
   int q = 0, qq = 0;
   String[] repeatSum = new String[640000];
   for (int p = 0; p <= j; p++) {
      for (int l = 0; l <= j; l++) {
         if (p != l) {
            if (sum[p].equals(sum[l])) {
               repeatSum[qq] = index[p] + "\t" + sum[p];
               repeatSum[qq+1] = index[l] + "\t" + sum[l];
               if (chckbxNewCheckBox.isSelected() == true) {
                  scrollText.append(" \n");
                  //scrollText.append("\nYra neunikalių sumų: \n");
                  for (int u = 0; u < qq; u++) {
                     scrollText.append(repeatSum[u] + "\n");
                  }
               }
               q = q + 1;
               qq = qq + 2;
            }
         }
      }
   }
   if (q > 0)
      return true;
   else
      return false;
}

public boolean thenCand3 (BigInteger[] rc, int v) {
   BigInteger[] sum = new BigInteger[640000];
   String[] index = new String[640000];
   int j = 0;
   if (chckbxNewCheckBox.isSelected() == true)
      scrollText.append("   Už 1 kand.\t Už 2 kand.\t Už 3 kand.\t Suma \n");
   int t = takeTime();
   long startTime = System.nanoTime();
   for (int i1 = 0; i1 <= v; i1++) {
      for (int i2 = 0; i2 <= v; i2++) {
         int i3 = v - i1 - i2;
         if (i3 >= 0) {
            sum[j] = ((rc[0].multiply(BigInteger.valueOf(i1))).add(rc[1].multiply(BigInteger.valueOf(i2)))).add(rc[2].multiply(BigInteger.valueOf(i3)));

            if (chckbxNewCheckBox.isSelected() == true) {
               scrollText.append("   " + i1 + "\t" + i2 + "\t" + i3 + "\t" + sum[j] + "\n");
               index[j] = i1 + "\t" + i2 + "\t" + i3;
            }
            if (checkSum(sum, index, j) == true) {
               scrollText.append("\n   Yra neunikalių sumų!\n");
               return false;
            }
            j = j + 1;
            long estimatedTime = System.nanoTime() - startTime;
            double seconds = (double)estimatedTime / 1000000000.0;
            if (seconds > t) {
               scrollText.append("   Nepatikrinti visi galimi variantai, kai yra 3 kandidatai. Patikrinta " + j + " var. \n");
               return true;
            } } } }
   if (chckbxNewCheckBox.isSelected() == true)
      scrollText.append("\n   Visos sumos unikalios. \n");
   return true;
}

public boolean thenCand4 (BigInteger[] rc, int v) {
   BigInteger[] sum = new BigInteger[640000];
   String[] index = new String[640000];
   int j = 0;
   if (chckbxNewCheckBox.isSelected() == true)
      scrollText.append("    Už 1 kand.\t Už 2 kand.\t Už 3 kand.\t Už 4 kand.\t Suma \n");
   int t = takeTime();
   long startTime = System.nanoTime();
   for (int i1 = 0; i1 <= v; i1++) {
```

```
            for (int i2 = 0; i2 <= v; i2++) {
               for (int i3 = 0; i3 <= v; i3++) {
                  int i4 = v - i1 - i2 - i3;
                  if (i4 >= 0) {
                     sum[j] =
((rc[0].multiply(BigInteger.valueOf(i1))).add(rc[1].multiply(BigInteger.valueOf(i2)))).add(rc[2].multiply(BigInteger.valueOf(i3))).add(rc[3].multiply(BigInteger.valueOf
(i4)));
                     if (chckbxNewCheckBox.isSelected() == true) {
                        scrollText.append("   " +i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + sum[j] + "\n");
                        index[j] = i1 + "\t" + i2 + "\t" + i3 + "\t" + i4;
                     }
                     if (checkSum(sum, index, j) == true) {
                        scrollText.append("\n   Yra neunikalių sumų. \n");
                        return false;
                     }
                     j = j + 1;
                     long estimatedTime = System.nanoTime() - startTime;
                     double seconds = (double)estimatedTime / 1000000000.0;
                     if (seconds > t) {
                        scrollText.append("   Nepatikrinti visi galimi variantai, kai yra 4 kandidatai. Patikrinta " + j + " var. \n");
                        return true;
                     } } } } }
      if (chckbxNewCheckBox.isSelected() == true)
         scrollText.append("\n   Visos sumos unikalios. \n");
      return true;
   }

   public boolean thenCand5 (BigInteger[] rc, int v) {
      BigInteger[] sum = new BigInteger[640000];
      String[] index = new String[640000];
      int j = 0;
      if (chckbxNewCheckBox.isSelected() == true)
         scrollText.append("    Už 1 kand.\t Už 2 kand.\t Už 3 kand.\t Už 4 kand.\t Už 5 kand.\t Suma \n");
      int t = takeTime();
      long startTime = System.nanoTime();
      for (int i1 = 0; i1 <= v; i1++) {
         for (int i2 = 0; i2 <= v; i2++) {
            for (int i3 = 0; i3 <= v; i3++) {
               for (int i4 = 0; i4 <= v; i4++) {
                  int i5 = v - i1 - i2 - i3 - i4;
                  if (i5 >= 0) {
                     sum[j] =
((rc[0].multiply(BigInteger.valueOf(i1))).add(rc[1].multiply(BigInteger.valueOf(i2)))).add(rc[2].multiply(BigInteger.valueOf(i3))).add(rc[3].multiply(BigInteger.valueOf
(i4))).add(rc[4].multiply(BigInteger.valueOf(i5)));
                     if (chckbxNewCheckBox.isSelected() == true) {
                        scrollText.append("   " + i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5 + "\t" + sum[j] + "\n");
                        index[j] = i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5;
                     }
                     if (checkSum(sum, index, j) == true) {
                        scrollText.append("   Yra neunikalių sumų. \n");
                        return false;
                     }
                     j = j + 1;
                     long estimatedTime = System.nanoTime() - startTime;
                     double seconds = (double)estimatedTime / 1000000000.0;
                     if (seconds > t) {
                        scrollText.append("   Nepatikrinti visi galimi variantai, kai yra 5 kandidatai. Patikrinta " + j + " var. \n");
                        return true;
                     }  } } } } }
      if (chckbxNewCheckBox.isSelected() == true)
         scrollText.append("\n   Visos sumos unikalios. \n");
      return true;
   }

   public boolean thenCand6 (BigInteger[] rc, int v) {
      BigInteger[] sum = new BigInteger[640000];
      String[] index = new String[640000];
      int j = 0;
      if (chckbxNewCheckBox.isSelected() == true)
         scrollText.append("    Už 1 kand.\t Už 2 kand.\t Už 3 kand.\t Už 4 kand.\t Už 5 kand.\t Už 6 kand.\t Suma \n");
      int t = takeTime();
      long startTime = System.nanoTime();
      for (int i1 = 0; i1 <= v; i1++) {
         for (int i2 = 0; i2 <= v; i2++) {
            for (int i3 = 0; i3 <= v; i3++) {
               for (int i4 = 0; i4 <= v; i4++) {
                  for (int i5 = 0; i5 <= v; i5++) {
                     int i6 = v - i1 - i2 - i3 - i4 - i5;
                     if (i6 >= 0) {
                        sum[j] =
((rc[0].multiply(BigInteger.valueOf(i1))).add(rc[1].multiply(BigInteger.valueOf(i2)))).add(rc[2].multiply(BigInteger.valueOf(i3))).add(rc[3].multiply(BigInteger.valueOf
(i4))).add(rc[4].multiply(BigInteger.valueOf(i5))).add(rc[5].multiply(BigInteger.valueOf(i6)));
                        if (chckbxNewCheckBox.isSelected() == true) {
                           scrollText.append("   " + i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5 + "\t" + i6 + "\t" + sum[j] + "\n");
                           index[j] = i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5 + "\t" + i6;
                        }
                        if (checkSum(sum, index, j) == true) {
                           scrollText.append("\n   Yra neunikalių sumų. \n");
                           return false;
```

```
                        }
                        j = j + 1;
                        long estimatedTime = System.nanoTime() - startTime;
                        double seconds = (double)estimatedTime / 1000000000.0;
                        if (seconds > t) {
                            scrollText.append("   Nepatikrinti visi galimi variantai, kai yra 6 kandidatai. Patikrinta " + j + " var. \n");
                            return true;
                        } } } } } }
        if (chckbxNewCheckBox.isSelected() == true)
            scrollText.append("\n   Visos sumos unikalios. \n");
        return true;
    }

    public boolean thenCand7 (BigInteger[] rc, int v) {
        BigInteger[] sum = new BigInteger[640000];
        String[] index = new String[640000];
        int j = 0;
        if (chckbxNewCheckBox.isSelected() == true)
            scrollText.append("    Už 1 kand.\t Už 2 kand.\t Už 3 kand.\t Už 4 kand.\t Už 5 kand.\t Už 6 kand.\t Už 7 kand.\t Suma \n");
        int t = takeTime();
        long startTime = System.nanoTime();
        for (int i1 = 0; i1 <= v; i1++) {
            for (int i2 = 0; i2 <= v; i2++) {
                for (int i3 = 0; i3 <= v; i3++) {
                    for (int i4 = 0; i4 <= v; i4++) {
                        for (int i5 = 0; i5 <= v; i5++) {
                            for (int i6 = 0; i6 <= v; i6++) {
                                int i7 = v - i1 - i2 - i3 - i4 - i5 - i6;
                                if (i7 >= 0) {
                                    sum[j] =
((rc[0].multiply(BigInteger.valueOf(i1))).add(rc[1].multiply(BigInteger.valueOf(i2)))).add(rc[2].multiply(BigInteger.valueOf(i3))).add(rc[3].multiply(BigInteger.valueOf
(i4))).add(rc[4].multiply(BigInteger.valueOf(i5))).add(rc[5].multiply(BigInteger.valueOf(i6))).add(rc[6].multiply(BigInteger.valueOf(i7)));

                                    if (chckbxNewCheckBox.isSelected() == true) {
                                        scrollText.append("   " +i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5 + "\t" + i6 + "\t" + i7 + "\t" + sum[j] + "\n");
                                        index[j] = i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5 + "\t" + i6 + "\t" + i7;
                                    }
                                    if (checkSum(sum, index, j) == true) {
                                        scrollText.append("\n   Yra neunikalių sumų. \n");
                                        return false;
                                    }
                                    j = j + 1;
                                    long estimatedTime = System.nanoTime() - startTime;
                                    double seconds = (double)estimatedTime / 1000000000.0;
                                    if (seconds > t) {
                                        scrollText.append("   Nepatikrinti visi galimi variantai, kai yra 7 kandidatai. Patikrinta " + j + " var. \n");
                                        return true;
                                    } } } } } } } }
        if (chckbxNewCheckBox.isSelected() == true)
            scrollText.append("\n   Visos sumos unikalios. \n");
        return true;
    }

    public boolean thenCand8 (BigInteger[] rc, int v) {
        BigInteger[] sum = new BigInteger[640000];
        String[] index = new String[640000];
        int j = 0;
        if (chckbxNewCheckBox.isSelected() == true)
            scrollText.append("    Už 1 kand.\t Už 2 kand.\t Už 3 kand.\t Už 4 kand.\t Už 5 kand.\t Už 6 kand.\t Už 7 kand.\t Už 8 kand.\t Suma \n");
        int t = takeTime();
        long startTime = System.nanoTime();
        for (int i1 = 0; i1 <= v; i1++) {
            for (int i2 = 0; i2 <= v; i2++) {
                for (int i3 = 0; i3 <= v; i3++) {
                    for (int i4 = 0; i4 <= v; i4++) {
                        for (int i5 = 0; i5 <= v; i5++) {
                            for (int i6 = 0; i6 <= v; i6++) {
                                for (int i7 = 0; i7 <= v; i7++) {
                                    int i8 = v - i1 - i2 - i3 - i4 - i5 - i6 - i7;
                                    if (i8 >= 0) {
                                        sum[j] =
((rc[0].multiply(BigInteger.valueOf(i1))).add(rc[1].multiply(BigInteger.valueOf(i2)))).add(rc[2].multiply(BigInteger.valueOf(i3))).add(rc[3].multiply(BigInteger.valueOf
(i4))).add(rc[4].multiply(BigInteger.valueOf(i5))).add(rc[5].multiply(BigInteger.valueOf(i6))).add(rc[6].multiply(BigInteger.valueOf(i7))).add(rc[7].multiply(BigInteger.
valueOf(i8)));
                                        if (chckbxNewCheckBox.isSelected() == true) {
                                            scrollText.append("   " + i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5 + "\t" + i6 + "\t" + i7 + "\t" + i8 + "\t" + sum[j] + "\n");
                                            index[j] = i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5 + "\t" + i6 + "\t" + i7 + "\t" + i8;
                                        }
                                        if (checkSum(sum, index, j) == true) {
                                            scrollText.append("\n   Yra neunikalių sumų. \n");
                                            return false;
                                        }
                                        j = j + 1;
                                        long estimatedTime = System.nanoTime() - startTime;
                                        double seconds = (double)estimatedTime / 1000000000.0;
                                        if (seconds > t) {
                                            scrollText.append("   Nepatikrinti visi galimi variantai, kai yra 8 kandidatai. Patikrinta " + j + " var. \n");
                                            return true;
                                        } } } } } } } } }
```

```
        if (chckbxNewCheckBox.isSelected() == true)
            scrollText.append("\n   Visos sumos unikalios. \n");
        return true;
    }


    public boolean thenCand9 (BigInteger[] rc, int v) {
        BigInteger[] sum = new BigInteger[640000];
        String[] index = new String[640000];
        int j = 0;
        if (chckbxNewCheckBox.isSelected() == true)
            scrollText.append("    Už 1 kand.\t Už 2 kand.\t Už 3 kand.\t Už 4 kand.\t Už 5 kand.\t Už 6 kand.\t Už 7 kand.\t Už 8 kand.\t Už 9 kand.\t Suma \n");
        int t = takeTime();
        long startTime = System.nanoTime();
        for (int i1 = 0; i1 <= v; i1++) {
            for (int i2 = 0; i2 <= v; i2++) {
                for (int i3 = 0; i3 <= v; i3++) {
                    for (int i4 = 0; i4 <= v; i4++) {
                        for (int i5 = 0; i5 <= v; i5++) {
                            for (int i6 = 0; i6 <= v; i6++) {
                                for (int i7 = 0; i7 <= v; i7++) {
                                    for (int i8 = 0; i8 <= v; i8++) {
                                        int i9 = v - i1 - i2 - i3 - i4 - i5 - i6 - i7 - i8;
                                        if (i9 >= 0) {
                                            sum[j] =
((rc[0].multiply(BigInteger.valueOf(i1))).add(rc[1].multiply(BigInteger.valueOf(i2)))).add(rc[2].multiply(BigInteger.valueOf(i3))).add(rc[3].multiply(BigInteger.valueOf
(i4))).add(rc[4].multiply(BigInteger.valueOf(i5))).add(rc[5].multiply(BigInteger.valueOf(i6))).add(rc[6].multiply(BigInteger.valueOf(i7))).add(rc[7].multiply(BigInteger.
valueOf(i8))).add(rc[8].multiply(BigInteger.valueOf(i9)));
                                            if (chckbxNewCheckBox.isSelected() == true) {
                                                scrollText.append("   " + i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5 + "\t" + i6 + "\t" + i7 + "\t" + i8 + "\t" + i9 + "\t" + sum[j] + "\n");
                                                index[j] = i1 + "\t" + i2 + "\t" + i3 + "\t" + i4 + "\t" + i5 + "\t" + i6 + "\t" + i7 + "\t" + i8 + "\t" + i9;
                                            }
                                                if (checkSum(sum, index, j) == true) {
                                                scrollText.append("\n   Yra neunikalių sumų. \n");
                                                return false;
                                            }
                                            j = j + 1;
                                            long estimatedTime = System.nanoTime() - startTime;
                                            double seconds = (double)estimatedTime / 1000000000.0;
                                            if (seconds > t) {
                                                scrollText.append("   Patikrinta " + j + " var. \n");
                                                return true;
                                            } } } } } } } } }
        if (chckbxNewCheckBox.isSelected() == true)
            scrollText.append("\n   Visos sumos unikalios. \n");
        return true;
    }
}
```