KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Petras Dičpinigaitis

Delninukų energijos suvartojimo apdorojant išretintas matricas saugomas stulpeliais modeliavimas

Magistro darbas

Recenzentas
doc. dr. S. Maciulevičius
2008-01-14

Vadovas
doc. dr. E. Toldinas
2008-01-14

Atliko
IFM-2/4 gr. stud.
Petras Dičpinigaitis
2008-01-14

Kaunas, 2008

1. Įvadas

Šiuo metu viena iš svarbesnių problemų yra nešiojamų įrenginių baterijos gyvavimo laikas, kadangi vis dar nėra išrasta ilgaamžė baterija, kuri nepkrauta laikytų ilgą laiko tarpą. Šiuo metu tobulėja tik patys įrenginiai. Todėl baterijos gyvavimo trukmės problema egzistuoja. Kol nėra ilgaamžės baterijos, energijos taupymą stengiamasi kompensuoti kitais būdais. Dažniausiai tai būna programiniai spendimai arba patarimai, kaip naudoti nešiojamą įrenginį, kad jis tarnautų kuo ilgiau. Tai būtų nenaudojamų parametrų išjungimas ir t.t. Literatūros šaltiniuose aprašomi teoriniai modeliai, kaip būtų galima optimizuoti nešiojamo įrenginio darbą. Vieni jų realizuoti, kiti ir lieka teoriniai. Keletą metodų išanalizuosime analizės dalyje.

Suprasdami šią problemą mes nusprendėme atlikti mokslinį tyrimą, tam, kad galėtumėme ištirti nešiojamo įrenginio – delninuko baterijos energijos suvartojimą. Iš visų energiją vartojančių delninuko elementų išsirinkome du – atmintį ir procesorių, kadangi norėjome pasižiūrėti, kokia yra šių dviejų komponentų įtaką bendram energijos suvartojimui. Toki pasirinkimą nulėmė tai, kad nėra daug tokių tyrimų atliktų šioje srityje.

Norėdami atlikti tokį tyrimą reikėjo pasirinkti metodą, kuris galėtų apkrauti atmintį ir procesorių. Todėl nusprendėme panaudoti matricų daugybą, kuri puikiai tinka apkrauti minėtus elementus. Panaudosime vieną iš išretintų matricų metodų - stulpelių saugojimo metodą. Išretintos matricos yra plačiai naudojamos realiose programose, todėl ir pasirinkome šį metodą.

Darbo metu naudodami Visual Studio .NET 2003 sukursime testavimo progamą, kuri atliks dviejų metodų daygybą: paprastą ir stulpelių saugojimo metodo. Dauginimo metu bus stebimą eilė baterijos parametrų ir iš gautų rezultatų bus daromos atitinkamos išvados apie metodo efektyvumą.

2. Analizės dalis

2.1. Analizės tikslas

Analizės tikslas – išanalizuoti mokslinius straipsnius apie energijos suvartojimą mobiliuose kompiuteriuose. Išanalizuoti sukurtas programas, kurios atlikdamos skaičiavimus stebi procesoriaus ir atminties suvartojimą. Ištirti metodus, bei sprendimus, kurie buvo pasiūlyti, tam, kad sumažintume bendrą energijos suvartojimą ir kurie leistų ilgiau naudoti įrenginį su išorine baterija.

2.2. Tyrimo sritis, objektas ir problema

- Tyrimo sritis: Informacinės sistemos.
- Tyrimo objektas: mobilūs įrenginiai, naudojami informacinėse sistemose.
- Problema: kiekvienas mobilus įrenginys turi bateriją, o tai reiškia, kad jų darbo laikas yra ribotas. Kaip pasiekti, kuo ilgesnį mobiliojo įrenginio darbo laiką, be papildomo pakrovimo.

2.3 Mobiliųjų informacinių sistemų vartotojai

Kiekvienais metais, tobulėja mobilios informacinės technologijos. Sukuriama įvairiausių naujų mobilių įrenginių, kurie vartotoją padaro mobilų. Paprastus telefonus keičia sumanūs, galingi ir daug funkcijų turintys protingi telefonai, kitaip dar vadinami SmartFone. Galime tiktai paspėlioti, kokie įrenginiai bus po kelerių metų, tačiau mūsų manymu, protingus telefonus (SmartFone) pakeis kišeniniai kompiuteriai (PocketPC). Šioje srityje būtų lyg ir viskas gerai , jei ne baterijos gyvavimo laikas. Kol kas tobulėja tik patys įrenginiai, o ne baterijos. Kuo įrenginys yra labiau sudėtingesnis, tuo daugiau jis suvartoja baterijos energijos.

2.4 Mobilių informacinių sistemų priklausomybė nuo energijos šaltinio analizė

Nors ilgaamžės baterijos dar nesukurta, tačiau tyrimai atliekami visame pasaulyje. Tyrimu metu bandoma išsiaiškinti, kas ir kiek įtakoja daugiausiai energijos suvartojimą,

todėl atliekami bandymai yra skirtingi, kadangi atrandama vis daugiau naujų metodų ir būdų, kaip ištirti baterijos energijos sunaudojimą. Toliau aptarsime keletą straipsnių šia tema.

2.4.1 Energijos optimizavimas nešiojamuose daugialypės terpės sistemose

Rajendra Turakani ir Krishna Kumar ištyrė nešiojamų muzikos grotuvų baterijos ypatybes.[1]

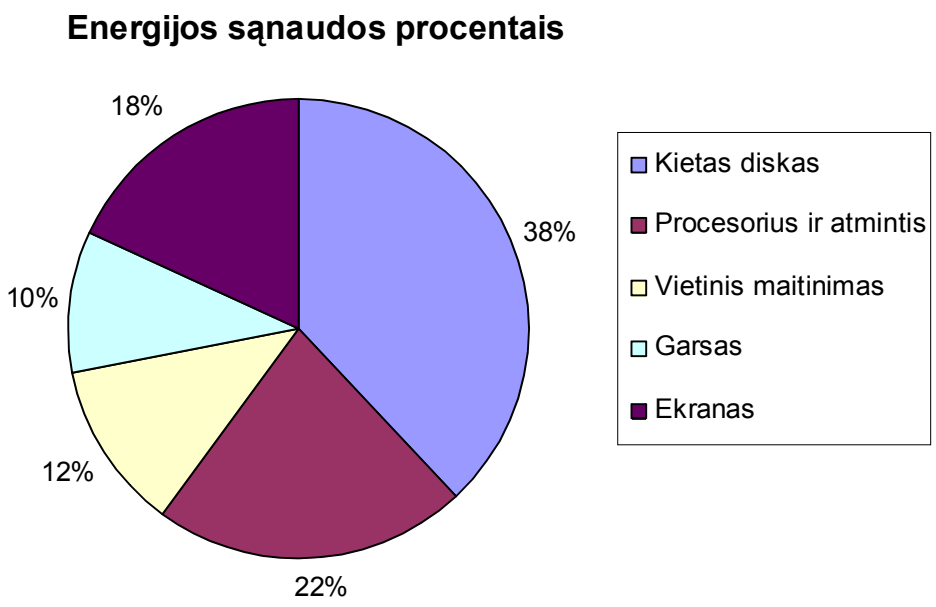
Pirmas žingsnis energijos optimizavime yra energijos apibūdinimas, kuris padeda projektuotojam vykdyti efektyvias strategijas metodiniu būdu. Pavyzdžiui, kietas diskas naudoja mechaninį variklį. Taigi, jis sunaudoja daug energijos. Neseniai pagaminti kieti diskai, būna su energijos optimizavimo galimybėmis, įtraukiant įvairius darbo rėžimus, kuriuose diskas gali dirbti sunaudodamas tik tam rėžimui būdingą energijos kiekį.

1 lentelė. Kieto disko, skirtingų darbo rėžimų, suvartojama energija

Rėžimas	Energijos suvartojimas 2.5“ kieto disko (mA)
Užkrovimas (start up)	450
Aktyvus Idle	150
Ieškoti	340
Skaityti/rašyti	350
Pristabdyti	30
Miegoti	20

2.1 lentelė parodo kieto disko skirtingus darbo rėžimus ir sunaudojamą energijos kiekį. Kaip matome daugiausia energijos suvartojama, kai kietas diskas įsijungia. Mažiausia, kai diskas yra miegojimo rėžime, tačiau tada negalime atlikti jokių operacijų, kadangi diskas nesisuka.

Sekanti diagrama parodo, koki pagrindiniai komponentai įeina į energijos sąnaudas. Pavaizduotos diagramos duomenys sudaryti nešiojamui muzikos grotuvui, kuris turi bazinį ARM tipo procesorių su Linux operacine sistema.



1 Pav. Skirtingų kompiuterio komponentų energijos sąnaudos diagrama

Plačiau panagrinėsime procesoriaus ir atminties bloką. Procesoriaus ir atminties blokas yra antras (22 %) dydis, pagal suvartojamą bendrą energijos kiekį grotuve. Skaitmeninėje sistemoje aukštesnis operuojamas dažnis matomai padidina perjungimo praradimą ir tai įtakoja didesnį energijos suvartojimo kiekį. Pavyzdžiui $E = 0.5CV^2F$, kur E apibūdina perjungimo (switching) maitinimo praradimą, C – efektyvi talpa, V – įtampa, F – dažnis.

Sistema turėtų būti pakankamai protinga, kad galėtų pakeisti sinchronizatoriaus nustatymus priklausomai nuo vartotojo pasirinktos kodavimo priemonės. Parenkant optimalų sinchronizatoriaus dažnį, leistų sutaupyti beveik 5% visos energijos.

SDRAM tipo atminties pasirinkimas sudaro pagrindinę rolę energijos diagramoje. Mobilūs SDRAM yra geriau tinkantys sistemose, kur naudojama baterija. Jų naudojimas sistemoje, energijos suvartojimą sumažina apie 60% lyginant su įprastine SDRAM atmintimi. SDRAM kontroleriai leidžia projektuotojams paruošti reikiamą kaupiklio patvarumą. Taigi, tai turėtų būti nustatyta į minimumą, kol konfigūruojami kontroleriai.

Modernūs daugialypės terpės procesoriai dažniausiai būna su keletu periferijų, tokie kaip UART, USB, Video iškodavimas ir t.t. Projektuotojai turi išjungti nenaudojamas periferijas. Išjungiant sinchronizatorių atitinkamiems moduliam, leistų sutaupyti apie 5 % bendros suvartojimo energijos.

Kai kurie procesoriai leidžia nustatyti kokią įtampą galima paduoti į branduolį. Pavyzdžiui, aprašyme būtų parašyta, kad branduolio įtampa gali būti nuo 1.2 ir 1.3 V. Dažniausiai yra patariama palaikyti mažiausią galimą įtampą. Tai leidžia sumažinti perjungimo energijos praradimą ir tuo pačiu sumažinti bendrą energijos kiekį, nepakeičiant visos sistemos efektyvumo.

2.4.2 Technologija qSleep (mikro–sleep)

Lawrence S. Brakmo, Deborah A. Wallach, Marc A. Viredaz [2] pasiūlė technologiją, kuri vadinasi qSleep (mikro–sleep), kuri sumažina energijos suvartojimą nešiojamuose įrenginiuose.

Pagrindinė metodo esmė yra tokia: vietoj to, kad visada procesorius būna savo laukimo būsenos (idle) rėžime trumpais laiko momentais (mažiau nei sekundė), procesorius perjungiamas į jo miegojimo rėžimą, kada tik tai įmanoma padaryti. Miegojimo būsena, kuri naudojama skiriasi nuo paprastos sistemos miegojimo būsenos. Pirmas skirtumas susijęs su įjungtu monitoriumi. Laikyti monitorių įjungtą, kol rodomas paveikslėlis, kuris buvo atidarytas prieš perjungiant sistemą į miegojimo rėžimą. Antras skirtumas susijęs su sistemos pažadinimu prieš kitą operacinės sistemos suplanuotą įvykį, toks kaip branduolio laiko aptarnavimas. Paskutinis skirtumas susijęs su sistemos pažadinimu, kai pasirodo išorinis įvykis, toks kaip mygtuko paspaudimas ar liečiamo ekrano palietimas. Pagrindinis visų skirtumų tikslas – padaryti taip, kad vartotojas neįtartų, jog procesorius miega.

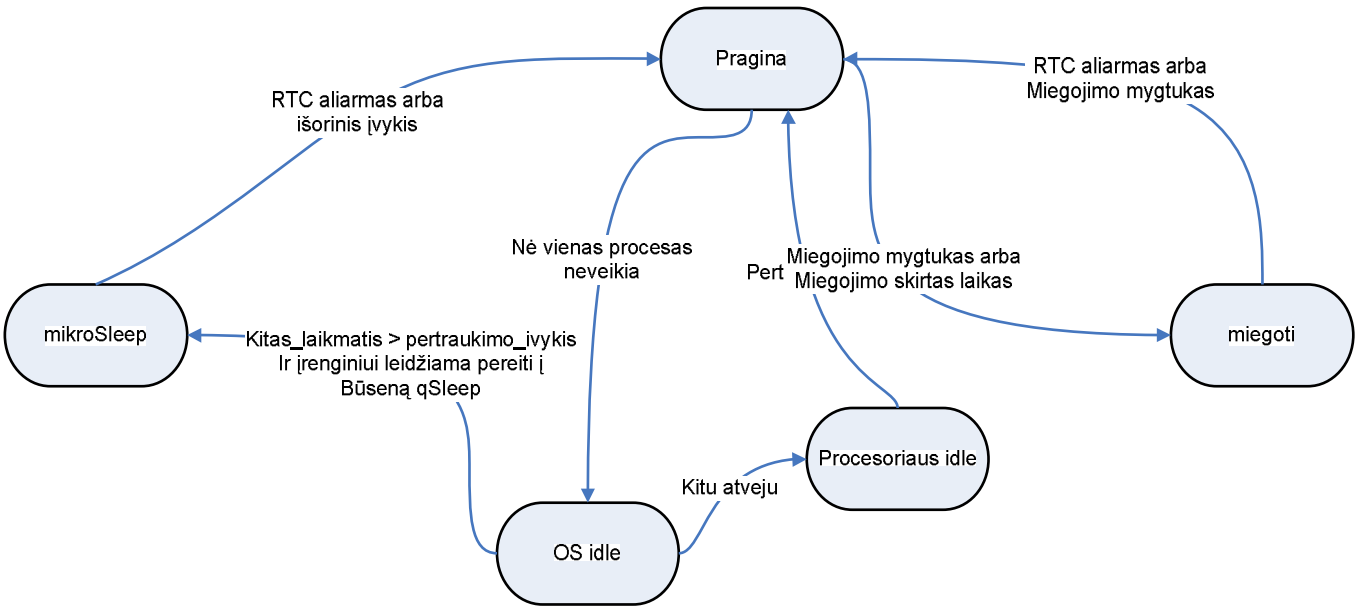
Vienintelis pokytis tai, kad prisideda vėlavimas, kai sistema pabunda, nuo išorinio įvykio pasirodymo. Tačiau šis vėlavimas nėra pastebimas vartotojo, kadangi ir blogiausiu atveju, procesoriaus pažadinimo vėlinimas ir paleidimo tęsimas yra mažiau nei 12 ms.

Šis metodas išbandytas praktiškai. Naudojamas Itsy kišeninis kompiuteris. Įrenginys susideda iš StrongARM SA-1100 procesoriaus, 32 Mb liekinės atminties (flash memory), 32 Mb dinaminės atminties. Naudojat qSleep technologijos prototipą, nešiojamuose kompiuteriuose, energijos sunaudojimas sumažėjo 60%.

Energijos sumažinimo tikslas yra pasiekiamas, draudžiant procesoriui persijungti miegojimo rėžimą, nebent jis nustatė, kad miegos tiek laiko, kurio pakaks sutaupyti energiją.

Norint įvykdyti qSleep, reikalingi specifiniai aparatūrinės ir programinės dalies reikalavimai. Yra keturi aparatūrinės dalies reikalavimai:

1. Procesorius turi turėti miegojimo rėžimą. Tai yra bendra moderniem procesoriams, kurie skirti žemos energijos taikymui.
2. Įrenginys turi turėti galimybę parodyti statinį paveikslėlį, kol procesorius miega. Tai gali būti pasiekta daugeliu atvejų; pvz., jeigu procesorius gali laikyti aktyvų integruotą LCD reguliatorių, kol procesorius eina miegoti. Turint LCD reguliatorių išorėje procesoriaus arba naudojant LCD ekraną su sudarytu freimo buferiu, kuris gali parodyti paveikslėlį frame buferyje, kai LCD reguliatorius išjungtas.
3. Sistema turi turėti galimybę pabusti veikiant išoriniams įvykiams, tokiems kaip mygtuko paspaudimas arba valdomo ekrano palietimas.
4. Įrenginys turi turėti programuojamą laikmatį, tobulai su geba nuo 1 iki 10 ms, kuris gali pažadinti sistemą. Šitas laikmatis yra naudojamas sistemos pažadinimui prieš kitą suplanuotą operacinės sistemos įvykį.



2 Pav. Qsleep vykdymo būsenų diagrama

2 paveikslas parodo sistemos būsenas, kai vykdomas qSleep. Šiame vykdymo pavyzdyje, realaus laiko laikrodis (RTC – real time clock) yra naudojamas, kaip qSleep pažadinimo laikrodis. Pagrindinė būsena yra tada, kada sistema veikia; veikia procesas arba gija, arba paleidžiamas branduolio kodas proceso ar gijos vardu. Sistema gali pereiti į savo miegojimo būseną, kaip rezultatas duoto mygtuko paspaudimas arba kaip rezultatas neveiklaus laikmačio. Į OS laukimo būseną (OS idle) pereinama tada, kai OS laukimo

būsenos procesas pradeda veikti, rezultate gauname, kad nė vienas procesas ar gija negali pasileisti. Kai sistema pereina į OS laukimo būseną, ji patikrina ką ji gali padaryti trumpo laiko miegojimui. Patikrinimas susideda iš lemiamo veiksnio ar sistema gali miegoti pakankamai ilgai, tam, kad sutaupyti energiją ir jei visi įrenginiai leidžia procesoriui eiti į miegojimo rėžimą. Jeigu taip, sistema eina į qSleep būseną, kitu atveju sistema eina į procesoriaus laukimo būseną.

Išanalizuotas qSleep metodas turi privalumų ir trūkumų. Privalumai:

- Sutaupo energijos vartojimą iki 60 %
- Vartotojas nepastebi, kad procesorius miega
- Mažas vėlinimo laikas, iki 12ms

Pagrindinis metodo trūkumas – reikia specialios aparatūrinės įrangos. Delninukas turi atitikti jam keliamus reikalavimus, tam, kad galėtų įvykdyti metodą. Todėl senuose kišeniniuose kompiuteriuose qSleep nepavyks paleisti.

2.4.3 Objektiškai orientuoto programavimo strategijos, naudojant C# programavimo kalbą, nešiojamuose įrenginiuose

Kayun Chantarasathaporn ir Chanowat Srisa-an [3] atliko tyrimą tema: Objektiškai orientuoto programavimo strategijos, naudojant C# programavimo kalbą, nešiojamuose įrenginiuose. Tyrimas yra pradininkas, kuris bando specifikuoti ir pasiūlyti optimizuotą OOP programavimo strategiją, rašant programas, kurios skirtos įrenginiams su išorine baterija. Metodas paremtas tikrais rezultatais.

Tyrime autoriai bandė tirti energijos suvartojimą lyginant OOP naudojimą. Buvo naudojami OOP programavimo elementai: klasės, metodai, atributai ir t.t. Buvo suprogramuotos paprastos programėles. Sakykim viena programa suprogramuota naudojat klases, o kita – struktūras. Programos paleidžiamos ir žiūrima per kiek laiko procesorius suskaičiuoja ir kiek atminties užima. Ir daroma išvada, kad programa parašyta naudojant struktūras, resursų naudoja mažiau. Taip atliktas visas eksperimentas su įvairiais programavimo elementais.

2 lentelė. CPU laiko naudojimas, skirtingose C# OOP programavimo sąlygose

ISSUE	SUB-ISSUE	MAX PROCESSOR TIME (MILLISECONDS)				MEMORY (KILOBYTES)	
		User Time	Privilege Time	Total Time	Difference (%)	Total Memory	Difference (%)
Classes & Structs	Class (data members only)	3,212.62	20.03	3,232.65	*used as base*	3,705.60	3.21
	Struct	1,808.60	18.03	1,826.63	43.49	3,828.40	*used as base*
Prototypes	Abstract Class	986.42	24.03	1,010.45	*used as base*	3,768.80	1.17
	Interface	992.43	16.02	1,008.45	0.20	3,813.60	*used as base*
Fields	Dynamic Field	3,301.75	24.03	3,325.78	*used as base*	3,730.80	1.34
	Static Field	1,811.60	18.03	1,829.63	44.99	3,781.60	*used as base*
Methods	Dynamic Method	3,571.14	16.02	3,587.16	*used as base*	3,771.20	*used as base*
	Static Method	1,620.33	24.03	1,644.36	54.16	3,744.80	0.70
Local Field Accessibility	Private	1,802.59	16.02	1,818.62	43.85	3,770.00	0.22
	Protected	1,804.59	17.02	1,821.62	43.75	3,778.40	*used as base*
	Public	3,219.63	19.03	3,238.66	*used as base*	3,700.00	2.07
Local Method Accessibility	Private	906.30	17.02	923.33	1.39	3,761.20	1.13
	Protected	911.31	22.03	933.34	0.32	3,798.80	0.14
	Public	914.31	22.03	936.35	*used as base*	3,804.00	*used as base*
Field Accessibility from Other Class	Protected	1,805.60	18.03	1,823.62	43.53	3,810.00	*used as base*
	Public	3,214.62	15.02	3,229.64	*used as base*	3,750.40	1.56
Method Accessibility from Other Class	Protected	708.02	17.02	725.04	0.82	3,845.20	*used as base*
	Public	711.02	20.03	731.05	*used as base*	3,821.20	0.62
Anonymous & Named Object	Anonymous Object	4,376.29	105.15	4,481.44	*used as base*	4,001.60	*used as base*
	Named Object	3,510.05	15.02	3,525.07	21.34	3,717.20	7.11

Iš 2 lentelės matosi tyrėjų gauti testų rezultatai. Gautos tyrimo rekomendacijos, gali būti kelrodis, kuris gali padėti programų kūrėjams geriau optimizuoti programas.

Naudojant šį tyrimo metodą galima šiek tiek optimizuoti programos kodą ir tai jau leistų programuotojams sumažinti energijos suvartojimą. Aišku, nežymiai, nes dar daugelis elementų nėra ištirta.

Dar vienas literatūroje aprašytas metodas: resursų rezervavimo metodas, skirtas energijos prognozės planavimui. Autoriai Claudio Scordino ir Giuseppe Lipari [4] pristato GRUB-PA algoritmą. Naujas planavimo algoritmas skirtas energijos sistemoms. Algoritmas gali efektingai tvarkyti sistemą, kuri susideda iš sunkių ir lengvų realaus laiko užduočių. Užduotys gali būti periodiškos, pavienės arba neperiodinės. Algoritmas gauna išretintą dažnių juostos plotį, iššauktą periodiškų užduočių, kurios paleidžiamos mažiau nei tikimasi arba pavienės užduotys, kurios atvyksta mažesniu dažniu ir naudoja šią informaciją naudodamos mažesnį procesoriaus dažnį.

GRUB-PA (Greedy Reclamation of Unused Bandwidth-Power Aware) algoritmas. Pirma nusistatome, kad procesoriaus greitis gali būti įvairus, nuo maksimalaus greičio iki

minimalaus. Algoritmas paremtas procesoriaus dažnio sumažinimu. Jeigu procesoriaus dažnių juosta nenaudojama ji sumažinama tiek, kad pakaktų įvykdyti užduotis. Procesorius dirba optimaliai, nevartodamas nereikalingos energijos. Tokiu būdu sumažinama sunaudojama energija.

Kadangi tai matematinis algoritmas, todėl jis aprašytas formulėmis ,bei teoremomis, kurias galima rasti autorių straipsnyje.

Eksperimento rezultatai rodo, kad naudojant algoritmą, buvo sutaupyta 38.4% bendros sistemos energijos eikvojimo.

2.4.3 Lietuvoje atliekami eksperimentai

Lietuvoje taip pat atliekami bandymai, kurie padėtų ištirti energijos suvartojimą nešiojamuose įrenginiuose su baterija. Vienas eksperimentas buvo atliktas KTU. J.Valančius ir V.Štuikys tyrė delninuko energijos suvartojimo įvertinimą taikomosios programos lygmenyje.[5] „Energijos suvartojimo įvertinimas taikomosios programos lygmenyje literatūroje dažniausiai siejamas su programomis, įvertinančiomis energijos suvartojimą - simuliatoriais (angl. *profiler, simulator*). Šios programos gali simuliuoti tam tikro taikymo (programos) energijos suvartojimą viename mobiliojo įrenginio lygmenyje (kompiliatoriaus, operacinės sistemos).

Jie pasiūlė delninuko energijos suvartojimo įvertinimo modelį, taikomosios programos lygmenyje, paremtą juodos dėžės metodu. Juoda dėžė apibrėžiama kaip sistema, kurios vidinė struktūra nekinta.

3 pav. Delninuko taikymo energijos suvartojimo įvertinimas juodos dėžės metodu.

Taikomoji programa arba taikymas yra bet kokia programa, veikianti įrenginio operacinėje sistemoje.

Įvertinant energijos suvartojimą delninuke juodos dėžės metodu atliekami šie žingsniai: a) delninuko energiją naudojančių komponentų išskyrimas, b) energiją naudojančių komponentų veiksenų nustatymas ir galimų taikymų išskyrimas, c) taikymų energijos suvartojimo matavimas programiniu matavimo metodu, d) taikymų energijos suvartojimo įvertinimas.

Kiekvienas delninukas turi specifinius energiją naudojančius komponentus, besiskiriančius techninėmis charakteristikomis. Tai ekranas, garsiakalbis, procesorius, atmintis, ryšio sistema ir kiti komponentai. 3 lentelėje aprašyti visi delninuko energiją vartojantys komponentai

3 lentelė. Delninuko energiją vartojantys komponentai

Komponento pavadinimas	Veiksenos būsena
Procesorius ir atmintis (CPĮ)	Visada jungti
Garsiakalbis (G)	Min, max, tarpinis garsumas
Ausinės (A)	Min, max, tarpinis garsumas
Ekranas (E)	Min, max ryškumas
Ryšys (R)	Bluetooth ryšys (BT)
	Infrared ryšys (IR)
	Plačiajuostis ryšys (WIFI)

Eksperimentas atliktas su delninuku Palm Zire 72, kuris leidžia įvertinti šio delninuko energijos suvartojimą. Palm Zire 72 delninuke veikia Palm OS operacinė sistema, pats delninukas pritaikytas spręsti multimedia uždavinius.

2.4 lentelėje išvardintos kelių delninuko Palm Zire energiją naudojančių komponentų veiksenos ir nustatoma taikymo galimybė. Laukelis „Galimas“ nurodo, ar veiksena galima („+“) ar negalima („-“) susieti į loginę seką – taikymą. Parametras „Garso l.“ reiškia delninuko garso lygį: aukštą („Max“) ir žemą („Min“). Parametras „Vaizdas“ reiškia vaizdo ryškumą delninuko ekrane: didžiausią („Max“) ir mažiausią („Min“). Parametras „Perdavimo būdas“ reiškia vieną iš kelių delninuko failo perdavimo būdų: Bluetooth („BT“) ar Wifi tinklu („WIFI“).

2, 6, 10 taikymai negalimi, nes delninukas ryšio sesijos metu veikia įjungtu ekranu. 12 – 15 taikymai negalimi, nes delninuke vienu metu negali veikti ausinės ir garsiakalbis. 5, 7, 9 – nematuojami, nes yra sudėtiniai, pailgintų eksperimento trukmę, juos galima paskaičiuoti teoriškai.

4 lentelė. Palm Zire 72 delninuko energiją naudojantys komponentai, jų veiksenos ir parametrai

Eil. nr.	G	A	R	E	Galimas taikymas	Garso l.		Vaizdo ryškumas		Perdavimo būdas	
						Min	Max	Min	Max	BT	WIFI
1	0	0	0	1	+	-	-	+	+	-	-
2	0	0	1	0	taikymas negalimas						
3	0	0	1	1	+	-	-	+	+	+	+
4	0	1	0	0	+	+	+	-	-	-	-
5	0	1	0	1	+	+	+	+	+	-	-
6	0	1	1	0	taikymas negalimas						
7	0	1	1	1	+	-	-	+	+	+	+
8	1	0	0	0	+	+	+	-	-	-	-
9	1	0	0	1	+	+	+	+	+	-	-
10	1	0	1	0	taikymas negalimas						
11	1	0	1	1	+	-	-	+	+	+	+
12	1	1	0	0	taikymas negalimas						
13	1	1	0	1	taikymas negalimas						
14	1	1	1	0	taikymas negalimas						
15	1	1	1	1	taikymas negalimas						

Išmatavę eksperimento rezultatus įsitikiname, jog delninukas daugiausiai energijos suvartoja veikdamas didžiausiu pajėgumu. Mažiausiai energijos suvartoja taikymai, kai garsas atkuriamas ausinuku. Didžiausias energijos naudotojas – bevielio ryšio sistema Wifi.

Tačiau tai tik vienas eksperimentas ir jo metu, kaip ir daugelyje tyrimų, bandymas buvo atliekamas su garso failais. O kiek dirbant programai baterijos energijos sunaudoja procesorius ir atmintis nėra ištirta, arba mes neradome atitinkamų straipsnių šia tema.
Norint ištirti, kiek energijos sunaudoja procesorius ir atmintis, reikalingas rezultatų gavimo tikslumas. Tyrimo metu turėtų būti išjungti visi kiti faktoriai, kurie įtakoja baterijos iškrovimą. Tokiu būdu gauti rezultatai būtų tikslūs ir neklaidingi. Bandymų metu turi būti išjungtas monitorius, kadangi jis įtakoja daugiausiai energijos suvartojimo. Eksperimento metu tektų apkauti procesorių ir atmintį. Kaip tai padaryti ir kokius metodus naudi, aptarsime sekančiame skyriuje.

2.4.4 Išretintų matricų panaudojimas realiose taikomosiose programose

Išretintos matricos plačiai naudojamos realiose kuriamose programose, tai įrodo Floridos universiteto išretintų matricų kolekcija [6]. Floridos universitete yra didelė išretintų matricų kolekcija, kuri visą laiką plečiasi. Čia talpinamos tik tokios matricos kurios naudojamos realiose programose. Matricos apima platų probleminių sričių panaudojimo spektrą. Kolekcija atitinka esminę svarbą, kurios negali atitikti dirbtinai sugeneruotos matricos ir yra plačiai naudojamos išretintų matricų algoritmų komuniteto, kuris projektuoja ir vertina išretintų matricų algoritmų spartą.

Matricų kolekcija padalinta į 99 skirtingas matricų grupes, su galimybę grupių skaičių praplėsti, kai įdedamos naujos matricų kolekcijos. Štai keletas kolekcijų:

- Harwell-Boeing kolekcija
- Tiesinio programavimo uždaviniai
- Stanfordo/Berklio tinklo matricos
- 2D ir 3D puslaidininkių fizika

2007 metų rugsėjį, kolekcija susidėjėjo iš 1877 problemų. Mažiausia matrica yra 5 eilučių ir 5 stulpelių, kuri turi 19 nenulinių elementų. Didžiausios matricos dimensija – 9.8 milijono, kuri turi apytiksliai apie 99.2 milijono nenulinių elementų. Visos matricos pateikiamos trimis formatais: Matlabo mat-faile, Rutherfordo-Boeingo ir matricų rinkos (market). Visos kolekcijos dydis yra apie 9 GB.

2.5 Architektūros ir galimų įgyvendinimo priemonių variantų analizė

Mes siekiame ištirti, kokią įtaką bendram energijos suvartojimui turi procesorius ir atmintis. Tyrimo įgyvendinimui reikia apkrauti tiek procesorių, tiek atmintį. Todėl reikia išanalizuoti ir pasirinkti atitinkamus realizavimo metodus, kurie leis įgyvendinti eksperimentą.

Pati programa turėtų būti rašoma pritaikant objektiškai orientuotą programavimą (OOP). Objektinis programavimas (OOP) - tai ne vien programinės įrangos kūrimo teorija, kiek pačių programų rašymo būdas. Nemažai žmonių mano OOP esant viena reikšmingiausių kada nors egzistavusių programinės įrangos kūrimo naujovių. OOP - tai būdas realaus gyvenimo objektus pateikti kaip programavimo kalbos objektus. Tai yra viena populiariausių šių laikų programavimo technologijų, kuri plačiai naudojama informacinių sistemų kūrime.

Toliau reikia pasirinkti platformą, kurią naudojant būtų kuriama testavimo programa. Tam geriausiai tinka .NET platforma, kadangi tai yra irgi populiari ir daug kur naudojama projektavimo platforma. **Microsoft.NET Framework** [7] yra Microsoft Windows operacinės sistemos komponentas, sukurtas 2002 metais. Jis suteikia kitoms programoms galimybę naudotis daugybe jau paruoštu įvairių bibliotekų (pvz., duomenų bazių komponentus, formų komponentus...). Be to, šis komponentas ir tvarko programos kodą jos vykdymo metu, jei programa parašyta specialiai šiam paketui (sukompiliuota su CIL suderinamu kompiliatoriumi). Tai reiškia, kad programa vienodai gerai turėtų veikti įvairiose platformose; nėra būtinybės 64-ių bitų procesoriams skirtą CIL programą perkompiliuoti į 32-ų bitų skirtą procesoriams programą. Visa tai atliekama labai greitai ir automatiškai. .NET platformos panaudojimas darbe palengvina programos rašymą, kadangi parašytą programą galima sukompiliuoti ir paleisti įvairiuose įrenginiuose, kur yra įdiegta Framework biblioteka. Tai liečia tiek delninius kompiuterius, kuriuos naudojame eksperimente, tiek nešiojamus kompiuterius.

Pasirenkame vieną iš objektiškai orientuotų programavimo kalbų. **C#** – objektiškai orientuota programavimo kalba, sukurta Microsoft kompanijoje kaip dalis .NET iniciatyvos. Kalba paremta C++ bei Java kalbomis. Kalba kurta balansuojant tarp galingumo (C++ įtaka) bei greito programavimo (Java, Visual Basic įtaka).

Kai jau turime išanalizavę ir matomai pasirinkę visus dalykus, belieka pasirinkti metodą, kurio pagalba būtų atliekamas tyrimas. Procesorių apkraut puikiai tinka aritmetikos veiksmai, pvz. daugyba. Dauginant didelius skaičius, daug kartų apkraunamas procesorius, o dauginamieji skaičiai yra saugomi atmintyje. Norint iškrauti kišeninio kompiuterio bateriją reikalingas ilgas dauginimo procesas, kuris leistų nepertraukiamai atlikti operaciją. Akivaizdu, kad dauginant du skaičius, kad labai didelius, tokio proceso nelaibai išgausime. Todėl tokiam atvejui panaudosime matricas, bei jų daugybą. Daugindami dvi matricas, tam tikro dydžio, pilnai apkausime kompiuterio procesorių ir galėsime reguliuoti dauginimo proceso laiką. Naudosime paprastą matricų daugybą ir tam tikrą dauginimo metodą, kuris leistų sutaupyti dauginimo laiko ir baterijos energijos. Sakykim turime dvi matricas ir jas reikia sudauginti. Dauginame jas paprastu būdu, kaip ir matematikoje ir dauginsime naudodami, taip vadinamą išretintų matricų meto. Kuris turėtų matricų daugybą atlikti griečiau negu dauginant paprastu metodu.

2.6 Siekiamos sistemos apibrėžimas

Mes siekiame sukurti programą, kuri bus skirta kišeninio kompiuterio baterijos testavimui. Programa atliks kelių metodų daugybą. Daugybos metu bus renkami baterijos atitinkami duomenys, kurie padės sudaryti atitinkamas išvadas apie naudojamo metodo efektyvumą ir nustatyti procesoriaus ir atminties įtaką.

2.7 Darbe naudojama programinė įranga

Aprašyti matematiniai modeliai bus realizuojami su MS Visual Studio 2005. Naudojama programavimo kalba C#. Parašytos programos bus vykdymos delniniuose kompiuteriuose, kuriuose yra Windows Mobile 5.0 operacinė sistema. Vykdymo rezultatai bus atvaizduojami per vartotojo sąsają. Pradiniai duomenys ir galutiniai rezultatai nuolat kaupiami ir saugomi faile.

Programoje bus realizuota išretintų matricų daugyba. Pasinaudojus išretintų matricų savybe, kuri leidžia sumažinti matricos formatą, išmetant nulinius elementus, galime stebėti, kaip keičiasi sistemos resursų sunaudojimas, keičiant pradinėje matricoje nulinių elementų skaičių.

2.8 Analizės išvados

- Baterijos gyvavimo ilgaamžiškumas išlieka pagrindine šių dienų problema.
- Atliekami bendri tyrimai šia kryptimi, tačiau nėra atlikta tyrimų, kurie pasakytų, kokia yra procesoriaus ir atminties įtaka bendram energijos suvartojimui.
- Šiuolaikinėse informacinėse sistemose naudojami objektiškai orientuoti programavimo principai ir pažangiausios technologijos – C# programavimo kalba.
- .NET platforma leidžia kurti programas, kurios skirtos įvairiom aplinkom.
- Išretintų matricų metodas plačiai naudojamas ir tinka sistemos resursų stebėjimui, kadangi keičiant įvairius parametrus, keičiasi procesoriaus ir atminties panaudojimas

3. Projektinė dalis

3.1 Išretintų matricų modelis

Darbe naudosime išretintų matricų metodą [8]. Matrica - stačiakampė mn skaičių lentelė, susidedanti iš m eilučių ir n stulpelių. Žymima:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \text{ arba } A = (a_{ij}), i = \overline{1,m}, j = \overline{1,n}.$$

Elementai a_{ij} vadinami matricos elementais; pirmasis indeksas i žymi eilutės, kurioje yra elementas, numerį, o antrasis indeksas j - stulpelio numerį. Kai matricoje yra m eilučių ir n stulpelių, ji vadinama $[m \times n]$ formato matrica. Išretintomis matricomis vadinamos matricos, kurios turi santykinai mažą skaičių nenulinių elementų. Išretinta matrica galima saugoti pilnų matricų metodų pagalba. Kai išretinta matrica yra saugoma, visi jos elementai, įskaitant ir nulinius, yra talpinami į masyvus. Darbe naudosime du saugojimo metodus: suspaustų matricų saugojimo metodą ir stulpelių saugojimo metodą.

Suspaustų matricų saugojimo metodas

Išretintą matricą A, saugant suspaustos matricos metodu, naudojami du dviejų dimensijų masyvai, nusakantys saugojamą matricą, AC ir KA. [m x n] formato išretinta matrica A, turi maksimumą *nz* nenuliniių elementų kiekvienoje eilutėje:

- AC yra apibrėžiamas kaip AC(lda, nz), kur pagrindinė dimensija, lda, turi būti didesnė arba lygi m. Kiekviena eilutė masyvo AC turi nenulinių elementų, kurie atitinka eilutes matricoje A. Kiekvienai eilutei matricoje A, turinčiai mažiau nei *nz* nenulinių elementų, atitinka masyvo AC eilutes su išmestais nuliais. Elementai kiekvienoje eilutėje gali būti saugomi bet kokia tvarka.
- KA yra sveikų skaičių masyvas apibrėžiamas kaip KA(lda, nz), kur pagrindinė dimensija lda, turi būti didesnė ar lygi m. Tai yra susiję su matricos A stulpeliu skaičiumi. Elementai, kurie yra laikomi KA, atitinka pozicijas masyve AC. Kiekviena matricos A eilutė, kuri susijusi su mažiau negu NZ ne nulinių elementų, atitinka KA masyvo eilutę yra išmesta su reikšme nuo 1 iki n.

Toks matricų saugojimo būdas reikalauja daug atminties. Tai riboja efektyvumą, kurį galima pasiekti naudojant šitą matricų saugojimo metodą.

PVZ. Matrica A 6x6. Maksimalus skaičius ne nuliniu elementų kiekvienoje eilutėje – 4. Parodysime, kaip matrica A gali būti laikoma masyvuose AC ir KA. Duota matrica A:

$$A = \begin{pmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 51 & 0 & 0 & 54 & 55 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{pmatrix}$$

Tada masyvai atrodytų taip:

$$AC = \begin{pmatrix} 11 & 13 & 0 & 0 \\ 22 & 21 & 24 & 0 \\ 33 & 32 & 35 & 0 \\ 44 & 43 & 46 & 0 \\ 55 & 51 & 54 & 0 \\ 66 & 61 & 62 & 65 \end{pmatrix} \quad KA = \begin{pmatrix} 1 & 3 & * & * \\ 2 & 1 & 4 & * \\ 3 & 2 & 5 & * \\ 4 & 3 & 6 & * \\ 5 & 1 & 4 & * \\ 6 & 1 & 2 & 5 \end{pmatrix}$$

* reiškia, kad galima saugoti bet kokia pozicijos reikšmę nuo 1 iki 6 šitame masyve.

Bendrai šitą saugojimo techniką galima išreikšti:

kiekvienam $a_{ij} <> 0, kai \quad i = 1, m$ ir $j = 1, n$ egzistuoja k, kur $1 <= k <= nz$, toks kad $AC(i,k) = a_{ij}$ ir $KC(i,k) = j$.

Kur:

- a_{ij} yra maksimumas nenulinių elementų nz kiekvienoje eilutėje. Elementai yra matricos A, kurios formatas $[m \times n]$.
- Masyvas AC aprašomas kaip AC(lda, nz), kur lda>=m.
- Masyvas KA aprašomas kaip KA(lda, nz), kur lda>=m.

Indeksų saugojimo metodas

Čia naudojami 3 vienos dimencijos masyvai, kurie apibūdina išretintą matricą. Tie masyvai – AR, IA ir JA. Duota mxn matrica A, kuri turi ne nenulinių elementų. Masyvai sudaromi taip:

- AR. Dydis ne, kuris susiję su NE ne nulinių matricos A elementų. Jie masyve AR išdėstomi, bet kokia tvarka.
- IA – integer masyvas, dydžio (ne), susiję su atitinkamais eilučių numeriais, kuriuose yra ne nulinis elementas Aij matricoje A.
- JA – integer masyvas, susijęs su elemento vieta stulpelyje matricoje A

Tarkim turime duotą matricą A:

$$A = \begin{pmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{pmatrix}$$

Tuomet masyvai AR, IA, JA atrodys taip:

$AR = (11,22,32,33,13,21,43,24,66,46,35,62,61,65,44)$

$IA = (1,2,3,3,1,2,4,2,6,4,3,6,6,6,4)$

$JA = (1,2,2,3,3,1,3,4,6,6,5,2,1,5,4)$

Bendrai šitą saugojimo techniką galima išreikšti:

kiekvienam $a_{ij} <> 0, kai \quad i = 1, m$ ir $j = 1, n$

egzistuoja k, kur $1 <= k <= ne$, toks kad $AR(k) = a_{ij}$ ir $IA(k) = i$, $JA(k) = j$.

Kur:

- a_{ij} yra elementai išretintos matricos A
- Masyvai AR, IA, JA turi *ne* elementus.

Stulpelių saugojimo metodas

Išretintai matricai A, laikymas pagal stulpelius naudoja tris vienos dimencijos masyvus, kurie skirti apibūdinti išretintai matricai. Tai masyvai AR, IA, JA. Duota m kart n išretinta matrica A, kuri turi *ne* nenulinių elementų, masyvai sudaromi tokia tvarka:

- AR masyvas yra bent *ne* dydžio, sudarytų iš *ne* nenulinių, išretintos matricos A elementų, laikomų gretimai. Matricos A stulpeliai laikomi iš eilės nuo 1 iki n masyve AR. Matricos A kiekvieno stulpelio elementai matricoje AR laikomi bet kokia tvarka.
- IA – integer tipo masyvas, *ne* dydžio, jame laikomi kiekvieno nenulinio elemento Aij, stulpelio numeris matricoje A.
- JA – integer masyvas, kurio dydis bent n+1, laikomi matricos A kiekvieno stulpelio pradžios pozicija. Tai yra kiekvienas JA(j) elementas parodo, kur prasideda j stulpelis masyve AR. Jeigu visi elementai j stulpelyje yra nuliai, tada JA(j) = JA(j+1). Paskutinis elementas, JA(n+1) parodo pozicija po paskutinio AR elemento, kuri yra ne+1

Tarkime turime matrica A:

$$A = \begin{pmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 0 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 61 & 62 & 0 & 0 & 0 & 66 \end{pmatrix}$$

Tuomet masyvai AR, IA, JA atrodys taip:

$AR = (11,61,21,62,32,22,13,33,43,44,24,46,66)$

$IA = (1,6,2,6,3,2,1,3,4,4,2,4,6)$

$JA = (1,4,7,10,12,12,14)$

Jeigu kalbėtumėme bendrai, visa tai galima aprašyti taip:

kiekvienam $a_{ij} <> 0$, kai $i = 1, m$ and $j = 1, n$ egzizstuoja k, kur $1 <= k <= ne$, toks kad $AR(k) = a_{ij}$,$IA(k) = i$ ir kai $j = 1, n$,

$JA(j) = k$, kur a_{ij}, $AR(k)$ masyve, yra pirmas elementas patalpintas masyve AR kai stulpelio numeris j.
$JA(j) = JA(j+1)$, kur visi $a_{ij} = 0$ stulpelyje j
$JA(n+1) = ne+1$

Eilučių saugojimo metodas

Matricos A saugojimas naudojant eilučių metodą, remiasi 3 vienos dimencijos masyvais, kurie apibūdina matricą A. Tie masyvai yra AR, IA ir JA. Duota matrica [m x n], kuri turi *ne* ne nulinių elementų. Masyvai sudaromi tokia tvarka:

AR – dydis bent *ne*, susijęs su ne nuliniais matricos A elementais, kurie laikomi gretimai. Matricos A eilutės yra laikomos iš eilės nuo 1 iki m masyve AR. Matricos A kiekvienos eilutės elementas yra laikomas bet kokia tvarka masyve AR.

IA – integer masyvas. Dydis m+1, susijęs su kiekvienos matricos A eilutės pražios pozicija masyve AR; tai yra, kiekvienas elementas IA(i) parodo, kur eilutė i prasideda masyve AR. Jeigu eilutės visi elementai nuliai, tada IA(i)=IA(i+1). Paskutinis elementas IA(m+1) parodo poziciją po paskutinio masyvo AR elemento, kuris yra ne+1

JA – integer masyvas. Dydis bent (ne), susijęs su stulpelio numeriu kiekvieno nenulinio matricos A elemento a_{ij} .

Tarkime turime duotą matrica A:

$$A = \begin{pmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 0 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 61 & 62 & 0 & 0 & 0 & 66 \end{pmatrix}$$

Tuomet masyvai AR, IA, JA atrodys taip:

$AR = (11,13,24,22,21,32,33,44,43,46,61,62,66)$
$IA = (1,3,6,8,11,11,14)$
$JA = (1,3,4,2,1,2,3,4,3,6,1,2,6)$

Bendrai šitą saugojimo techniką galima išreikšti:

kiekvienam $a_{ij} <> 0, kai \;\; i = 1, m$ ir $j = 1, n$

egzistuoja k, kur $1 <= k <= ne,$ toks kad $AR(k) = a_{ij}$, $JA(k) = j$ ir kai $i = 1, m,$

$IA(i) = k$, kur a_{ij} yra masyve AR(k) pirmas elementas patalpintas kai eilutės numeris i.

$IA(i) = IA(i+1)$, kur visi $a_{ij} = 0$ kiekvienoje eilutėje i.

$IA(m+1) = ne+1$

3.1.1 Matricos modelių saugojimo palyginimas

Norėdami įsitikti išretintų matricų saugojimo metodo efektyvumu, palyginsime du saugojimo metodus: suspaustą ir nesuspaustą. Įrodymui susidarome vieną išretintą matricą.

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \end{pmatrix}$$

Išretinta matrica apibūdinama nenulinių elementų skaičiumi. Mes projekte naudosime išretinimą išreikštą procentais. Sakykime matricos išretinimas yra 90%, tai reiškia, kad matricoje yra 10% nenulinių elementų, o visi kiti likę 90% yra nuliniai elementai.

Prieš tai aprašytoje matricoje A yra 7 nenuliniai elementai ir 9 nuliniai elementai. Matricos A išretinimą gauname 56%. Atlikdami daugybos veiksmą su tokia matrica, dauginami visi elementai. Tokiu atveju susidaro nereikalingų dauginimo operacijų, nulio daugyba iš nulio. Įvairūs išretintų matricų saugojimo būdai pašalina iš matricos nulinius elementus. Naudojame stulpelių saugojimo metodą. Į pirmą masyvą surašome nenulinius elementus, gauname AR=(1,1,1,1,3,1,2). Kiti du masyvai aprašys matricos elementų vietą matricoje. Masyvas IA aprašys elemento stulpelio numerį – IA=(1,4,3,4,1,2,3). Paskutinis masyvas aprašys stulpelio pradžios poziciją – JA = (1,3,4,6,8). Visa matrica aprašyta trimis masyvais. Turėdami taip išsaugotą matricą ir atlikdami daugybos veiksmą, tarpusavyje dauginami masyvo AR elementai, o kiti du naudojami tam, kad išrinktumėme atitinkama tvarka matricos AR elementai. Rezultate gauname sudaugintas matricas. Daugindami paprastą išretintą matricą ir matricą išsaugotą stulpelių metodų rezultatą gausime tą patį, tačiau turėtų skirtis dauginimo laikas.

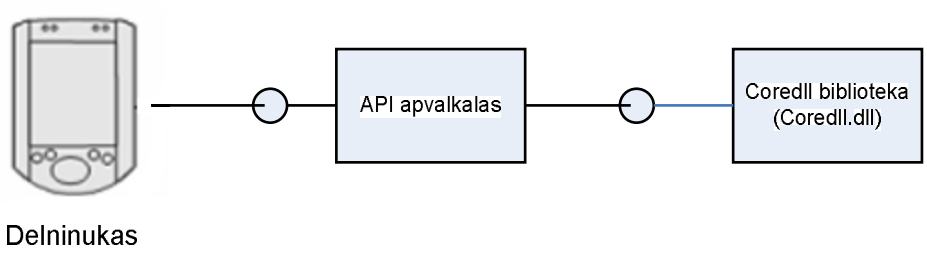
3.2 Projektuojamos informacinės sistemos modelis

Informacinė sistema (IS) – tai sistema, kurios tarpusavyje susiję komponentai dirbdami kartu surenka, apdoroja, saugo ir platina informaciją, kuri organizacijoje padeda priimti sprendimus, koordinuoti ir kontroliuoti veiklą, analizuoti problemas, vizualizuoti sudėtingus objektus, kurti naujus produktus.

Mūsų sukurta informacinė sistema surinks ir saugos testavimo duomenis, kuriais remiantis pateiksime atitinkamas išvadas apie baterijos gyvavimo laiką.

Programa kuriama naudojant Microsoft Visual Studio .NET 2003 programinę įrangą. Ši programa pritaikyta delninukų programinės įrangos kūrimui. T.y naudojant Visual Studio galime parašyti, bet kokią programą, kuri bus skirta delninukui. Parašytos programos veikia naudodamos .NET bazę. Ji suteikia programoms galimybę naudotis daugybę jau paruoštų įvairių bibliotekų (pvz. duomenų bazių komponentus, formų komponentus ...). Be to, šis komponentas ir tvarko programos kodą, jos vykdymo metu, jei programa parašyta specialia šiam paketui.

Kuriamoje programoje naudosime .NET bazės aprašytus formų komponentus, kurių pagalba perduosime reikalingus parametrus testavimo programai. Baterijos parametrai yra specifiniai ir jie nėra aprašyti .NET bazėje. Todėl norėdami gauti atitinkamus baterijos parametrus turime pasitelkti kitą metodą. Šitaip mūsų kuriama programa skyla į dvi dalis [9]. Pirmoje dalyje apsirašysime formos elementus ir patį dauginimo algoritmą, o kitoje dalyje arba projekte apsirašysime norimus stebėti baterijos parametrus, kuriuos importuosime į pirmą projektą šitaip susiedami dvi programos dalis.



4 pav. Kuriamos programos struktūra

Kuriamos programos dalis, kuri parodys delninuko baterijos informaciją, funkcijas paveldės iš Windows coredll bibliotekos. Coredll modulis yra pagrindinis operacinės sistemos modulis, kuris suteikia baranduoliui funkcionalumo su kitais moduliais ir savyje turi daug funkcijų, kurios suteikia įvairią informaciją apie sistemos konfigūraciją. Norint panaudoti šias funkcijas, reikia sukurti nuorodą į coredll.dll bylą ir panaudoti vieną iš funkcijų tam, kad galėtumėme išvardinti įrenginio baterijos informaciją, tokia kaip, koks šaltinis įjungtas, kiek liko baterijos gyvavimo procentais ir t.t.

Visa tai galima atlikti pasinaudojus API (application programming interface) sąsają. API – aplikacijų programavimo sąsaja, tai sąsaja, kurią suteikia kompiuterinė sistema, biblioteka ar programa tam, kad programuotojas per kitą programą galėtų pasiekti jos funkcionalumą ar apsikeistų su ja duomenimis.

3.3 Energijos suvartojimo modeliavimo delninukams, taikomosios programos lygmenyje projektas

Kursime testavimo programą, kuri sudaugins dvi matricas. Rezultate gausime trečią matrica ir eilę pareikalautų parametrų.

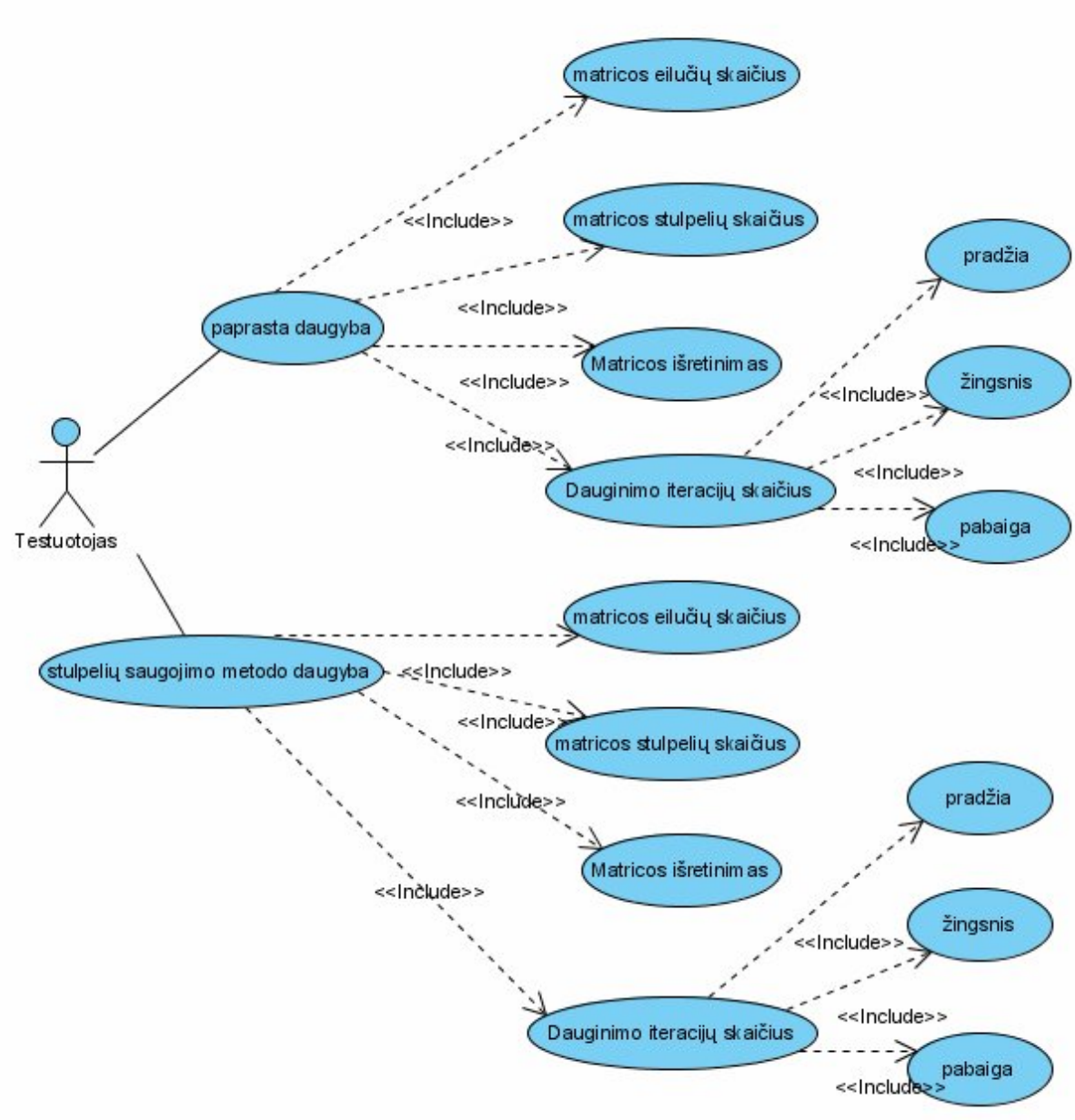
3.3.1 Kuriamos programos aprašymas

Programa turės sudauginti dvi matricas. Daugybos veiksmas pasirinktas delninuko pastoviai apkrovai palaikyti. Matrica aprašoma stulpelių ir eilučių skaičiumi. Programoje bus įvedama vienos matricos eilučių ir stulpeliu skaičius. Kitos matricos duomenys yra toki patys, t.y dauginimui naudojamos dvi vienodos matricos, todėl užtenka nurodyti vienos matricos duomenis, kitą turėsime automatiškai. Sukurtą matricą reikia užpildyti skaičiais. Tokiu atveju naudojame atsitiktinių skaičių generavimą ir jais užpildome matricą. Programos viduje nurodome maksimalų skaičių, kurį gali sugeneruoti ir užpildome matricą. Skaičiai gali būti nuo 0 iki nurodyto maksimalaus skaičiaus (maksimalaus generuojamo skaičiaus iš išorės nurodyti negalima, jis yra nurodomas programos viduje). Generuojami sveiki skaičiai.

Turėdami iš atsitiktinų skaičių sugeneruotą matricą, galime atlikti daugybos veiksmą. Dauginame matricą iš savęs pačios. T.y turime susigeneravę matricą A tai sudauginę AxA gausime kita matricą B.

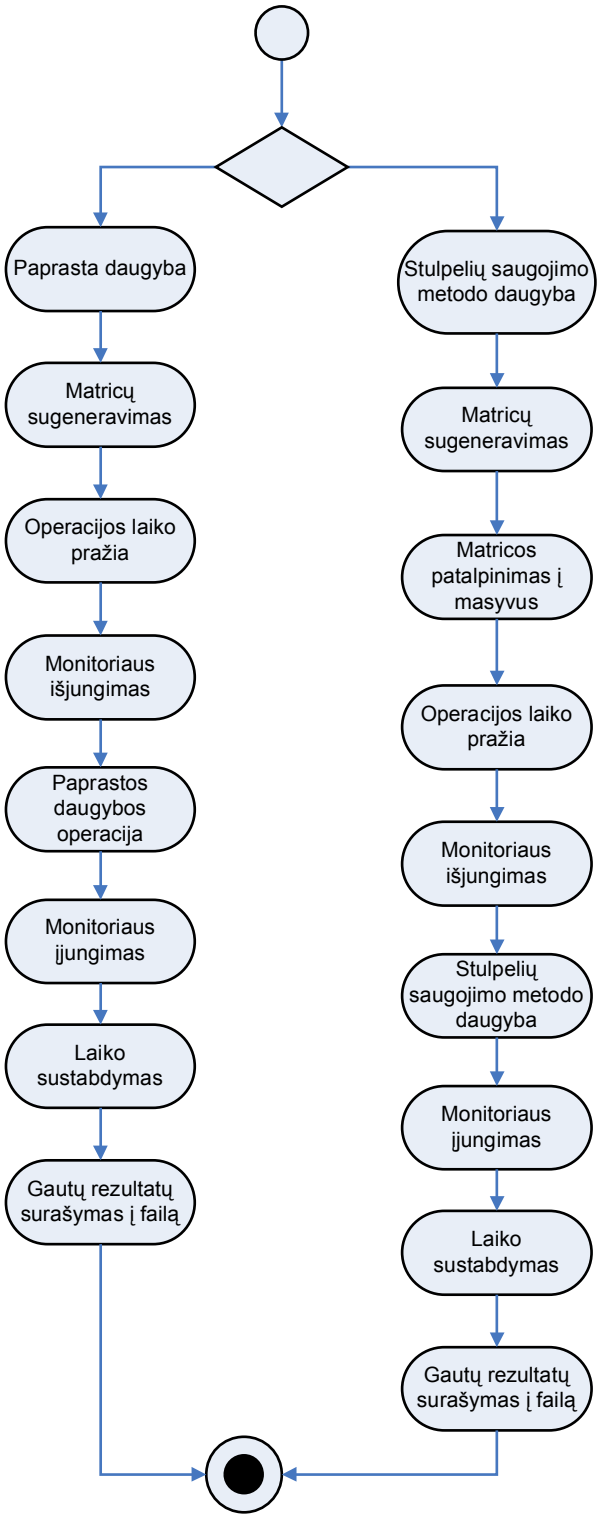
Delninukas turi ribotą atminties kiekį, todėl dauginti didelių matricų negalime, nes operacijai paprasčiausiai pritruks atminties. Tam reikia generuoti mažesnės dimensijos matricas. Tačiau tokias matricas greitai sudaugins ir apkrovimo neužteks baterijai nusodinti. Tam įvedame dauginimo iteracijas. Nurodysime, kiek kartų turi būti atlikta dauginimo procedūra. Tokiu atveju išvengiame atminties perpildymo, kadangi užtenka sugeneruoti mažos dimensijos matricą ir įvesti dauginimo iteracijų skaičių. Delninukas bus apkrautas atitinkamai tiek laiko, kiek dauginimo iteracijų įvesta.

Sistemos projekto Use case modelis pateiktas žemiau.



5 pav. Kuriamos programos panaudos atvejų diagrama

Modelis vaizduoja programos projekto sudėtį. Naudosime dvi dauginimo rūšis: paprastą matricų daugybą ir matricos stulpelių saugojimo metodo daugybą. Daugybos metu stebime ir išaugome eilę reikalingų parametrų.



6 pav. Programos veiklos diagrama

6 paveikslėlyje pavaizduota visos programos veikimą apibūdinanti veiklos diagrama. Pasirinkus vieną iš daugybų viskas vykdoma panasiai, tik sulpelių saugojimo metode atsiranda matricos patalpinimas į masyvus pagal taisykles, kadangi reikia pasiruošti daugybai. Į programą suvedus duomenis ir paspaudus vieną iš mygtukų, takim paprastą daugybą, toliau sugeneruojamos matricos, surašomi nuliniai elementai atisiktine tvarka. Paleidžiamas laiko matuoklis, išjungiamas monitorius ir pradedama dauginimo operacija tiek, kiek iteracijų įvedėme. Baigus dauginti įjungiamas monitorius ir sustabdomas laiko matuoklis ir paskui gauti rezultatai surašomi į rezultatų failą.

3.3.2 Stebimi parametrai

Atlikdami daugybos veiksmą stebėsime eilę parametrų, kurie parodys baterijos atitinkamus parametrus.

Pirmas stebimas parametras – baterijos voltažas. Parametras parodo baterijos voltų skaičių. Baterijai išsikraunant voltažas po truputį mažėja. Įtampos mažėjimas rodo, kad baterija išsikrauna. Voltažą stebėsime tam tikrais laiko tarpais, kuriuos nurodysime dauginimo iteracijų skaičiumi. Iteracijas išskaidome į intervalus. Bus iteracijų stebėjimo pradžia, tai skaičius dauginimo operacijų, kurias įvykdžius užfiksuojamas baterijos voltų kiekis. Kitas intervalo skaičius yra iteracijų žingsnis, tai skaičius, kuris parodo, kas kiek dauginimo iteracijų reikia užfiksuoti baterijos voltažą. Ir paskutinis skaičius nurodo iki kokio skaičiaus bus atliekama dauginimo operacija ir nuimamas paskutinį kartą baterijos voltų skaičius. Sakykime turime iteracijų intervalą – 100, 200, 1000. Tai reiškia, kad praėjus 100 dauginimo operacijų užfiksuojamas voltų skaičius ir toliau, kas 200 operacijų fiksuojamas baterijos voltažas tol, kol pasiekiamas galutinis skaičius – 1000.

Kitas stebimas parametras yra baterijos išsikrovimo lygis procentais. Intervalas kintantis nuo 100% iki tiek procentų, kiek mes norime iškrauti bateriją. Mūsų projekte bateriją iškrausime iki 30%. Pakrovimo procentą užfiksuosime kartu su voltų skaičiumi. Tokiu būdu išsiaiškinsime kokios procentų reikšmės atitinka voltų reikšmes.

Tai pat fiksuosime dauginimo operacijų laiką, t.y. per kiek laiko yra įvykdomos visos užsiduotos dauginimo iteracijos.

3.3.3 Paprasto ir stulpelių saugojimo metodo dauginimo algoritmai

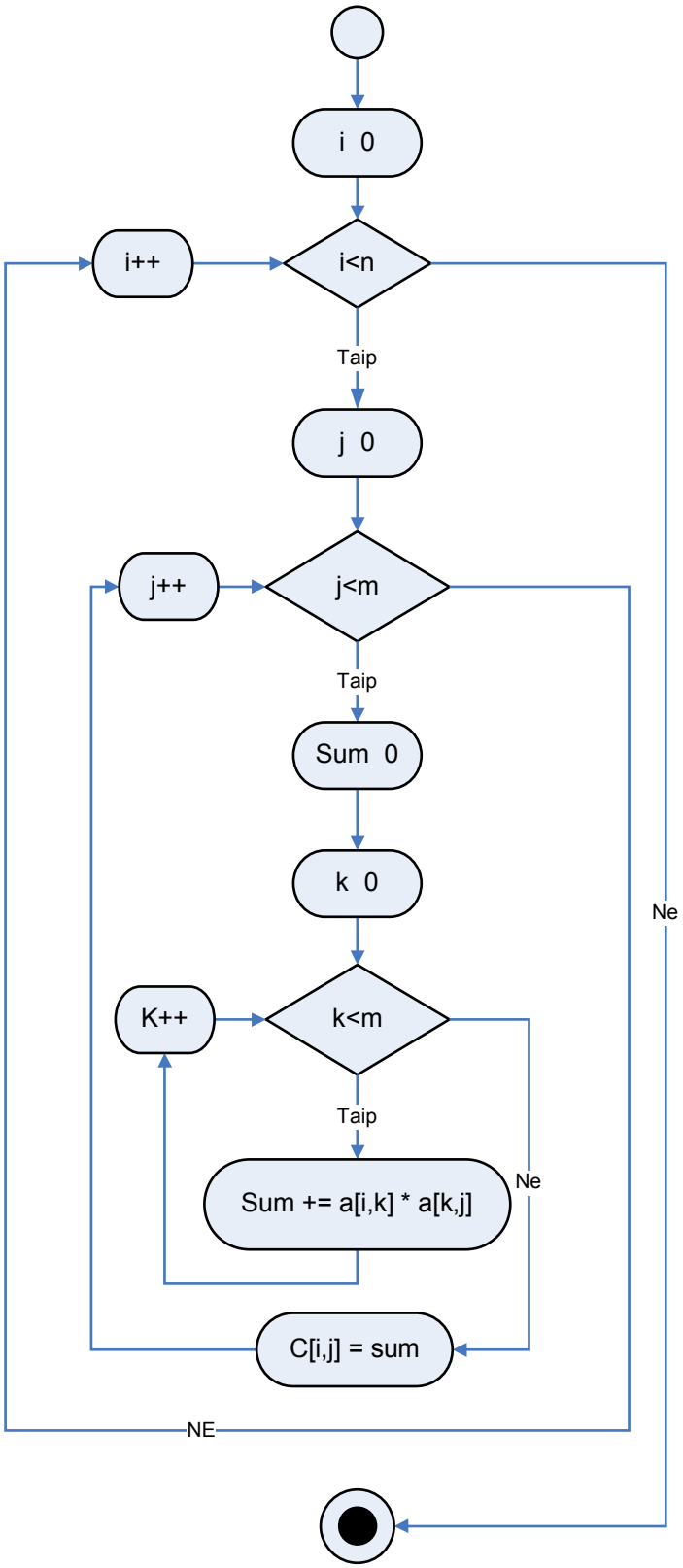
Standartinės matricos dauginimo algoritmas nėra sudėtingas. Prieš daugybą nurodome eilučių, stulpelių skaičių ir išretinimo procentą. Programa sugeneruos nurodytos dimensijos matricą, užpildydama ją atsitiktinai sugeneruotais skaičiais. Naudojant įvestą išretinimo procentą suskaičiuojama, kiek matricoje yra nulinių ir nenulinių elementų. Atsitiktiniu būdu gauti nuliniai elementai įrašomi į sugeneruotą matricą. Tai yra perrašomos tam tikros reikšmės nuliniais elementais. Pasiruošėme matricas paprastai daugybai.

Matricas galima dauginti tarpusavyje, jeigu sutampa jų vidiniai matmenys. Mes naudosime kvadratines matricas (eilučių ir stulpelių skaičius sutampa). Dauginsime matricą iš savęs pačios. Tokiu atveju užtenka sugeneruoti vieną matricą ir ją padauginti iš savęs. Daugybos metodas realizuotas pasinaudojus matricų daugybos formule.

$$c_{ij} = \sum_{k=1}^{s} a_{ik} b_{kj}$$

čia 1<= i <= m ir 1<= j <= n (m – eilutės, n – stulpeliai, a ir b matricos, c rezultatų matrica).

Iš formulės matome, kad dauginimo operacijos ilgumas priklauso nuo matricos dimensijos, todėl visiškai nėra svarbu kiek matricoje yra nulinių elementų. Visas paprasto metodo dauginimo algoritmas pavaizduotas panaudojus UML veiklos diagramą.



7 pav. Paprastos daugybos algoritmo veiklos diagrama

Veiklos diagramoje: n – matricos eilučių skaičius, m – matricos stulpelių skaičius, a – matrica, c – rezultatų matrica, i,j,k – pagalbiniai kintamieji.

Galime išvesti formulę, kiek tokios daugybos metu yra įvykdoma sudėties ir daugybos komandų. Paskaičiavę gauname, formulę - $2*m*n^2$ (m – eilutės, n – stulpeliai). Naudodami šią formulę galime nesunkiai apskaičiuoti, kiek daugybos ir sudėties komandų įvykdoma skirtingose dimensijos matricose. Sakykim turime matricą, kuri sudaryta ir 100 eilučių ir 100 stulpelių. Tokiu atveju išviso daugybos metu bus įvykdyta 2000000 (du milijonai) dauginimo ir sudėties komandų.

Stulpelių saugojimo metodo dauginimo algoritmas yra žymiai sudėtingesnis negu paprastos matricos daugybos. Dauginti reikia laikantis standartinės matricų daugybos taisyklėmis tam, kad gautumėme teisingą rezultatą. Pagrindinė esmė yra ta, kad keičiasi pačios matricos saugojimo metodas ir tai neturi įtakoti galutiniam rezultatui.

Prieš atlikinėdami daugybą, pagal išretintos matricos stulpelių saugojimo metodą (aprašytas 3.1 dalyje), susigeneruojame tris masyvus. Pirmame masyve talpinami visi matricos nenuliniai elementai. Antrame masyve laikomas stulpelio numeris, kiekvieno nenulinio elemento. Trečiame – kiekvieno matricos stulpelio pradžios pozicija. Pirmo ir antro masymo sudarymas realizuojamas C# kalboje pavaizduotas 5 lentelėje.

5 lentelė. C# išeities kodas

```
For (int i = 0; i < n; i++)
          {
           for (int j = 0; j < m; j++)
             {
                if(a[i,j] > 0 )
                   {
                     AR[g] = a[i,j];
                     IA[g] = j;
                      g++;
                    }
               }
            }
```

Čia n – matricos eilutės, m – stulpeliai, a – matrica, AR – nenulinių elementų masyvas (pirmas), IA – nenulinių elementų stulpelių numeriai, g – pagalbinis kintamasis, kuris padidinamas vienetu, jeigu randamas nenulinis elementas.

Galime atlikinėti daugybos procesą. Jeigu pasižiūrėtumėme į paprastų matricų daugybos formulę, tai pastebėtumėme, kad dauginami toki matricos elementai, kurių eilutės ir

stulpelio indeksai sutampa. Dauginama eilutė iš stulpelio. Eilutės indeksai, skaičiuojant stulpelius, sutampa su kitos matricos stulpelio indeksais, skaičiuojant eilutes. Tokiu pagrindu veikia sukurtas dauginimo algoritmas. Kadangi dauginimo algoritmas sudėtingas 6 lentelėje parodysime tik svarbiausią jo dalį.

6 lentelė. C# išeities kodas

```
for ( int j = JA[f]; j< upper; j++)
{
   if (indx <= eil)
      if (IA[j-1] == IA11[indx-1])
         {
           AR_dau[f] = AR_dau[f] + AR[j-1] * AR11[indx-1];
            indx++;
         }
        else
           {
             for (int t = indx; t <= eil; t++)
                {
                  if (IA[j-1] == IA11[t-1])
                     {
                       AR_dau[f] = AR_dau[f] + AR[j-1] * AR11[t-1];
                      }
                       else
                          {
                            AR_dau[f]= AR_dau[f] + 0 ;
                          }
                  }

            }

}
```

Čia JA – masyvas, kuriame saugomi matricos kiekvieno stulpelio pradžios pozicija, upper – kintamasis, kuris lygus JA masyvo n+1 nariui, eil – kintamasis, kuris parodo, kiek eilutėje yra nenulinių elementų, indx – pagalbinis kintamasis, kuris didinamas vienetu iki tiek, kiek yra eilutėje nenulinių elementų. Atlikus eilutės daugybą iš stulpelio, kintamsis atstatomas į pradinę padėtį, AR_dau – rezultatų masyvas, AR, AR11 – nenulinių elementų masyvai.

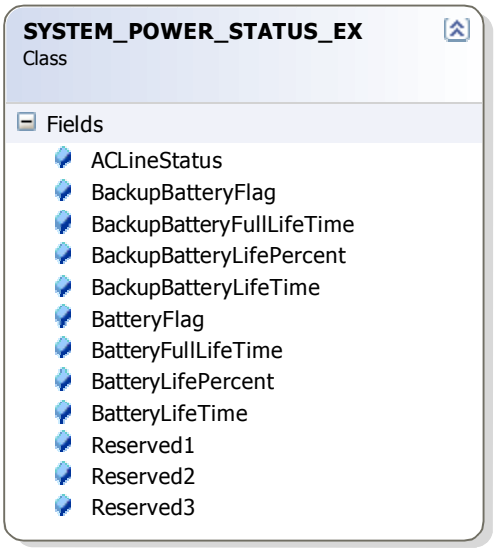
Vieno masyvo indeksai lyginami su kito masyvo indeksais ir jeigu jie sutampa daugybą atliekama, jeigu ne, pirmo masyvo elementas paliekamas tas pats, o prabėgama pro kito masyvo likusius stulpelio elementus ir vėl tikrinama ar indeksai sutampa. Jei sutampa – sudauginame, o jei ne įrašome nulį.

Is algoritmo matome, kad svarbi yra i masyva surasytu elemetnu tvarka.

3.3.4 Klasių diagramos

Apsirašome stebimus parametrus ir juos surašysime į DLL tipo failą, kurį vėliau įtrauksime i pagrindinį kuriamą projektą.

Visi galimi stebėti baterijos parametrai surašyti standartinėse klasėse, kurias apsirašome. Naudojamos dvi pagrindinės klasės: System_power_status_ex ir System_power_status_ex2. Pirmoje stebimų parametrų mažiau, kitoje daugiau. Tačiau projekte naudojamos abi klasės. Klasėse apsirašomi kintamieji, kurie parodys atitinkamus baterijos parametrus.



8 pav. Baterijos parametrų klasės diagrama

Pavaizduotoje klasėje System_Power_Status_EX parametras ACLineStatus rodo ar prijungtas išorinis šaltinis. Parametro galimos tokios reiksmės:

7 lentelė. ACLineStatus parametro reikšmės

Reikšmė	Apibūdinimas
0	Išjungtas pakrovėjas
1	Įjungtas pakrovėjas

255	Nežinomi duomenys

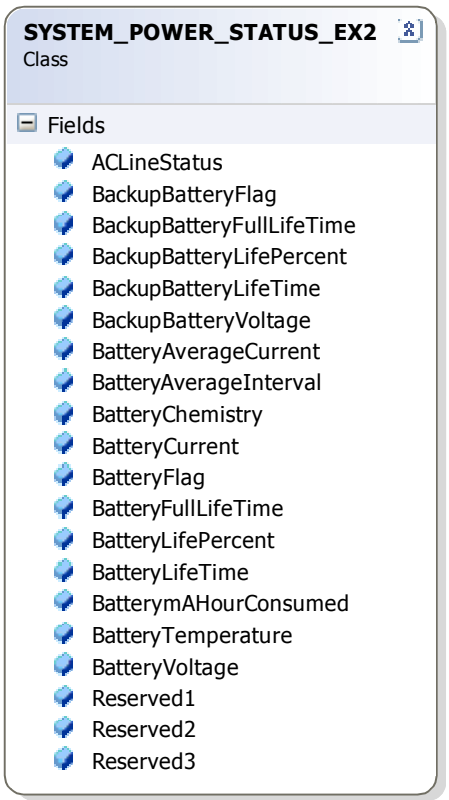
BatteryFlag parodo baterijos įsikrovimo lygį. Galimos tokios reiksmės:

8 lentelė. BatteryFlag parametro reikšmės

Reikšmė	Apibūdinimas
1	Aukštas (High)
2	Žemas (Low)
4	Kritinis (Critical)
8	Kraunama baterija
128	Nėra baterijos
255	Nežinoma būsena

Kitas parametras BatteryLifePercent – parodo, kiek procentų liko pilnai pakrautos baterijos. Reikšmės kinta nuo 0 iki 100.

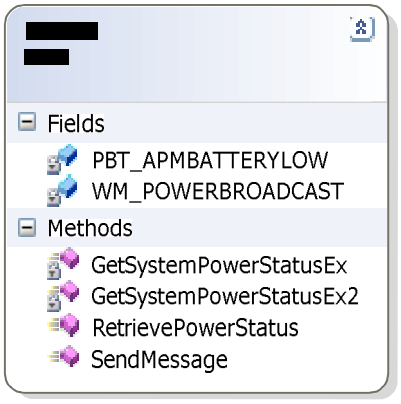
Kiti standartiniai šios klasės parametrai nenaudojami. Kitoje klasėje System_Power_status_ex2 n audojami tie patys parametrai, kaip ir prieš tai aprašytoje klasėje ir pridėta naujų. Tai pirmos klasės papildymas.



9 pav. Baterijos parametrų klasės diagrama

Pagrindinis ir mūsų projekte svarbiausias šios klasės parametras yra BatteryVoltage, kuris parodo baterijos voltažą mili voltais. Reikšmė gali kisti intervale nuo 0 iki 65535. BatteryChemistry parodo, kokio tipo yra baterija. Galimos tokios reikšmės: Alkaline, Nicd, Nimh, Lion, Lipoly.

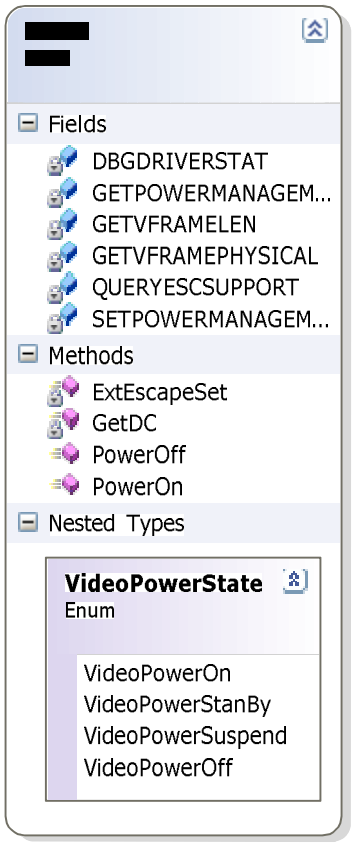
Prieš tai aprašytų klasių parametrus panaudosime pagrindinėje klasėje Power, kurioje parašysime metodus, kurių pagalba bus nuimami duomenys.



10 pav. Power klasės diagrama

Metodas RetrievePowerStatus gauna įvairius norimus stebėti baterijos parametrus. Čia juos surašome ir stebime. Metodais GetSystemPowerStatusEx ir Ex2 kreipiamės į išorinę biblioteką, kurioje ir yra anksčiau aprašytos dvi klasės.

Dauginimo metu kišeninio kompiuterio ekranas yra išjungtas, tam, kad gautume kuo tikslesnius parametrus. Ekraną išjungimas aprašytas klasėje Video. Metodas PowerOff išjungia monitorių, o PowerOn įjungia.



11 pav. Monitoriaus išjungimo klasės diagrama

VideoPowerState – parodo monitoriaus būseną. Galimos tokios reikšmės: monitorius įjungtas, išjungtas, ekranas yra miegojimo rėžime.

3.4 Testavimo modelis bei duomenys, kontrolinis pavyzdys

Sukūrę programą turime parengti pradinį testavimo modelį. Testuodami ir analizuodami gautus pradinius rezultatus padarysime atitinkamas išvadas ir pakoreguosime testavimo modelį sau reikalinga linkme. Kadangi dabar negalime nieko konkretaus pasakyti, kiek laiko užtruks, sakykim, baterijos iškrovimas nuo 100% iki 30%.

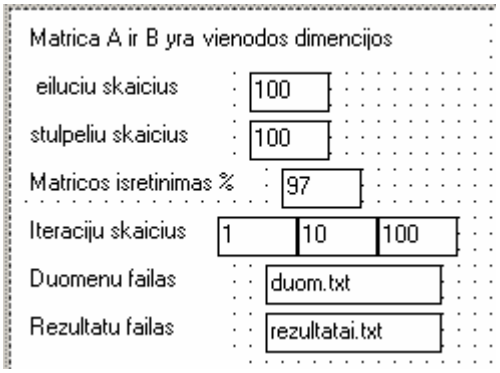
Pradinis testavimo modelis yra aiškus. Į programą suvedame duomenis ir žiūrime, kiek laiko užtrunka visa operacija. Gautus rezultatus analizuojame ir suvedame kitus duomenis. Toki procesą kartojame keletą kartų, kol gausime kažkokį vaizdą.

Į programą reikia suvesti tokius duomenis:

- Matricos eilučių skaičius;
- Matricos stulpelių skaičius;
- Matricos išretinimas procentais;
- Dauginimo iteracijos (pradinis, žingsnis, galutinis iteracijų skaičius);
- Duomenų failo pavadinimas;
- Rezultatų failo pavadinimas;

Dauginamos dvi matricos bus vienodų dimensijų, todėl užtenka įvesti vienos matricos dimensiją. Įvedame stulpelių ir eilučių skaičių (pvz. 100 eilučių ir 100 stulpelių). Toliau vedame matricos išretinimą procentais. Kuo didesnis išretinimo procentas, tuo daugiau matricoje yra nulinių elementų. Sakykime įvedame matricos išretinimą 97%. Tokiu atveju turėsime, matricą, kurioje bus 3% nenulinių elementų ir 97% nulinių elementų. Dauginimo metu naudosime aukštos eilės išretinimo procentą (91%, 95%, 97%, t.t). Toliau vedame iteracijų skaičius. Pirma vedame pradinį iteracijų skaičių, kuris parodo nuo kelintos dauginimo operacijos išsaugoti baterijos duomenis. Jeigu įvedame 5, tai reiškia, kad įvykdžius penkias dauginimo iteracijas programa išsaugos baterijos duomenis. Kitas skaičius yra iteracijos žingsnis, kuris parodo, kas kiek įvykdytų dauginimo iteracijų reikia išsaugoti baterijos duomenis. Tarkime, kad įvedame skaičių – 10. Tai reiškia, kad dauginant, kas dešimt iteracijų baterijos duomenys bus išsaugoti. Ir paskutinis įvedamas iteracijų skaičius – galutinis dauginimo skaičius. Kuris parodo, kiek dauginimo iteracijų reikia sudauginti išviso. Įvedame – 100. Išviso bus įvykdyta šimtas dauginimo iteracijų.

Toliau belieka įrašyti duomenų failo pavadinimą, kuriame bus išsaugota pradinė matrica ir rezultatų failo pavadinimą. Šiame faile bus išsaugojami dauginimo iteracijos išsaugoti duomenis, kuriuos mes analizuosime ir padarysime atitinkamas išvadas.



12 pav. Sukurtos programos langas

Paveiksle pavaizduotas kontrolinis programos duomenų suvedimo pavyzdys. Tokiu duomenų suvedimo atveju, programa sugeneruos dvi matricas, kurios turės 100 eilučių ir 100 stulpelių. Matricų išretinimas bus 97%, tai reiškia, kad matrica turės 97% nulinių elementų. Iš viso bus atliekama 100 dauginimo iteracijų, tai parodo paskutinis iteracijų skaičius. Įvykdžius viena dauginimo iteraciją, bus įrašyti gauti baterijos duomenys ir taip bus kartojama kas 10 dauginimo iteracijų, tol kol bus įvykdytos visos šimtas dauginimo iteracijų. Duomenų faile „duom.txt“ bus išsaugota viena matrica, kadangi abi matricos yra vienodos. Faile „rezultatai.txt“ bus išsaugoti visi dauginimo metu gauti rezultatai. Baterijos duomenys (voltažas, gyvavimo trukmė procentais), kokios matricos buvo sudaugintos, laikas, kiek truko visas dauginimo procesas, įvykdytų iteracijų skaičius, išretinimo procentas.

Įvedę tokius duomenis belieka pasirinkti dauginimo metodą, kurio dėka bus sudauginamos matricos. Paprasčiausiai pasirenkame kokį mygtuką norime paspausti: jei norime, kad matricos būtų sudaugintos naudojant paprasta dauginimo metodą, spausti reikia mygtuką pavadinimu – „Daug_paprasta“, o jei norime, kad matricos būtų sudaugintos naudojant išretintų matricų dauginimo metodą, skaičiuojant stulpelius, spaudžiame mygtuką – „Daug_isretinta“.

4. Eksperimentinis sistemos tyrimas

Sukūrę programą atliksime eilę dauginimo operacijų tam, kad galėtume pasiekti užsibrėžtus darbo tikslus. Kadangi iš pradžių mes nežinome nieko apie tai kiek laiko užtruks daugyba, kokias matricas dauginti ir po kiek kartų reikia, pradžiai bus atliekami bandomieji dauginimai. Po bandomųjų dauginimų galėsime padaryti tam tikras išvadas ir nukreipti eksperimentą sau norima linkme.

4.1 Eksperimento rezultatai

Iš pradžių atlikinėsime paprastą matricų daugybą. Tam pilnai pakrauname delninuką. Į programą įvedame sugalvotus duomenis: 100 eilučių, 100 stulpelių, matricos išretinimas

95%. Pradžiai atliksime vieną dauginimo iteraciją ir pažiūrėsime, kokius gausime rezultatus.

Įvykdžius vieną dauginimo iteraciją programa pateikia sekančius rezultatus, kurie pavaizduoti lentelėje.

9 lentelė. Paprasto metodo dauginimo rezultatai

Stebimi parametrai	**Reikšmės**
Matrica	100 x 100
Matricos išretinimo procentas	95%
Baterijos įkrovimo procentas	100%
Baterijos voltažas	4.108V
Įvykdytos iteracijos	1
Dauginimo laikas	4s

Iš gautų rezultatų matome, kad pilnai pakrautos baterijos voltų skaičius – 4.108V. Mažėjant baterijos įkrovimo procentui, mažės ir voltų skaičius. Naudodami šiuos du parametrus, palyginsime abu dauginimo metodus ir pamatysime, kuris yra efektyvesnis.

Visas dauginimo procesas užtruko tik 4 sekundes. Matydami pradinius rezultatus, didinsime iteracijų skaičių tam, kad baterija išsikrautų bent keliasdešimt procentų. Bandymų metu keisime ne tik iteracijų skaičių, bet ir matricos dimensiją bei išretinimo laipsnį procentais.

Naudodami dauginimo iteracijas galime dauginti mažos eilės matricas ilgą laiko tarpą. Tokiu atveju bus užimama mažiau atminties.

Pasirenkame matricą, kuri bus sudaryta iš 15 eilučių ir 15 stulpelių. Matricos išretinimas 95%. Daugybos metu bus atliekama 100000 dauginimo iteracijų. Gauti rezultatai pavaizduoti lentelėje.

10 lentelė. Paprasto metodo dauginimo rezultatai

		Max			vid			min		
		100%		90%	90%		80%	80%		70%
15 x 15	pr: 2000	4.094 V	4.050 V	3.938 V	3.923 V	-	-	-	3.802 V	3.796 V
	ž: 3000									
	pab:100000	100%	99%	91%	88%				79%	78%
Įvykdytos iteracijos		2000	29000	56000	71000	-	-	-	89000	10000

11 lentelė. Paprasto metodo dauginimo rezultatai

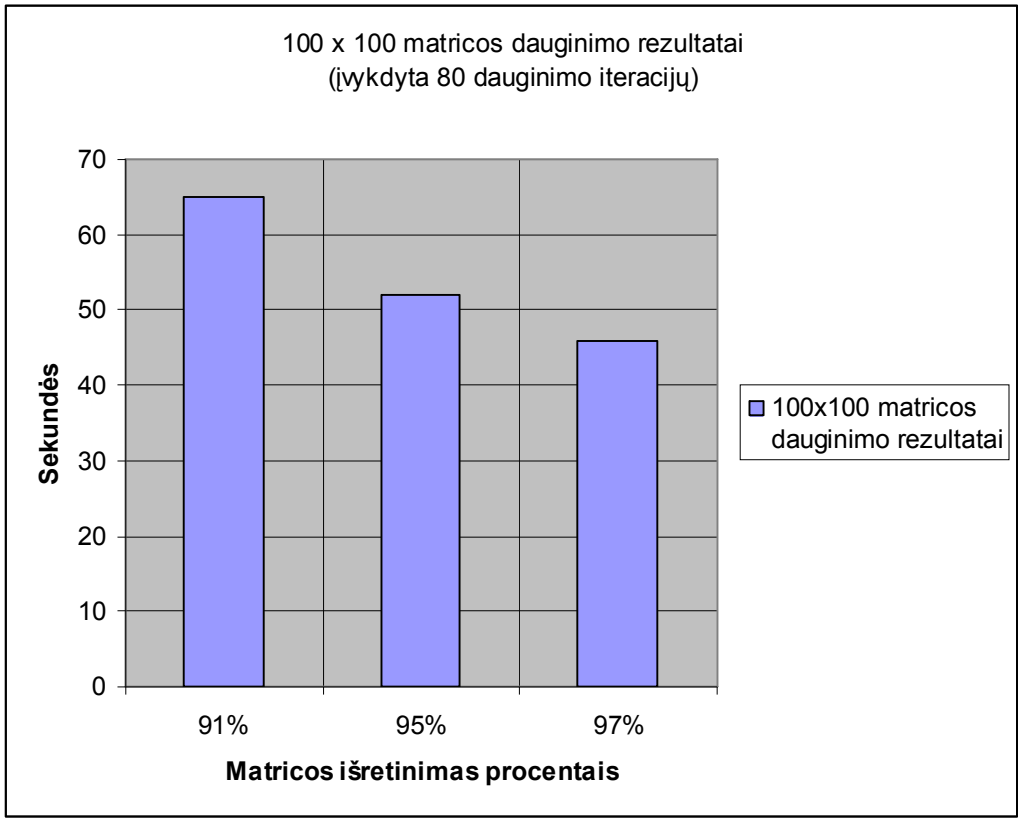
		Max			vid			min		
		90%		80%	80%		70%	70%		55%
30 x 30	pr: 100	3.874 V	3.845 V	3.835 V	3.830 V	-	-	3.723 V	3.694 V	3.708 V
	ž: 300									
	pab:10000	85%	84%	80%	75%			64%	62%	56%
Įvykdytos iteracijos		Įvykdytos Iteracijos 200	700	1000	3700			5200	6400	10000

Baterijos duomenys išsaugoti kas 3000 iteracijų, pradedant nuo 2000 (lentelėje pavaizduoti sutraukti duomenys, kadangi iš viso gauta 50 tarpinių rezultatų). Iš gautų rezultatų matome, kad baterija išsikrovė nuo 100% iki 78%, o voltažas nuo 4.094 V nukrito iki 3.769 V. Analizuodami visus gautus rezultatus pastebėjome, kad baterijos įkrovos procentų skaičius mažėja nepastoviai. Įkrovos skaičius nuo 88% pasikeitė į 79%. Lentelėje trukstami duomenys pažymėti minusiuko ženklu. Tokiu atveju visą nusėdimo intervalą suskaldome į tris dalis. Pirmas intervalas įkrovimo maksimumas, kintamos reikšmės intervale nuo 100 % iki 90 %. Kitas intervalas – vidutinis, reikšmės nuo 90 % iki 80 %. Ir paskutinis intervalas gaunsi minimumas – nuo 80 % iki 70 %. Nuoseklių reikšmių negavome antrame intervale, tas pats ir dauginant kitą matricą 30 x 30 ir vykdant 10000 iteracijų. Todėl atlikinėdami sekančius bandymus orientuosimės į šitus intervalus.

Atlikę pradinį eksperimentą ir pamatę programos bei delinuko baterijos elgseną, nustatome, kokios maksimalios dimensijos matricą dar galima sudauginti delninuku. Eksperimento būdu nustatyta, kad maksimalios dimensijos matricą, kurią dar galima sudauginti vieną iš kitos yra 500 x 500 (500 eilučių ir 500 stulpelių). Didesnės dimensijos matricas sudauginti negalime, kadangi trūksta atminties. Nustatyta sąlyga galioja tik tai mūsų naudojamam delninukui. Dauginamų matricų maksimali dimensija priklauso nuo delninuko konfigūracijos. Kuo naujesnį delninuką turėsime, tuo didesnės dimensijos matricas galėsime sudauginti, kadangi jame bus daugiau operatyvinės atminties. Eksperimentui naudojamas delninukas nėra naujas, todėl jis turi mažai operatyvinės atminties, todėl norėdami, kad dauginimo procesas greitai nepasibaigtų įvedėme dauginimo iteracijas. Dabar galime dauginti mažesnės dimensijos matricas ilgą laiko tarpą.

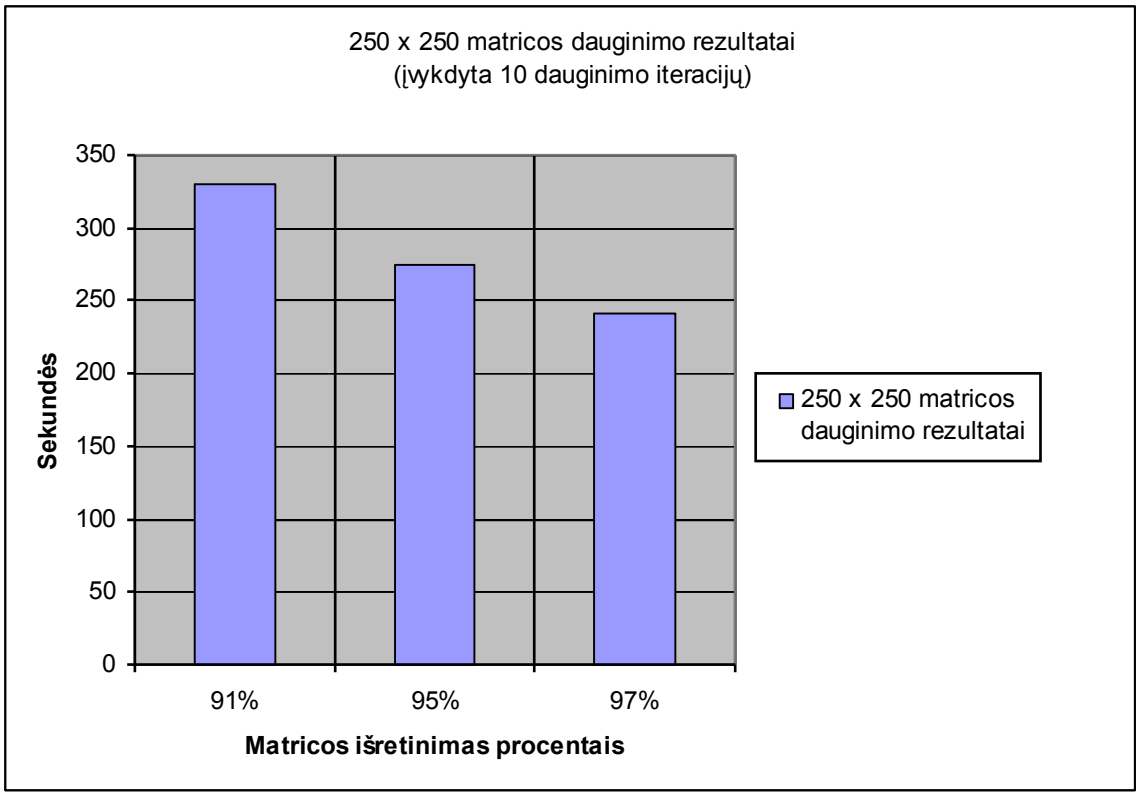
Atsižvelgę į iki šiol gautus rezultatus tolimesnę eksperimento eigą nustatome taip. Bus dauginamos tokios matricos: 100 x 100, 250 x 250, 400 x 400. Matricų išretinimai parenkami: 91 %, 95 %, 97 %. Daugyba bus atliekama paprastu metodu ir stulpelių saugojimo metodu, gauti rezultatai sulyginami tarpusavyje.

Iš pradžių vykdysime stulpelių saugojimo metodo daugybą, naudojant 100 x 100 matricą. Ir pasižiūrėsime ar dauginimo laikas tikrai priklauso nuo išretinimo procento.



13 pav. Stulpelių saugojimo metodo dauginimo laikų diagrama

Pavaizduotoje 13 diagramoje matome išretintos matricos, kuri buvo išsaugota naudojant stulpelių saugojimo metodą, dauginimo rezultatus. Išviso buvo atlikta trys daugybos su skirtingais išretinimo procentais (91%, 95%, 97%). Matrica šimtos eilės. Dauginimo metu įvykdyta 80 iteracijų. Kai matricos išretinimas yra 91%, visas dauginimo procesas užtruko 65 sekundes. Esant 95% išretinimui, daugyba užtruko 52 sekundes ir 97% - 46 sekundes. Matome, kad kuo didesnis išretinimo procentas, tuo greičiau įvykdoma daugyba. Gautų rezultatų patvirtinimui atliksime kitus eksperimentus, panaudojant skirtingos dimensijos matricas.

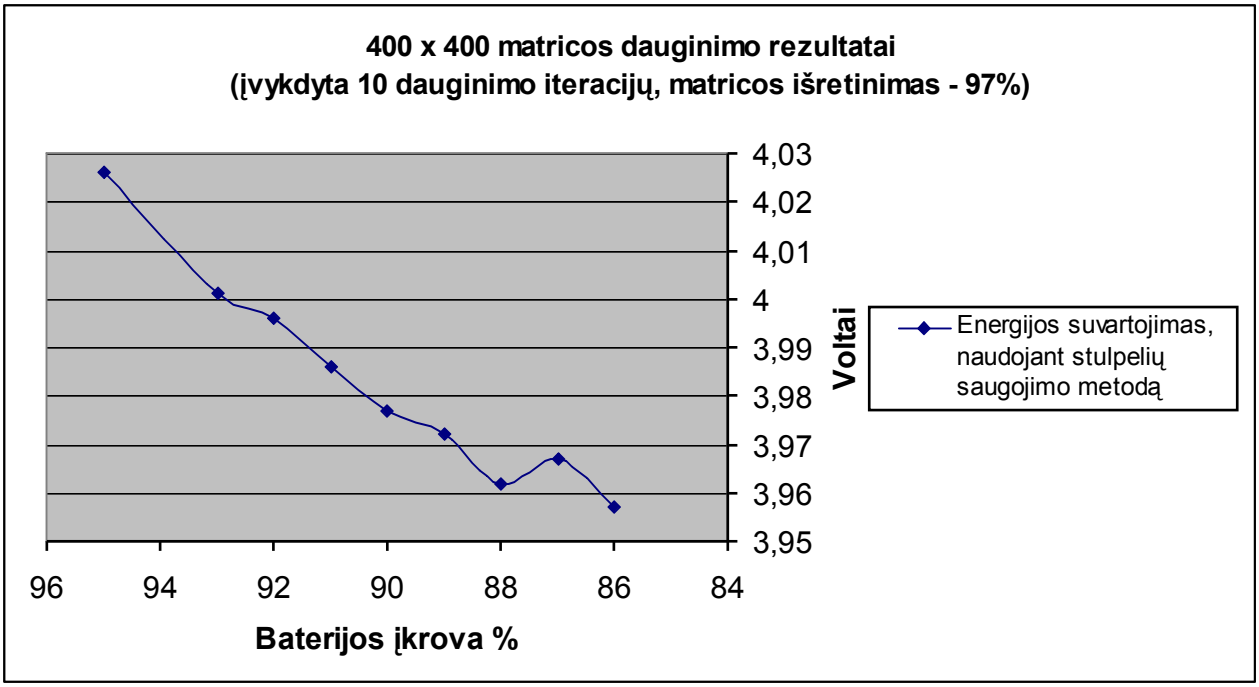


14 pav. Stulpelių saugojimo metodo dauginimo laikų diagrama

Naudojant stulpelių saugojimo metodą, sudaugintos dvi matricos, kurios turi 250 eilučių ir tiek pat stulpelių. Dauginimo metu įvykdyta 10 iteracijų su skirtingais išretinimo procentais. Esant 91% matricos išretinimui visas dauginimo procesas užtruko 330 sekundžių, 95% - 274 sekundes ir 97% - 241 sekundę. Kaip matome esant didžiausiam išretinimui, dauginimo laikas yra trumpiausias. Tai įrodo, kad išretintų matricų dauginimo proceso trukmė, naudojant stulpelių saugojimo metodą, priklauso nuo išretinimo procento. Kuo išretinimas didesnis, tuo greičiau įvyks daugyba, bus sunaudojama mažiau atminties, kadangi saugomi tiktai nenuliniai elementai masyvuose.

Dabar palyginsime ar stulpeliu saugojimo metodas yra pranašesnis lyginat su paprastu saugojimo metodu.

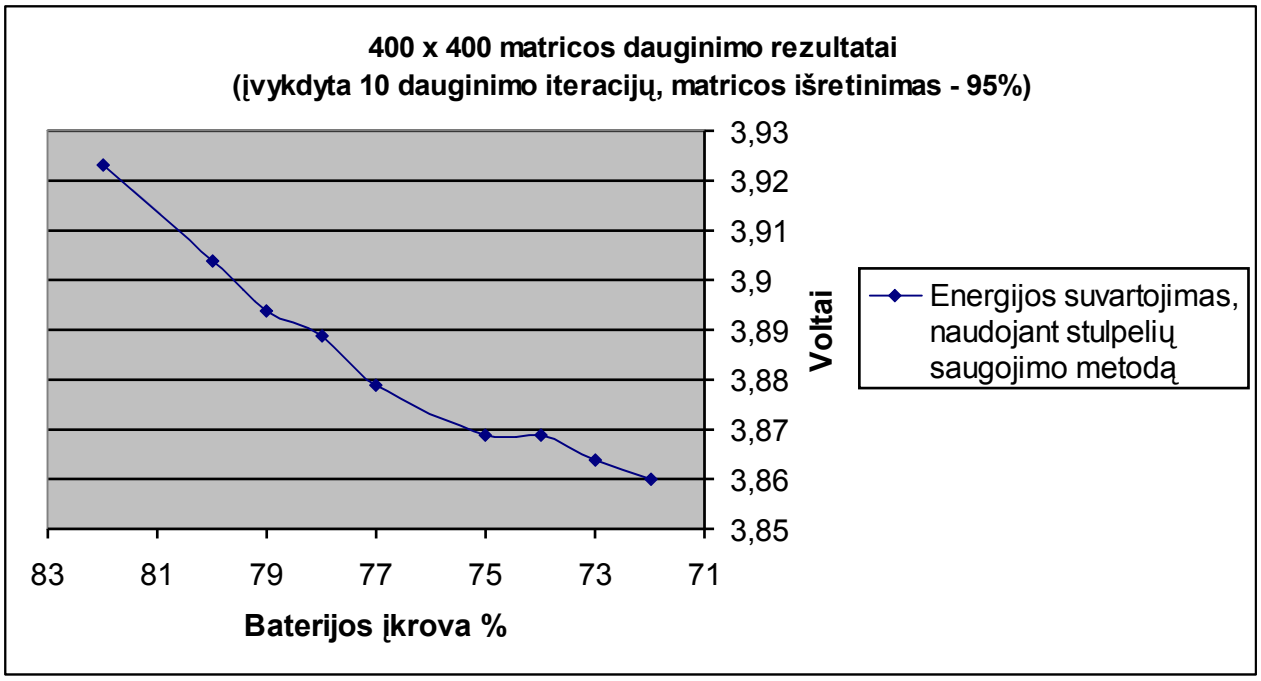
Dauginsime matricą, kuri bus sudaryta iš 400 eilučių ir 400 stulpelių. Išretinimai bus: 97%, 95%, 91%. Atliekame paprastą ir stulpelių saugojimo metodo daugybą. Daugybą pradedame, kai delninukas pakrautas iki 95%. Vykdome išviso 10 dauginimo iteracijų.



15 pav. Energijos suvartojimo grafikas, naudojant stulpelių saugojimo metodą

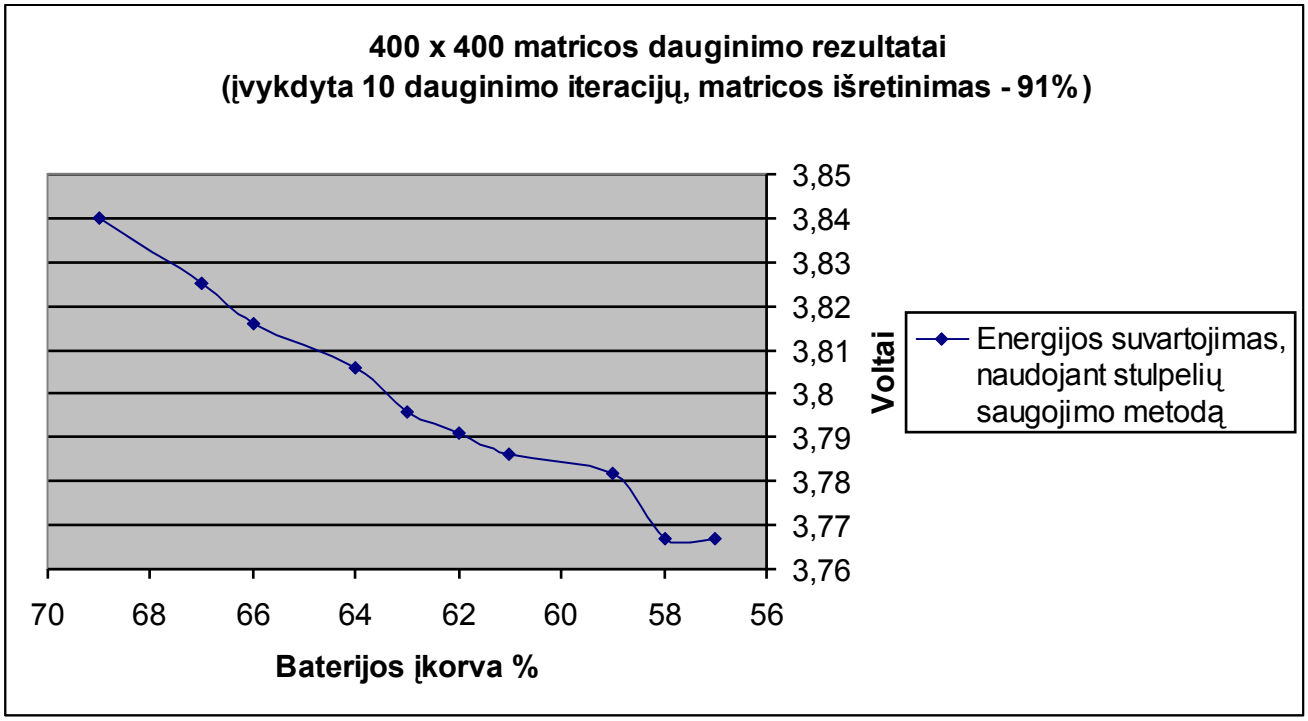
15 paveiksle pavaizduotas energijos suvartojimas, naudojant stulpelių saugojimo metodą. Kaip matome visos operacijos metu delninuko baterija nuo 95 % išsikrovė iki 86%, o voltažas nuo 4.026 V krito iki 3,957V. Visa operacija užtruko 27 minutes ir 15 sekundžių. Apytikslis vienos iteracijos dauginimo laikas – 2 minutes ir 44 sekundės. Baterija išsikrovė 9%.

Baterijos nekrauname ir iškart paleidžiame tą pačią daugybą, tik pakeičiame išretinimo procentą į 95%. Gauname sekanti energijos suvartojimo grafiką, kuris pavaizduotas 16 paveiksle.



16 pav. Energijos suvartojimo grafikas, naudojant stulpelių saugojimo metodą

Daugybą pradėta, kai baterijos įkrova yra 82%, baigėsi esant 72%. Voltažas nuo 3,923 V nukrito iki 3,860 V. Visa dauginimo operacija užtruko 28 minutes ir 20 sekundžių. Apytikslis vienos iteracijos dauginimo laikas – 2 minutės ir 50 sekundžių. Baterija išsikrovė 10%. Energijos suvartojimo grafikas labai panašus į prieš tai buvusį grafiką. Pradžioje yra staigesnis voltažo kritimas, paskui pokytis susinormalizuoja ir kinta lėčiau. Operacijos pabaigoje matomas trumpas pokyčio sumažėjimas, o 97% išretinimo energijos suvartojimo grafike ir pokyčio padidėjimas.



17 pav. Energijos suvartojimo grafikas, naudojant stulpelių saugojimo metodą

17 paveiksle matome tos pačios matricos su 91% išretinimu, energijos suvartojimo pokytį. Baterija po pirmų dviejų bandymų nebuvo pakrauta ir iškarto paleistas šitas ekesperimentas. Baterijos iškrovimas prasidėjo nuo 69% ir baigėsi esant 57%. Baterijos voltų skaičius pasikeitė nuo 3,840 V iki 3,767 V. Visa dauginimo operacija užtruko 35 minutes, o apytiksliai viena iteracija įvykdyta per 3 min ir 30 sekundžių.

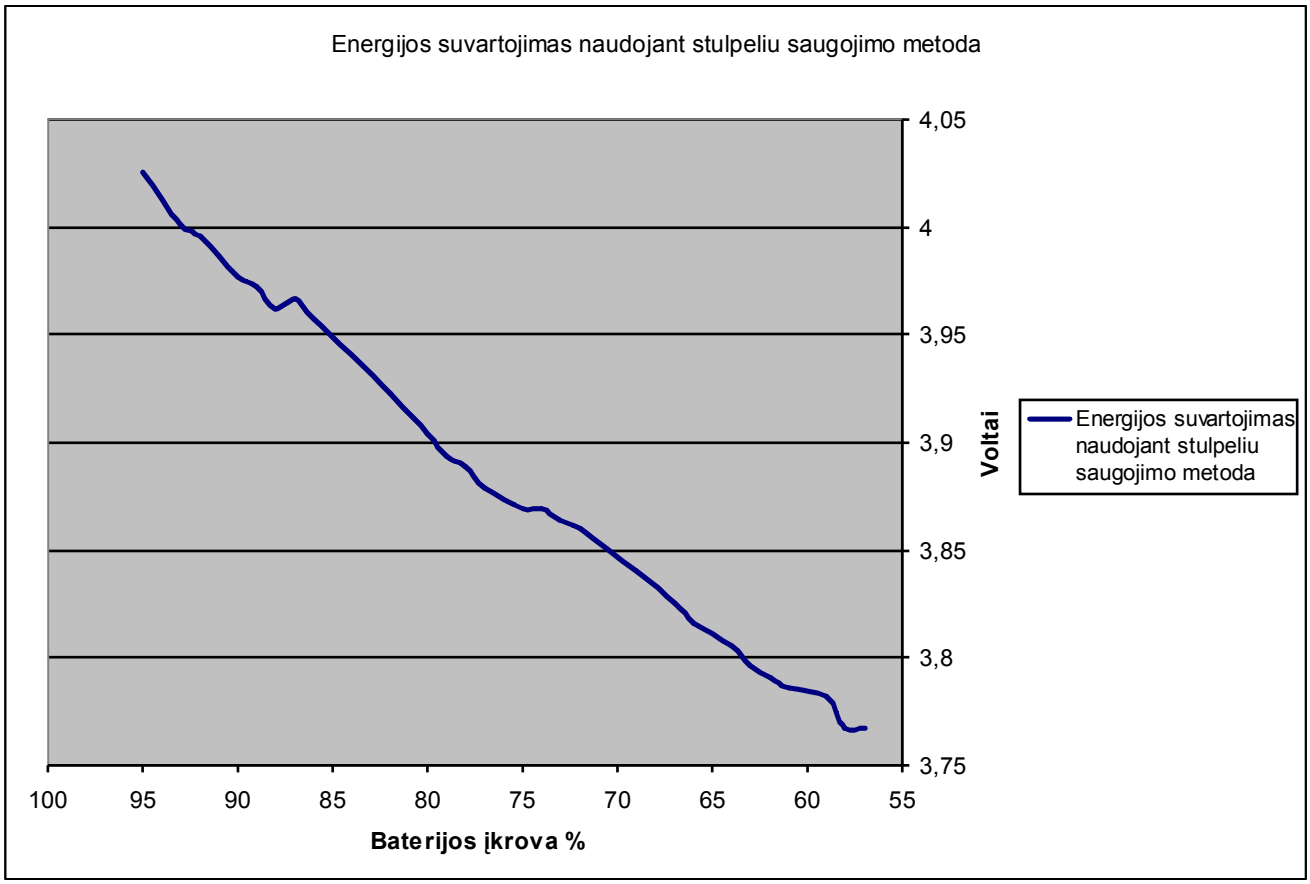
Apibendrinant paskutinių trijų eksperimentų rezultatus galime sudaryti 12 lentelę.

12 lentelė. Stulpelių saugojimo metodo gautų rezultatų palyginimas

Parametrai	Matricos išretinimai		
	91%	95%	97%
Bendras operacijos laikas (įvykdyta 10 dauginimo iteracijų)	35 min	28 min 20s	27 min 15s
Vienos iteracijos vykdymo laikas	3 min 30s	2 min 50s	2 min 44s
Operacijos metu baterija išsikrovė	12%	10%	9%
Voltažo pokytis	0,073 V	0,063 V	0,069 V

Vėl matome, kuo didesnis matricos išretinimas, tuo trumpiau trunka visa dauginimo operacija, iškraunama mažiau baterijos.

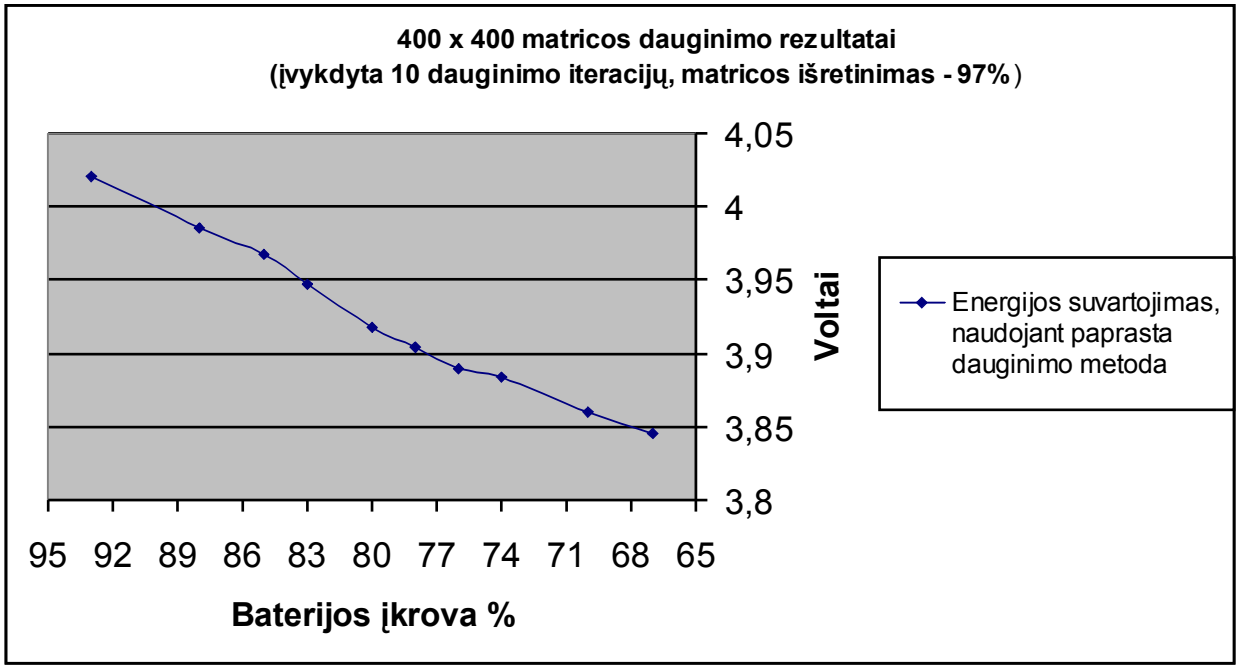
Visus tris energijos suvartojimo grafikus sudedame į vieną grafiką, sujungiame taškus ir gauname bendrą energijos suvartojimą, kuris pateiktas 18 paveiksle.



18 pav. Energijos suvartojimo grafikas, naudojant stulpelių saugojimo metodą

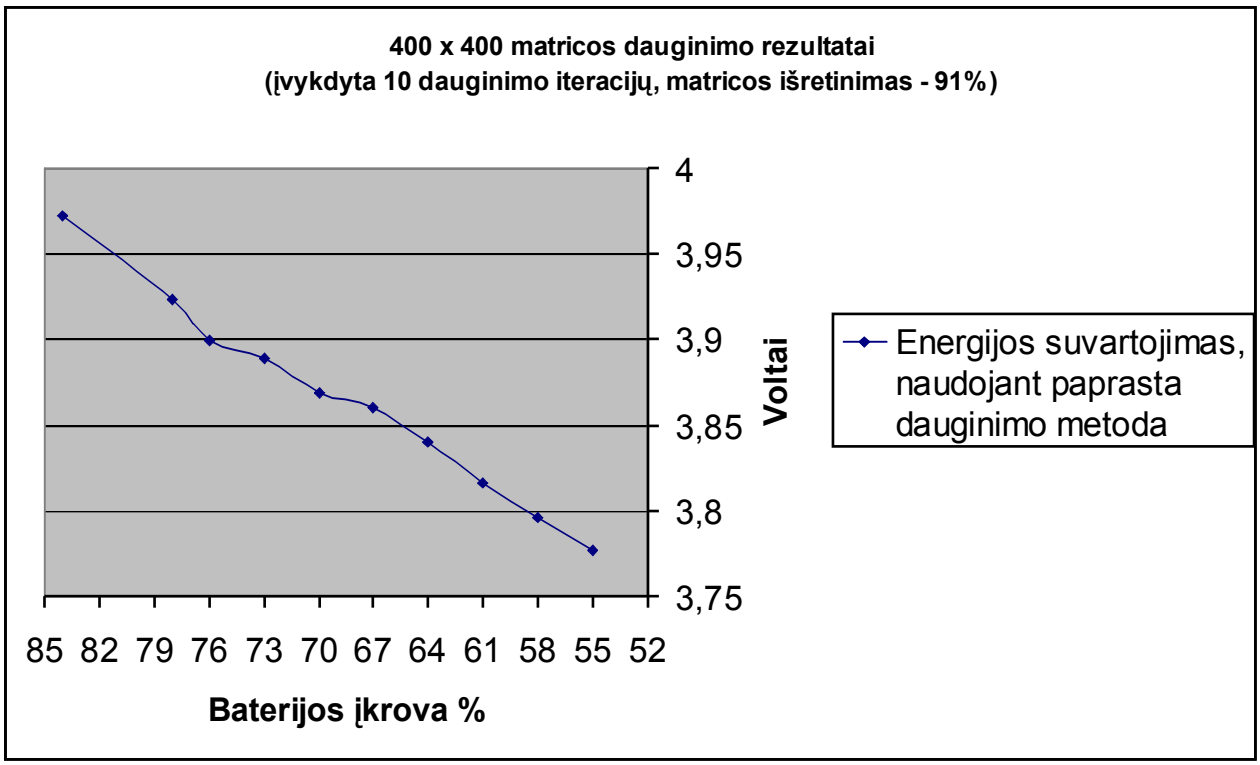
Apytiksliai gavome, kad visų trijų operacijų metu baterija išsikrovė 31%, o voltažas nukrito – 0,205 V.

Toki patį eksperimentą atliksime toliau, tik vietoje stulpelių saugojimo metodo daugybos naudosime paprastą matricų daugybą. Naudojame tą pačią sugeneruotą matricą, kad eksperimento rezultatai būtų kuo tikslesni. Pirmoji dauginimo operaciją, kurią atliekame yra esant 97% matricos išretinimui. Eksperimento rezultatai pavaizduoti sekančiame grafike.



19 pav. Energijos suvartojimo grafikas, naudojant paprastą daugybos metodą

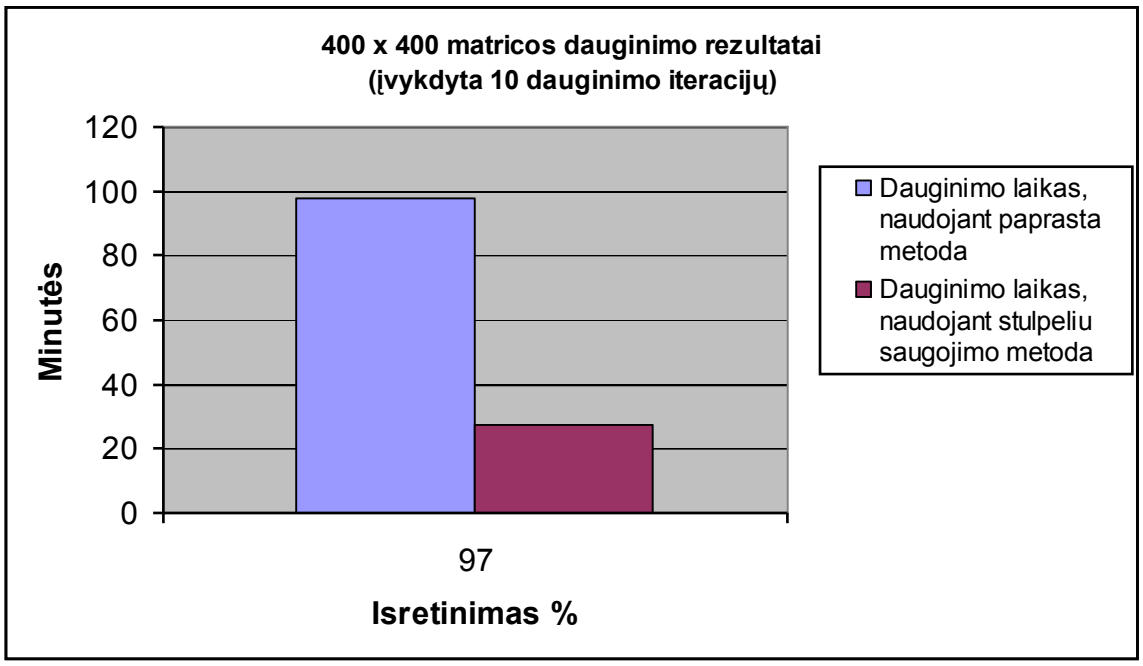
Pavaizduotas energijos suvartojimo grafikas 19 paveiksle, naudojant paprastą dauginimo metodą. Per visą dauginimo laiką, baterija išsikrovė nuo 93% iki 67%. Voltažas nukrito nuo 4,021 V iki 3,845 V. Visa operacija užtruko 97 minutes ir 45 sekundes. Apytiksliai vienos iteracijos įvykdymo laikas – 9 minutės ir 46 sekundės. Iš grafiko matosi, kad baterija išsikrauna tolydžiai, nedarydama staigesnių šuolių. Kadangi baterija išsikrovė iki 67%, kito eksperimento metu nenorėdami rizikuoti per daug iškrauti baterija, pakrauname iki 85%. Atliekame kitą eksperimentą su ta pačia matrica, tik pakeičiame išretinimo procentą.



20 pav. Energijos suvartojimo grafikas, naudojant paprastą daugybos metodą

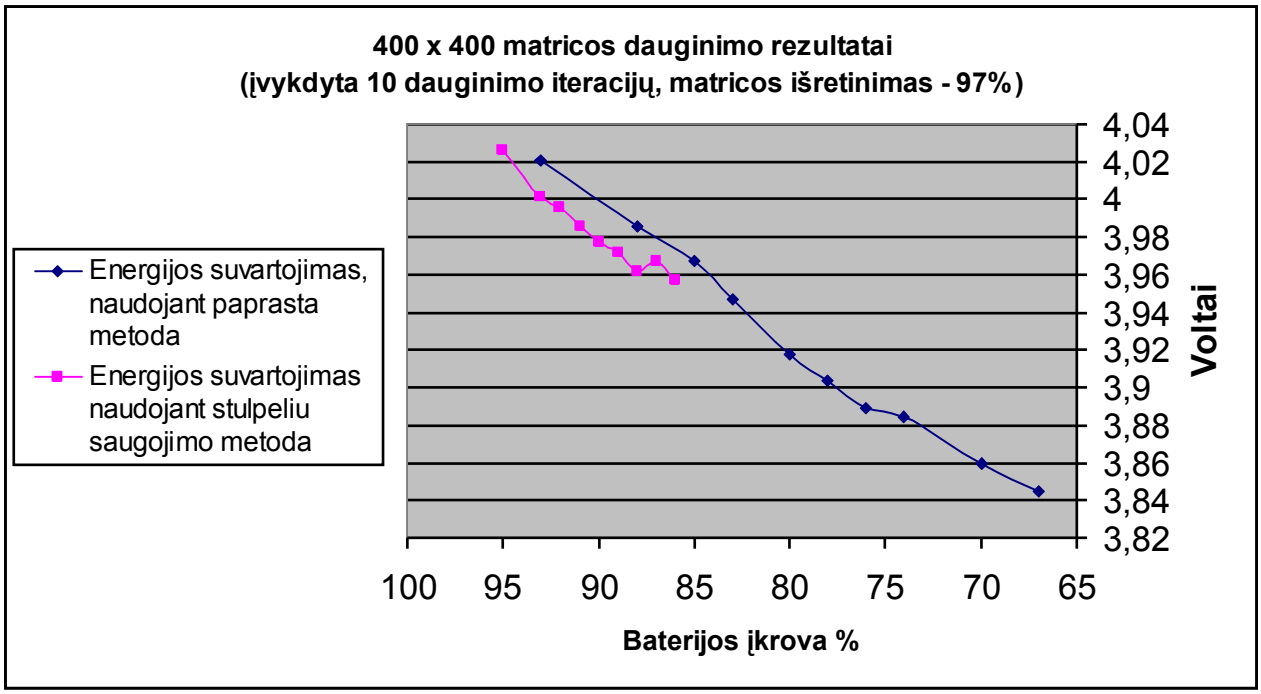
Grafike pavaizduotas energijos suvartojimas (20 paveikslas), matricos išretinimui esat – 91%. Po eksperimento įvykdymo baterija išsikrovė nuo 84% iki 55%. Voltažas nukrito nuo 3,972 V iki 3,777V. Visa dauginimo operacija užtruko 107 minutės ir 12 sekundžių. Apytikslis vienos iteracijos vykdymo laikas – 10 minučių ir 35 sekundės. Kaip jau galime pastebėti laikas ir baterijos iškrovimas šiuo metodu užtrunka ilgiau negu stulpelių saugojimo metodo. Toliau palyginsime dviejų metodų: paprasto ir stulpelių saugojimo, gautus daugybos rezultatus.

Lyginame tarpusavyje du metodus: paprastą ir stulpelių saugojimo metodą. Matricų išretinimas 97%.



21 pav. Paprasto metodo ir stulpelių saugojimo metodo dauginimo laikų palyginimo diagrama

21 paveiksle palyginti paprasto metodo ir stulpelių saugojimo metodo daubybos laiko rezultatai. Kaip matome atliekant 400 x 400 matricos su 97% išretinimu daugybą paprastu metodu, visa operacija trunka 97 minutes ir 45 sekundes, o tą pačią matricą dauginant panaudojus stulpelių saugojimo metodą – 27 minutes ir 15 sekundžių. Laikų skirtumas – 70 minučių ir 30 sekundžių. Skirtumas akivaizdus. Naudodami stulpelių saugojimo metodą vien tik šitai operacijai sutaupome daugiau negu valandą laiko, o tai žiūrint delninulų mastu yra daug. Svarbiausia, kad dauginimo rezultatai tokie patys, skyrėsi tiktai pačios matricos saugojimo metodas. Tai ir įrodo išretintų matricų stulpelių saugojimo metodo algoritmo efektyvumą. Toliau palyginame, kaip keitėsi baterijos voltų skaičius.



22 pav. Energijos suvartojimo grafikai naudojant paprastą ir stulpelių saugojimo metodą

Matome energijos suvartojimo grafikus (22 pav.), paprasto metodo ir stulpelių saugojimo metodo. Dauginant stulpelių saugojimo metodu, baterija nuo 95% išsikrovė iki 86%, o operaciją atliekant paprastu metodu – nuo 93% iki 67 %. Atitinkamai baterijos voltų skaičius stulpelių metodu nuo 4.026 V nukrito iki 3,957 V, o paprastu metodu – nuo 4,021 V iki 3,845 V. Baterijos iškrovimo skirtumas, lyginant abu metodus – 17%, voltažo pokyčio skirtumas – 0,107 V.

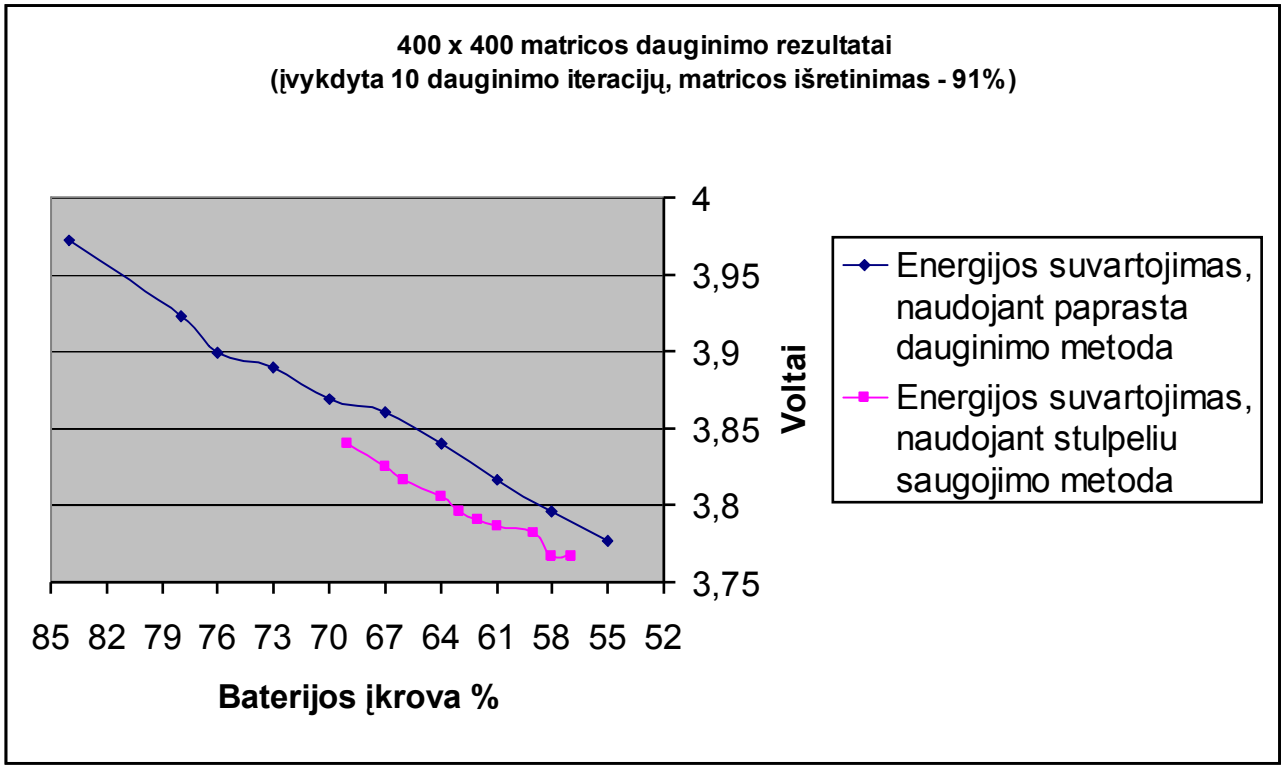
Delninukas vykdydamas tą pačią operaciją paprastu metodu išsikrovė 26%, o stulpelių saugojimo metodu 9%. Gauname, kad stulpelių saugojimo metodas yra 2.89 karto efektyvesnis negu paprastas dauginimo metodas, lyginant baterijos išsikrovimo lygį. Palyginus voltų pokytį, stulpelių saugojimo metodas 2,55 karto efektyvesnis negu paprastas. Ir palyginus operacijos vykdymo laiką – 3,587 karto efektyvesnis stulpelių saugojimo metodas. Vidutiniškai, dauginant matricą, kuri susideda iš 400 eilučių ir 400 stulpelių, matricos išretinimas 97%, stulpelių saugojimo metodas yra 3 kartus efektyvesnis negu paprastas dauginimo metodas.

Apibendrinant rezultatus galime sudaryti 13 lentelę.

13 lentelė. Abiejų dauginimo metodų rezultatų palyginimas

Parametrai	Matrica 400 x 400, 97% išretinimas, įvykdyta 10 dauginimo iteracijų	
	Paprastas dauginimo metodas	Stulpelių saugojimo metodas
Bendras operacijos laikas	97 min 45 s	27 min15 s
Vienos iteracijos vykdymo laikas	9 min 46 s	2 min 44s
Operacijos metu baterija išsikrovė	26%	9%
Voltažo pokytis	0,176 V	0,069 V

Palyginame, tos pačios matricos daugybą išretinimui esant 91%. Grafike matome du grafikus. Vienas gautas operaciją atlikus paprastu metodu, kitas stulpelių saugojimo metodu.



23 pav. Paprasto metodo ir stulpelių saugojimo metodo energijos suvartojimo grafikai

Čia vėl matosi akivaizdus stulpelių saugojimo metodo pranašumas (23 pav). Tos pačios operacijos metu, vienu metodu baterija išsikrovė 12% (stulpelių saugojimo), kitu – 29%. Iš šito ir iš kitų energijos suvartojimo grafikų galime pastebėti tendenciją, kad skirtingų eksperimentų metu, nesutampa baterijos įkrovimo procentų skaičius su voltažo skaičiumi. Sitame grafike, daugybą atliekant paprastu metodu, esant baterijos įkrovimui 61%, voltažas – 3,816 V, o dauginant stulpelių metodu – ties 61%, voltažas – 3,786 V. Tai pat kai turime 64% įkrovimo, paprastu metodu – 3,840 V, stulpelių merodu – 3,806 V. Tokie nesutapimai galimi dėl to, kad skirtingu laiku vyko eksperimentai, nevienodai buvo pakrauta baterija. Dar labai priklauso nuo to, kas prieš tai buvo su delninuku daryta.

Atlikdami kitą eksperimentą su matrica, kuri bus sudaryta iš 100 eilučių ir 100 stulpelių, atsižvelgiame į gautus rezultatus matricos, kuri buvo sudaryta iš 400 eilučių ir 400 stulpelių. Žiūrėdami į stulpelių saugojimo metodo rezultatus, kai operaciją įvykdo maždaug per 28 minutes, apskaičiuojame iteracijų skaičių, kad tiek panašiai truktų matricos 100 x 100 daugyba stulpelių saugojimo metodu. Gauname, kad reikėtų įvykdyti 2600 iteracijų. Mes taip ir atliekame. Vykdome eksperimentą, kurio metu dauginama 100 x 100 matrica, su 97% išretinimu, 2600 dauginimo iteracijų.

Atlikę eksperimentą gauname sekančius rezultatus, kurie surašyti 14 lentelėje.

14 lentelė. Stulpelių saugojimo metodo daugybos rezultatai

Parametrai	Matrica 100 x 100, 97% išretinimas, įvykdyta 2600 dauginimo iteracijų (stulpelių saugojimo metodas)
Bendras operacijos laikas	25 min 46 s
Vienos iteracijos vykdymo laikas	0,59 s
Operacijos metu baterija išsikrovė	9 %
Voltažo pokytis	0,068 V

Gautus rezultatus sulyginame su atitinkamais rezultatais (15 lentelė), kurie buvo gauti dauginant 400 x 400 matricą su 97 % išretinimu, vykdant 10 įteracijų.

15 lentelė. Stulpelių saugojimo metodo daugybos rezultatai

Parametrai	Matrica 400 x 400, 97% išretinimas, įvykdyta 10 dauginimo iteracijų	Matrica 100 x 100, 97 % išretinimas, įvykdyta 2600 dauginimo iteracijų
Bendras operacijos laikas	27 min15 s	25 min 46 s
Vienos iteracijos vykdymo laikas	2 min 44s	0,59 s
Operacijos metu baterija išsikrovė	9%	9 %
Voltažo pokytis	0,069 V	0,068 V

Matome, kad bendras operacijų vykdymo laikas beveik sutampa. Tai pat sutampa baterijos išsikrovimo lygis ir voltažo pokytis.

Toliau dauginame tą pačia matricą, tik pakeičiame matricos išretinimą į 95%. Gauti rezultatai pateikti 16 lentelėje.

16 lentelė. Stulpelių saugojimo metodo daugybos rezultatai

Parametrai	Matrica 100 x 100, 95% išretinimas, įvykdyta 2600 dauginimo iteracijų (stulpelių saugojimo metodas)
Bendras operacijos laikas	28 min 26 s
Vienos iteracijos vykdymo laikas	0,656 s
Operacijos metu baterija išsikrovė	10 %
Voltažo pokytis	0,068 V

Taip pat sulyginame lentelės rezultatus, su 400 x 400 matricos rezultatis, su 95 % išretinimu.

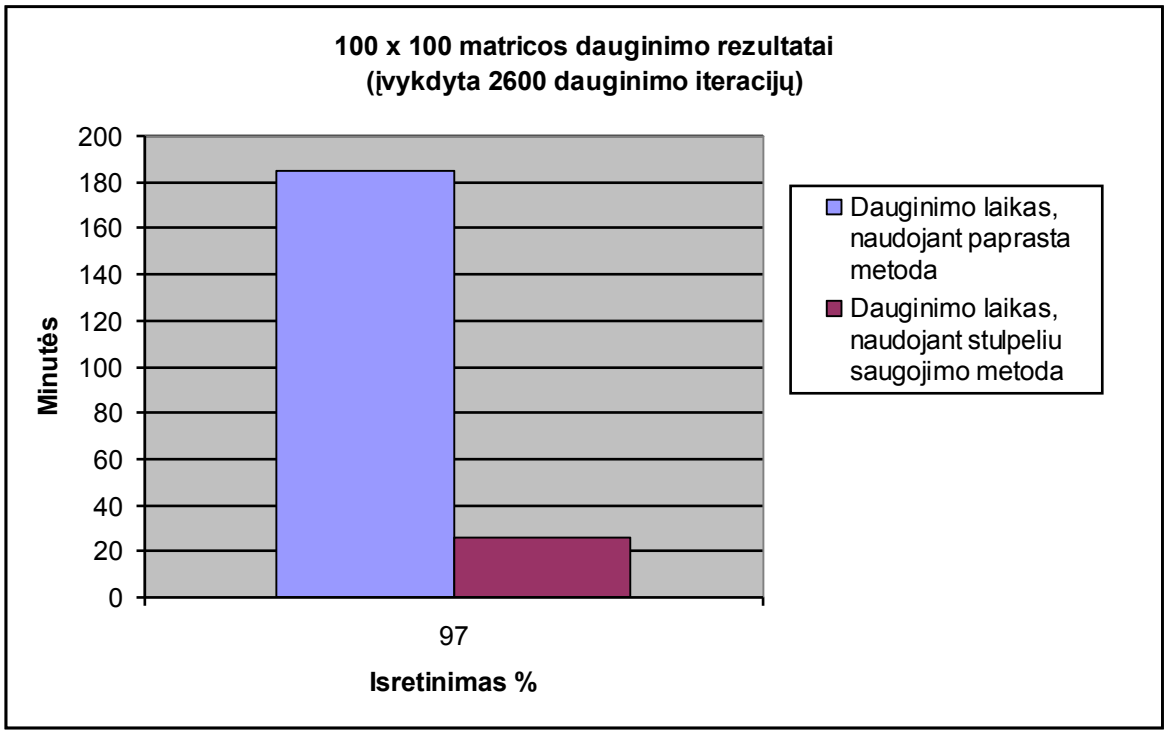
17 lentelė. Stulpelių saugojimo metodo daugybos rezultatai

Parametrai	Matrica 400 x 400, 95% išretinimas, įvykdyta 10 dauginimo iteracijų	Matrica 100 x 100, 95 % išretinimas, įvykdyta 2600 dauginimo iteracijų
Bendras operacijos laikas	28 min 20s	28 min 26 s
Vienos iteracijos vykdymo laikas	2 min 50s	0,656 s
Operacijos metu baterija išsikrovė	10%	10 %
Voltažo pokytis	0,063 V	0,060 V

Iš 17 lentelės pastebime, kad sutampa bandras operacijos vykdymo laikas, baterijos išsikrovimo lygis ir voltažo pokytis.

Iš gautų palyginimų galime padaryti išvadą, kad nesunkiai galima prognozuoti dauginimo operacijos vykdymo laiką. Turėdami matricos dauginimo rezultatus: visą vykdymo laiką ir norėdami su kitos dimensijos matrica gauti panašų laiką, pakanka žinoti naujos matricos vienos iteracijos vykdymo laiką.

Su tapačia matrica atliekame eksperimentą naudodami paprastą dauginimo metodą. Ir gautus rezultatus sulyginame su stulpelių saugojimo metodo gautas rezultatais.

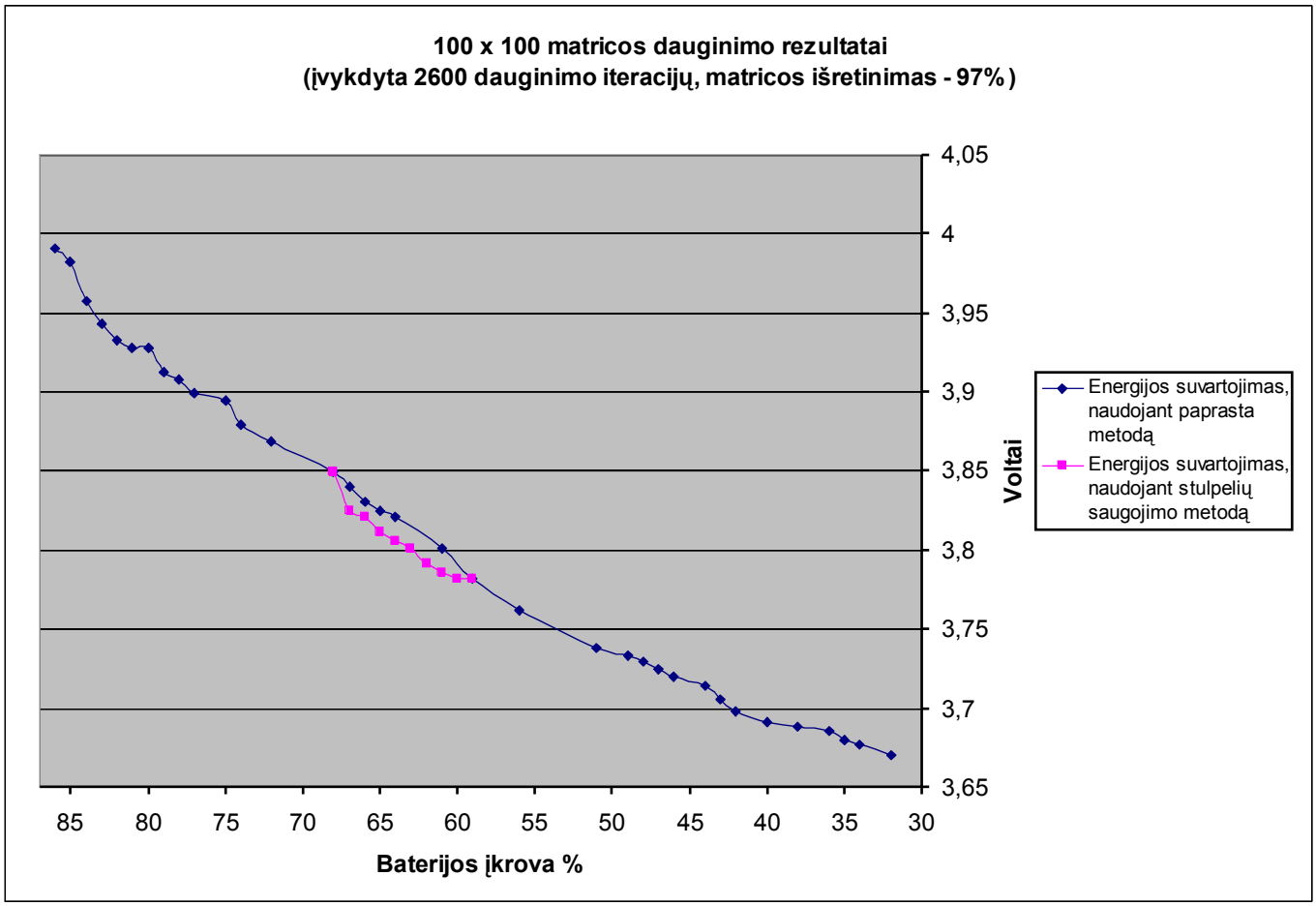


24 pav. Paprasto metodo ir stulpelių saugojimo metodo dauginimo laikų palyginimo diagrama

24 diagramoje matome abiejų metodų dauginimo laikus, kurie labai skiriasi. Paprasto metodo operacija užtruko 183 minutes (3 valdandos ir 3 minutės), o stulpelių saugojimo metodas – 25 minutes ir 46 sekundes. Skirtumas didelis, o rezultatas gaunamas tas pats. Tai dar kartą įrodo išretintų matricų stulpelių saugojimo metodo efektyvumą. Palyginus gautus rezultatus su 400 x 400 matricos dauginimo laiku, galime padaryti išvadą, kuo ilgiau operacija trunka paprastu metodu, tuo efektyvesnis pasidaro stulpelių saugojimo

metodas. Ir, kad efektyvumas priklauso nuo operacijos vykdymo laiko, kuo ilgiau vykdomos operacijos tuo trumpiau viskas įvyksta.

Dabar palyginsime baterijos nusėdimo lygį.



25 pav. Energijos suvartojimo grafikai naudojant paprastą ir stulpelių saugojimo metodus.

25 paveiklse matome energijos suvartojimo grafikus. Dauginant paprastu metodu baterija išsikrovė – 54 %, o stulpelių saugojimo metodu – 9 %. Lygintant gauname, kad stulpelių saugojimo metodas, šiuo atveju yra 6 kartus efektyvesnis už paprastą.

Apibendinant viso eksperimento rezultatus, galime padaryti tokias trumpas išvadas:

- Eksperimeto metu akivaizdžiai įrodėme, kad stulpelių saugojimo metodas yra efektyvesnis už paprastą dauginimo metodą.
- Išretintų matricų stulpelių saugojimo metodo dauginimo operacijos trukmė priklauso nuo matricos išretinimo procento. Kuo matricos išretinimas didesnis, tuo operacija užtunka trumpiau.

- Stulpelių saugojimo metodo efektyvumas lyginant su paprastu metodu, priklauso nuo operacijos vykdymo trukmės. Kuo ilgiau operacija truks paprastu dauginimo metodu, tuo efektyvesnis bus stulpelių saugojimo metodas.
- Žinodami vienos iteracijos vykdymo laiką, galime nesunkiai prognozuoti visos operacijos vykdymo laiką.
- Stulpelių saugojimo metodas užima mažiau vidinės atminties, negu paprastas metodas.
- Kadangi procesoriaus apkrautumo negalime niekaip išmatuoti, sprendžiame, kad visų dauginimo operacijų metu jis dirba pilnu pajėgumu – 100 %. Todėl procesoriaus palyginimas atkrenta ir lieka atminties palyginimas, kuris yra akivaizdus.

4.2 Surašymas į masyvus stulpelių saugojimo metode

Norint, kad stulpelių saugojimo metodas būtų maksimaliai efektyvus, svarbi yra į masyvus surašymo tvarka. Viską sprendžia dauginimo algoritmas. Mūsų atveju yra tikrinami tarpusavyje indeksai ir jei jie sutampa daugyba atliekama, jei ne einama prie kito elemento. Sakykime yra vienos matricos vienas eilutes elementas, jo indeksas lyginamas su kitos matricos stulpelio indeksu. Nesutapimo atveju eilutes narys išlieka tas pats, o kitoje matricoje pereinama prie kito stulpelio elemento. Ir taip kol bus sudauginta visa matrica.

Tarkime, kad surašymas į masyvus vyksta atsitiktine tvarka. Tada vienodų indeksų paieška išauga ir visas dauginimo efektyvumas labai krenta. O tai svarbu didelės dimensijos matricose.

Optimaliausias įrašymo metodas seka iš matricų daugybos taisyklės. Pagal taisyklę, eilutė dauginama iš stulpelio. Pirmos matricos nenulinius elementus į masyva surašome keliaudami per eilutes, iš kairės pusės į dešine ir taip, kol surašome visą matricą. Kitos matricos nenulinius elementus surašome keliaudami per stulpelius, iš viršaus į apačią. Dauginimo metu elementai atsidurs jau reikiamoje vietoje ir vienodų indeksų paieškos laikas minimizuojasi.

26. pav Matricos elementų surašymo į masyvus tvarka

Išvados

- Atlikome energijos suvartojimo įvertinimą taikomosios programos lygmenyje, naudojant išretintų matricų stulpelių saugojimo metodą.
- Išanalizavome ir pamatėme, kad baterijos gyvavimo ilgaamžiskumas išlieka pagrindine šių dienų problema.
- Išanalizavome išretintų matricų stulpelių saugojimo metodą.
- Sukūrėme paprastos ir stulpelių saugojimo metodo daugybos algoritmus.
- Parašėme testavimo programą, kuri atlieka daugybą dviem metodais ir išsaugo įvairius baterijos parametrus.
- Eksperimento metu pamatėme ir nustatėme, kad stulpelių saugojimo metodas yra žymiai efektyvesnis lyginant su paprastu metodu.
- Stulepių saugojimo metodas, vykdant daugimo operacijas užima mažiau vidinės atminties negu paprastas metodas.
- Kadangi negalėjome jokiais būdais išmatuoti procesoriaus apkrautumo lygį, priėmėme prielaidą, kad procesorius visų operacijų metu dirba pilnu pajėgumu.
- Visiem programuotojam ir ne tik, kurie kurdami programas naudoja matricas, siūlome naudoti išretintų matricų stulpelių saugojimo metodą. Jis tikrai padarys programą žymiai efektyvesnę ir baterijos resursų bus sunaudojama žymiai mažiau.

Literatūra

1. Rajenda Turakani and Krishna Kumar. Power optimization can extend the battery life in your portable Multimedia system. // *Techonline community. Mobile handset designline 2006.*
2. Lawrence S. Brakmo, Deborah A. Wallach, Marc A. Viredaz. qSleep: A Technique for Reducing Energy Consumption in Handheld Devices. *// International Conference On Mobile Systems Applications And Services, Proceedings of the 2nd international conference on Mobile systems, applications, and services. [2004].*
3. Kayun Chantarasathaporn and Chonawat Srisa-an. Object-Oriented Programming Strategies in C# for Power Conscious System *// International journal of computer science. ISSN 1306-4428. 2006, Volume 1, Number 1, p. 54*
4. Claudio Scordino, Giuseppe Lipari. Using resource reservation techniques for power-aware scheduling. *// International Conference On Embedded Software; Proceedings of the 4th ACM international conference on Embedded software. [2004]*
5. J. Valančius, V. Štuikys. Delninuko energijos suvartojimo įvertinimas taikomiosios programos lygmenyje. *// KTU IT konferencija 2007.*
6. *Floridos universiteto išretintų matricų kolekcija.* [interaktyvus] [Florida] [žiūrėta 2007-11-21]. Prieiga per internetą: < http://www.cise.ufl.edu/research/sparse/matrices/ >
7. *Vikipedia. Laisvoji enciklopedija.* [interaktyvus] [Lietuva] [žiūreta 2007-10-15]. Prieiga per internetą: < http://lt.wikipedia.org/wiki/.NET_Framework >
8. Sparse Matrix. Engineering and scientific subroutine library (ESSL) documentation. ESSL version 3, release 1.2 Guide and Reference. Prieiga per internetą: < http://www.navo.hpc.mil/usersupport/IBM/ESSL/essl148.html >
9. Dan LePre ir Jon Rauschenberger. Using Smart Device Extention and .NET Compact Framework Beta 1 to Build Applications in Visual Studio .NET. *// Microsoft Corporation. [2002]*
10. Tajana Simunic, Luca Benin, Giovanni De Micheli. Energy-efficient design of battery-powered embedded systems. // International Symposium on Low Power Electronics and Design; Proceedings of the 1999 international symposium on Low power electronics and design. [1999]

Pocket PC Energy Consumption Using Sparse Matrix Storage by Columns Modeling

Summary

Nowadays major problem is energy consumtion in portable devices which has a battery. We have decided to investigate Pocket Pc power consumtion, because there are a lot of programs which are not effective and takes much battery energy. So in this job we have evaluated energy consumption for Pocket PC. Inspected power aware problem in these computers and saw, that there are quite alot suggestion how to solve this problem in application level. But we wanted to see memory and processor influence in battery energy consumption.

We have created a program which can do matrix multiplication and sparse matrix „storage by collumns“ multiplication. During multiplication program takes battery information and save's it into the file. After that I have investigated the result and saw, that sparse matrix storage by columns multiplication is much more effectived than normal matrix multiplication. So we suggest to use sparse matrix storage by columns model instead simple model, because you can save much more operation time, battery resources and memory. These parameters are very important to portable devices such as Pocket Pc.

Priedai

Pateikta sugeneruota paprasta išretinta matrica, kuri turi 100 eilučių ir 100 stulpelių. Išretinimo procentas - 91

10 0 0 0 0 39 0 0 0 0 0 0 0 3 0 18 35 40 0 15 0 25 0 0 0 0 0 23 0 34 0 0 25 0 0 46 6 12 0 26 39 0 0 30 0 0 0 0 0 0 21 0 0 41 25 0 7 0 9 0 0 34 0 0 0 33 0 0 29
28 11 12 0 0 0 0 21 45 0 32 40 0 0 12 0 0 19 41 0 12 0 0 0 0 40 0 0 23 22 26 0 10 0
0 0 48 0 0 46 0 0 0 11 0 26 0 18 0 0 0 0 0 2 0 0 0 0 0 11 0 0 3 48 0 0 22 0 0 5 13 0 28 0 33 0 41 0 41 0 0 0 0 0 0 44 19 0 7 0 0 0 38 0 0 21 0 0 22 0 21 4 0
43 0 39 36 29 0 19 0 24 3 0 44 10 0 0 0 3 49 0 14 34 1 0 0 47 23 0 0 3 18 12 19
16 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 39 0 0 0 14 0 0 42 0 14 0 0 0 4 31 6 0 49 0 0 24 46 0 0 0 14 0 0 0 0 0 0 0 33 0 0 0 18 0 0 0 35 23 0 19 0 3 0 0 0 0 18 0 43
0 0 8 0 0 49 0 0 40 0 6 0 0 5 29 0 0 0 0 0 45 0 22 0 0 0 11 0 0 0
0 0 0 0 39 0 0 0 0 0 0 32 45 49 0 0 9 47 0 0 0 0 24 0 0 0 19 0 0 11 0 0 0 31 0 0 0 0 16 0 0 30 0 0 35 13 44 23 0 46 13 14 38 36 22 0 14 0 38 0 13 12 0 0 0 0
0 0 16 41 12 0 0 8 0 0 0 35 0 0 10 0 0 0 0 23 0 21 0 9 0 0 49 0 0 0 0 18 0 0
0 30 0 0 0 0 0 0 30 0 15 42 0 15 0 0 0 0 25 0 43 0 0 0 7 0 0 21 0 0 0 3 0 0 0 0 0 0 0 0 0 5 0 0 4 43 17 0 0 0 0 13 0 0 0 27 0 0 0 0 0 0 0 0 32 4 41 0 30 0 0 0 44
0 21 8 0 0 0 49 0 0 14 0 26 0 0 0 26 0 36 0 0 0 38 0 32 0 8 0 13 0 0
9 0 35 12 34 0 19 0 24 0 0 0 0 0 33 44 0 42 39 36 0 0 0 0 0 40 0 25 0 43 0 0 28 0 0 0 40 30 0 0 6 39 0 0 26 37 0 13 0 40 0 3 24 45 0 0 0 0 0 0 0 0 43 30 0 0
0 7 31 0 3 0 0 38 29 25 0 42 21 0 0 7 0 0 0 12 11 0 0 24 0 0 0 34 0 12 0 0 0 0
49 0 0 0 36 10 0 35 0 0 14 0 16 0 0 0 17 0 0 34 0 4 27 0 0 0 0 0 17 0 0 0 33 0 0 0 0 37 0 0 0 25 17 0 9 0 0 39 0 49 44 0 0 0 0 45 43 0 0 36 10 9 18 0 0 0 0
21 0 40 0 0 0 42 0 0 0 0 0 13 14 10 26 0 0 0 9 0 7 0 0 0 18 0 0 0 0 0 14 28
0 14 0 0 0 14 0 32 0 24 8 0 0 0 0 49 0 1 0 32 0 0 0 0 0 0 0 0 0 45 6 0 0 0 0 49 35 0 0 0 0 0 44 0 49 0 0 0 0 0 32 0 39 0 0 6 0 0 0 0 0 0 0 0 17 38 0 0 43 0
0 0 0 21 0 0 0 0 7 0 0 0 0 0 0 0 0 49 0 0 0 16 0 8 0 6 0 0 15 0
0 16 18 0 0 0 19 0 33 0 0 0 0 26 33 3 3 0 0 0 0 16 0 0 0 47 0 27 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 48 28 0 17 25 0 0 0 21 0 0 0 0 30 2 43 39 0 0 42 0 0 0 0 0
0 23 0 14 5 0 46 0 44 0 0 0 0 0 17 0 0 1 0 0 0 0 0 46 1 9 0 11 23 0 14
0 0 0 0 0 48 0 0 0 5 0 0 40 6 0 0 0 0 0 5 9 0 36 19 0 0 24 0 0 46 0 0 27 44 25 0 0 0 3 41 39 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 44 19 0 0 37 31 0 0 0
0 35 0 0 0 0 0 0 41 0 15 19 0 0 44 6 0 0 42 0 29 0 44 0 49 37 0 38 19 41
0 0 5 0 0 15 0 0 0 0 0 0 0 28 0 0 0 49 0 0 27 0 22 49 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 29 19 8 40 5 29 0 31 0 0 0 0 0 16 5 0 49 0 5 0 0 0 36 0 21 0 0 30 0 25
0 5 0 25 42 0 0 37 0 0 0 26 0 47 36 34 11 27 4 34 0 0 8 0 0 0 0 0 19 0 0
7 0 40 0 0 0 0 23 0 0 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 27 0 0 0 0 0 0 0 3 26 0 34 7 12 0 19 0 42 0 41 0 23 0 0 6 0 0 0 0 0 0 0 0 0 26 0 13 0 0 2 0 0 32 32 0 0
0 31 35 0 0 0 0 0 31 0 24 0 18 0 0 22 0 0 20 0 0 0 0 0 0 30 18 17 0 0 0
0 40 0 46 42 0 0 0 24 0 48 13 0 0 1 0 32 6 36 26 9 0 11 23 0 46 49 0 11 7 0 0 27 0 0 0 0 22 0 0 33 0 45 0 0 0 31 0 34 0 0 10 0 0 37 0 29 4 0 0 0 39 0 16 0 6
0 0 10 0 0 10 0 0 42 15 0 0 0 28 0 0 14 0 21 0 38 0 0 48 0 0 46 41 40 0 15 0 33 0
0 0 0 17 0 0 40 0 21 0 0 37 26 0 0 33 0 0 44 0 0 22 4 11 0 21 13 0 0 48 0 39 0 0 23 0 18 47 0 0 13 0 42 24 15 0 0 0 0 0 0 0 33 0 0 0 25 0 3 0 11 0 28 0 21 0
0 0 0 0 0 0 34 0 0 1 0 0 0 46 0 14 0 36 29 0 6 0 17 41 0 20 0 27 34 0 0 19 0 22
36 37 0 0 41 41 0 0 0 0 2 0 2 0 0 0 44 0 0 0 0 0 0 0 0 0 44 35 0 14 0 13 0 0 0 0 0 0 0 0 0 0 0 21 0 0 39 0 17 35 16 36 0 46 0 34 0 0 0 0 0 0 5 0 19 0 0 0 0 29 0
3 0 0 46 5 0 0 0 0 0 0 27 0 41 28 12 16 0 0 0 0 0 0 43 0 0 0 34 0 13 41 47
0 0 0 12 0 14 0 0 5 0 15 0 1 0 0 6 0 41 49 39 0 0 0 1 0 0 0 48 0 0 12 0 0 44 25 39 0 0 0 0 0 13 36 0 30 44 42 12 28 25 0 11 0 0 0 19 48 0 49 0 0 1 0 0 0 0 0
0 34 0 0 36 0 0 0 0 40 0 0 19 0 0 0 47 0 1 20 29 0 6 40 0 0 40 0 15 15 24 0 49
30 0 27 0 0 0 0 11 0 46 0 27 0 0 0 36 0 0 37 5 0 0 2 0 0 0 0 0 48 6 28 0 0 21 0 0 0 0 0 39 29 0 0 0 0 0 0 48 0 0 0 15 37 0 0 0 0 0 0 0 0 0 13 0 0 13 19 42 0
0 6 0 33 0 0 0 34 0 0 0 0 0 0 0 0 0 28 0 33 42 0 31 0 0 0 0 0 0 0 13
0 19 0 0 10 0 31 0 0 0 11 39 47 31 7 0 28 0 0 4 0 0 0 48 0 24 0 0 0 20 0 0 0 0 0 32 0 2 41 0 0 0 0 20 0 0 0 33 0 0 0 27 0 0 0 0 0 0 0 0 0 35 27 0 0 10 0 0 0 8 0
0 0 0 0 0 0 0 0 47 0 0 28 41 0 36 0 43 0 28 0 11 0 26 0 32 34 0 30 36 0 0 0
0 0 0 24 28 40 1 0 0 31 0 49 40 0 0 0 14 0 17 0 0 18 0 44 0 0 0 0 32 0 0 0 39 0 48 16 0 0 0 0 0 0 3 5 0 0 0 3 0 39 0 0 23 0 0 0 0 27 0 0 6 0 0 0 0 0 0 0 0 0 0 0
30 0 49 0 0 0 0 0 17 0 18 0 36 25 0 0 0 2 0 38 0 0 0 0 15 0 0 0 0 0
0 0 20 0 26 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 34 0 0 0 0 0 0 0 0 0 0 12 26 0 0 0 14 31 0 48 0 0 0 0 11 40 30 25 0 18 0 0 41 0 24 36 0 43 0 22 42 0 0 0 23 15 0 49
0 0 0 0 0 0 0 24 0 0 3 48 0 0 9 49 24 0 0 21 0 34 0 44 0 0 0 0 0 30 0 0 46 19
0 0 11 43 29 16 0 23 0 0 0 45 24 0 0 0 0 11 29 0 0 14 0 15 0 0 0 0 2 0 0 0 1 33 0 20 0 0 38 27 49 5 0 42 0 0 0 39 23 28 0 0 2 2 0 0 0 0 0 31 6 7 20 10 10 0
43 0 35 11 6 0 0 2 44 0 0 47 0 29 0 0 40 0 0 19 18 0 0 0 21 41 40 6 0 0 0 0 32 6
31 0 0 0 0 0 0 0 0 38 0 0 0 10 14 35 0 46 0 0 0 0 33 22 40 0 0 17 0 2 3 0 15 0 47 27 0 0 1 20 17 0 0 0 28 19 20 0 0 0 24 0 29 0 42 37 0 31 21 7 0 34 0 5 0 0
0 0 0 36 0 0 44 22 0 45 0 15 0 20 0 0 0 46 27 16 0 0 0 0 22 34 0 0 0 0 49 35 0 0
0 0 0 0 15 0 0 6 0 0 34 39 37 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 32 0 0 26 36 0 0 0 23 0 0 28 25 32 46 0 0 0 0 26 7 0 0 18 0 15 0 29 4 0 41 28 0 0 0 49
0 37 21 40 0 0 45 19 0 0 26 32 38 25 0 0 15 0 0 17 0 0 36 0 0 0 37 32 0 0 28 0
24 0 0 0 0 32 0 0 19 41 10 0 0 8 41 0 20 0 0 0 0 0 0 0 42 0 0 0 40 49 13 38 0 20 0 8 40 0 29 37 0 0 39 0 35 0 46 31 0 0 0 0 5 0 49 0 0 25 0 41 0 0 43 21 0 3
29 20 0 0 0 0 0 0 0 0 17 42 0 0 0 0 0 31 39 0 32 17 0 0 24 14 30 38 0 0 0 0 42 0 0
25 9 17 8 0 21 16 33 6 33 41 0 17 36 9 0 12 0 0 20 0 0 0 26 35 0 2 30 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 33 25 13 1 0 0 0 37 0 11 0 0 0 0 23 0 0 0 0 0 0 0 0
0 0 0 0 0 24 0 28 0 0 0 10 2 0 0 27 19 0 0 43 0 3 5 0 0 0 11 30 16 0 47 0

0 0 0 0 40 0 32 12 0 0 0 30 0 0 45 0 0 0 0 0 0 40 0 0 0 0 0 40 24 0 23 11 26 0 1 0 0 0 0 37 0 0 0 36 0 0 24 0 0 34 0 0 26 0 0 13 0 25 38 4 0 49 0 49 0 0 0 0
47 0 43 9 20 0 6 44 0 0 0 41 0 26 4 0 18 0 35 0 1 0 0 0 0 0 0 9 0 0 0 3 0 0
19 0 0 0 0 0 0 0 0 0 35 0 0 47 0 0 0 17 0 36 0 0 0 0 26 0 0 0 0 42 27 0 0 0 3 41 0 37 0 0 33 0 10 0 22 0 21 0 21 0 0 0 0 0 0 31 0 0 0 15 22 0 15 32 38 0 7 46 45
0 34 0 32 17 0 0 0 20 0 8 4 40 0 0 0 43 0 8 0 0 49 34 0 0 0 0 0 0 28 0 17 44 24
0 0 10 0 1 34 20 19 49 21 0 20 0 0 0 19 0 0 33 0 0 0 6 0 5 35 0 0 0 0 0 14 13 0 9 3 0 37 0 36 0 0 22 0 0 37 0 0 0 4 0 5 0 45 0 0 0 10 40 0 0 0 45 0 0 0 0 31
22 22 0 0 0 0 0 0 0 0 28 0 0 0 0 0 18 0 0 0 17 17 1 45 0 47 0 42 27 19 0 0 0
30 4 22 0 0 0 0 38 35 0 7 30 0 22 16 0 0 0 22 0 0 0 0 20 0 6 13 0 3 0 0 0 0 17 0 44 0 0 0 0 0 0 0 2 0 0 0 24 8 0 15 42 2 0 42 0 0 0 0 14 4 0 0 0 0 0 30 0 34
0 25 0 0 0 21 47 6 10 0 11 0 0 19 8 0 0 22 0 0 0 3 21 46 33 1 0 6 30 0 0
0 0 39 36 0 0 0 0 0 0 0 0 25 0 22 0 0 0 35 0 10 22 19 0 0 3 0 0 0 0 41 26 44 14 0 21 0 0 0 0 0 0 0 30 12 22 45 0 10 0 35 0 0 16 0 0 0 0 0 15 7 16 0 0 0 7 36
0 0 0 17 0 0 0 0 0 0 38 0 0 0 9 35 29 0 10 31 27 11 0 0 47 0 0 0 0 0 48 0 4
0 0 22 0 0 14 0 14 0 28 0 0 0 0 0 0 31 40 0 0 0 0 49 0 0 19 5 13 33 20 0 0 0 0 0 0 0 37 26 11 0 0 16 0 33 0 0 0 43 15 0 0 25 7 34 27 27 0 0 28 0 9 0 30 18 0
37 0 4 0 0 0 4 29 0 0 0 0 16 0 23 0 0 0 34 0 0 31 26 0 0 38 23 0 0 0 0 0 19 0 0
5 17 31 31 38 0 46 48 8 0 0 0 44 0 0 0 33 45 23 0 6 25 10 0 45 0 7 0 0 0 0 0 0 0 0 31 29 15 45 24 0 0 0 25 0 0 28 42 12 0 1 40 0 0 0 0 6 0 36 0 0 0 12 16 44
0 0 0 37 0 0 0 0 48 0 23 7 0 0 25 0 0 4 0 0 33 0 29 0 0 0 0 29 44 0 6 0 0 30 0
4 0 0 0 0 0 0 0 0 0 0 20 0 0 0 48 0 0 26 11 1 0 0 0 44 0 39 0 0 0 0 5 4 37 0 0 0 11 0 7 0 7 0 0 0 0 0 18 2 13 31 17 25 0 16 0 43 0 0 16 0 34 10 0 0 0 0 0 39
0 0 0 0 0 18 0 27 0 35 0 0 0 0 0 0 36 0 0 21 2 0 3 0 0 13 0 47 32 0 47
0 0 0 18 23 18 18 0 0 22 41 35 19 0 8 0 0 0 0 0 0 31 37 37 18 0 0 0 0 0 0 45 29 0 2 0 0 34 0 0 0 23 33 0 0 29 33 0 27 0 28 0 20 0 10 0 0 0 0 0 0 0 0 0 0 0 0
31 0 24 0 0 0 0 0 13 48 0 0 38 0 47 0 0 0 0 0 0 27 46 0 0 34 14 0 11 0 41 0 14
0 1 0 2 0 33 0 0 0 0 0 7 47 0 0 0 0 0 0 0 0 0 0 0 0 9 26 31 0 0 39 3 40 6 0 0 14 0 0 31 0 47 9 0 0 30 39 0 0 23 30 5 0 0 30 0 0 0 47 37 0 26 22 0 15 25 0 1 0 0
0 32 0 0 26 0 0 0 17 0 20 0 0 0 40 33 0 0 0 0 0 11 3 0 5 0 25 0 0 28 0 37
18 0 26 0 0 0 0 17 0 0 36 0 0 0 28 35 0 31 0 3 0 47 35 35 29 8 0 0 0 0 48 0 49 0 0 0 10 0 0 0 0 0 7 0 28 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 24 0 42
0 0 0 0 0 0 0 14 6 0 0 13 28 0 0 0 25 23 0 0 0 22 0 0 22 0 0 0 0 37 0
0 10 0 0 0 22 39 16 0 42 0 0 30 19 0 0 28 0 0 0 0 45 13 0 0 0 0 10 3 40 40 7 0 32 43 2 16 47 0 0 0 0 0 28 28 0 38 4 8 0 0 0 0 0 0 0 0 0 5 0 0 0 0 13 0 0 0 0
47 25 0 0 0 0 0 30 17 25 34 27 3 0 0 1 0 0 0 44 0 0 0 0 10 43 25 0 0 43 2 0 0
15 40 28 0 36 0 0 0 0 0 34 0 0 0 0 0 0 18 21 7 0 0 0 41 4 0 0 21 0 0 24 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 32 0 34 47 0 0 31 0 0 0 21 37 0 0 26 27 0 16 0 0 0 0
13 13 0 23 0 0 0 0 13 0 0 0 29 0 37 5 0 0 0 38 26 42 0 0 0 33 6 17 42 0 0 39
0 0 0 0 25 5 0 0 0 34 30 0 0 0 0 0 40 0 0 0 19 0 3 0 8 47 13 0 0 45 0 23 0 2 22 0 15 0 30 0 0 47 31 0 0 9 28 26 0 0 0 0 0 17 0 0 49 8 0 0 0 0 0 0 0 0 49 44
10 17 17 24 0 0 0 0 34 0 0 48 18 37 0 7 0 0 0 28 0 49 0 6 0 0 48 0 35 0 0 0
10 0 16 21 6 0 0 37 38 0 0 0 0 13 0 0 0 4 41 0 0 20 0 10 0 0 21 35 0 0 0 0 7 0 0 0 21 0 0 16 0 15 0 35 0 0 6 0 42 0 0 0 29 0 0 0 0 5 30 46 31 0 0 4 11 0 0
45 26 24 33 0 0 24 0 37 0 26 0 2 0 0 0 0 38 0 0 20 0 47 12 15 0 3 35 0 11 35 0 37
0 0 13 0 11 0 0 0 3 12 0 45 0 0 0 0 0 15 0 5 0 10 41 0 12 0 49 0 38 0 0 44 2 10 45 0 17 0 28 30 0 0 0 0 0 16 9 8 0 36 0 45 19 0 0 46 20 0 5 0 9 0 0 3 0 16 0
0 0 0 0 0 37 0 0 12 0 0 0 0 0 0 0 2 0 30 0 33 14 0 20 32 0 0 0 0 12 0 0 0
0 0 27 0 4 0 0 0 41 0 0 21 0 0 36 0 0 0 25 0 42 0 31 32 0 0 16 2 15 0 0 0 0 20 0 44 12 29 48 0 22 40 42 29 0 0 38 9 48 0 15 0 38 0 28 0 0 0 0 0 19 5 0 0 0 0
0 23 0 0 5 0 14 0 32 0 17 0 0 0 10 0 0 0 8 1 0 0 0 10 0 17 49 43 0 16 0 0 0 0
0 0 22 0 0 8 0 5 0 0 0 30 8 49 0 0 0 0 0 32 33 33 0 0 0 0 7 20 11 0 29 0 6 15 0 44 0 0 0 0 0 0 0 0 0 0 8 0 0 10 16 16 0 0 37 0 0 46 26 0 0 0 43 0 0 0 0 42 0
0 26 1 36 0 0 47 0 0 0 0 0 0 0 0 0 49 0 0 0 0 18 0 0 22 0 0 0 0 0 0 9
0 16 8 49 33 0 0 0 0 0 30 20 0 0 27 0 0 0 2 33 0 29 0 0 0 0 0 0 0 0 5 35 45 2 39 0 0 0 0 0 0 11 47 0 0 0 0 44 0 0 0 49 20 0 0 28 1 34 0 46 35 3 3 34 0 45 0
31 38 0 45 19 25 0 0 20 0 0 0 0 0 0 0 43 0 0 0 0 9 0 11 0 13 0 21 0 0 0 13 0 0
12 0 6 0 0 0 24 2 0 0 0 0 0 5 0 1 0 0 0 0 30 16 0 27 0 31 5 0 17 0 12 0 0 0 0 9 28 24 41 7 31 0 0 16 14 0 0 7 0 0 0 0 28 0 11 0 0 0 15 0 36 0 34 0 45 0 0 0
0 0 0 35 9 0 0 0 0 41 0 0 16 47 0 19 0 0 0 49 0 0 44 0 0 13 0 0 0 0 0 38
0 0 42 0 0 32 45 0 0 43 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 24 26 0 0 36 40 0 0 0 0 6 5 43 44 0 11 37 0 0 41 0 32 0 2 32 37 12 0 0 0 29 0 0 0 0 6 0 0 0 0 0 3 0
13 5 33 0 20 40 0 48 0 0 0 0 41 14 0 0 0 0 0 0 22 0 0 33 1 43 13 0 0 0 37 19
0 0 0 47 48 8 0 0 0 0 0 0 9 0 2 9 42 6 24 0 0 0 17 0 0 0 0 0 0 27 15 0 0 42 0 24 1 16 46 0 0 0 0 0 20 0 0 0 0 0 0 0 5 0 42 0 0 0 20 0 0 48 30 14 0 0 44 0 17 0
28 8 0 0 49 0 0 0 0 49 2 0 0 0 43 5 9 41 0 0 0 0 30 25 0 26 0 0 0 0 25 0
0 9 20 0 0 0 0 0 0 0 0 0 0 25 0 0 21 27 0 0 0 0 49 0 0 0 0 23 0 0 0 0 23 0 0 0 0 0 0 0 0 0 8 0 12 0 37 0 37 0 0 0 0 22 6 2 9 17 48 0 23 0 0 0 0 14 0 0 0 0
42 0 0 17 0 0 28 21 0 0 0 0 0 32 20 0 0 0 42 41 0 40 19 0 0 0 13 0 8 48
11 0 0 48 16 45 0 39 0 0 0 0 27 29 32 14 7 17 31 42 24 46 0 39 0 0 0 0 27 45 0 0 0 0 7 13 0 36 16 0 0 0 0 20 17 1 12 40 0 25 0 0 25 16 23 0 2 42 0 0 12 0 0
0 0 0 26 0 22 0 0 0 1 0 0 0 0 25 22 47 38 42 0 0 0 0 0 0 0 0 0 11 35 0 0 0 0 38 0 0 15
0 0 0 14 0 0 0 0 0 0 0 22 26 20 0 0 0 0 23 0 0 25 0 29 18 0 0 0 34 0 0 12 35 0 6 6 21 0 25 0 14 0 0 27 0 0 19 0 0 0 0 0 0 20 29 27 44 0 0 0 0 6 0 0 43 0 0 0 0 0 0
0 0 44 0 44 8 43 27 0 0 9 0 0 0 26 0 0 0 0 0 46 25 0 37 46 0 0 0 2 0 0 0
0 0 0 0 48 0 34 19 0 32 0 9 0 33 0 17 0 25 3 7 0 8 16 6 0 0 0 0 0 0 0 24 0 43 0 9 0 49 0 0 0 0 0 0 0 14 21 0 9 0 29 0 22 0 0 0 0 0 35 45 46 0 27 19 0 45 0 0 0 1
24 14 18 0 24 12 0 0 0 2 0 0 0 0 36 24 32 0 0 0 21 0 0 3 0 0 13 2 27 0 0 17
27 0 39 0 14 0 2 24 0 0 0 0 11 0 0 0 0 36 0 9 41 0 0 20 15 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 8 0 37 0 0 45 15 0 0 0 45 0 0 0 0 0 0 0 0 0 49 0 0 0 25 28 0 0 38 43
36 0 0 0 0 0 0 3 0 24 0 0 0 32 0 0 14 20 0 49 0 0 0 0 0 0 0 10 0 0 0 0 19
0 0 7 0 0 0 15 0 31 0 16 0 0 0 0 0 0 0 0 0 34 46 0 0 0 46 45 0 0 43 8 0 0 24 0 23 0 0 42 6 0 0 0 17 0 18 0 0 0 33 0 18 41 0 15 0 0 45 0 0 2 0 49 0 23 0 0 28
37 0 3 0 0 14 6 29 0 0 0 0 39 0 0 31 0 39 0 0 0 0 35 0 0 0 4 0 0 0 0 36 36
28 21 45 0 0 0 27 33 0 0 16 11 0 0 0 46 0 5 34 32 37 0 0 0 0 23 0 0 0 25 0 0 47 44 0 14 0 1 46 0 0 0 23 26 19 16 0 0 0 0 36 0 0 4 0 8 0 4 5 29 7 0 0 0 15 0
0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 42 0 0 0 0 46 17 0 3 0 13 0 0

1 12 45 38 21 19 9 0 0 23 0 0 0 0 0 0 0 0 0 42 21 21 0 8 0 0 20 41 8 0 0 0 0 40 0 18 0 0 0 18 12 25 0 21 0 8 35 0 0 0 0 0 0 0 0 0 0 36 0 0 49 0 33 0 0 0 21
43 0 0 0 0 0 3 30 0 0 13 18 0 0 0 22 0 0 0 0 0 0 0 0 41 0 0 0 0 0 44 42 0 0 0
0 41 11 0 0 41 0 0 0 13 9 0 0 41 1 0 0 0 0 0 39 0 0 0 0 37 10 17 10 34 35 0 4 32 0 0 25 0 0 0 0 0 0 2 0 0 17 0 0 5 0 39 0 0 0 20 0 0 0 0 0 0 12 0 47 0 0 0 4 0 0 0
25 20 12 8 0 0 36 0 45 45 0 0 0 28 44 35 0 20 0 0 0 0 10 30 0 37 26 0 9 0 0
0 47 0 5 0 0 0 0 0 0 0 32 0 0 9 0 18 8 0 14 34 0 0 45 0 0 0 44 0 0 0 0 38 32 12 5 43 0 0 49 45 5 28 0 33 14 0 0 0 26 0 0 0 0 28 0 0 0 0 7 0 0 0 24 47 5 0 21
29 0 0 0 0 0 0 19 0 40 18 0 13 4 44 40 0 0 0 8 45 3 34 0 19 0 18 0 39 0 3 0 27
0 35 0 0 8 0 7 24 0 0 0 0 11 13 3 0 0 0 0 0 47 0 0 0 24 0 0 13 22 0 0 36 0 0 0 18 0 39 0 38 38 34 21 3 0 0 0 33 14 36 24 0 0 5 30 0 18 0 0 10 13 0 0 0 44 0
45 0 19 34 22 0 29 34 11 12 27 13 0 0 0 0 0 0 0 0 0 0 6 0 23 0 0 12 0 24 44 0 28 0 0
26 30 0 30 13 0 17 45 43 0 43 0 13 0 0 0 0 0 0 36 45 0 0 0 0 49 13 0 10 16 0 0 4 0 0 0 1 18 23 7 40 10 13 36 29 0 0 0 16 26 0 0 40 21 47 37 8 0 0 0 35 0 35
0 0 0 0 7 0 9 0 26 0 0 0 0 25 0 0 0 40 0 0 33 0 0 9 32 0 38 0 33 0 10 22 0 25 0 0 0
0 2 0 0 0 23 0 9 0 0 0 22 0 0 23 0 0 1 9 26 34 0 48 0 8 13 0 31 0 28 0 0 0 0 0 0 34 0 0 44 34 0 7 45 25 0 0 15 36 0 0 0 27 21 1 28 0 0 48 15 0 0 1 47 0 0 34
0 0 0 26 0 0 0 42 34 9 0 0 0 0 37 0 0 0 35 0 0 28 0 0 0 21 39 0 0 0 0 0 0
0 0 0 0 35 45 26 0 0 0 0 0 0 0 0 32 0 0 0 0 12 0 43 0 0 22 4 0 0 0 0 49 10 33 0 0 36 42 0 0 0 49 0 11 0 34 7 0 0 0 0 0 20 0 0 24 15 49 0 0 0 19 21 0 0 19 0
12 34 0 45 0 43 27 0 0 0 0 0 16 15 0 0 0 0 0 29 0 45 10 0 36 41 0 0 0 0 11 0 0
44 0 0 0 0 26 0 0 0 0 0 32 43 0 0 38 0 0 0 0 0 0 16 0 0 8 0 44 0 20 36 0 1 0 21 0 0 2 0 0 0 0 21 0 0 0 0 6 0 0 0 4 0 44 35 40 0 44 0 0 9 0 0 0 0 0 0 11 13 0
0 0 22 0 0 0 0 45 5 0 0 0 0 21 0 0 16 0 46 0 0 0 0 12 0 39 47 0 0 0
46 32 43 0 30 0 0 40 0 0 8 33 0 24 23 0 12 17 0 0 2 23 15 0 36 0 10 0 0 0 0 6 0 0 0 32 16 17 0 43 0 0 0 15 0 0 0 0 38 0 27 0 0 0 0 0 0 0 0 17 0 0 7 0 24 0
21 32 47 0 0 0 0 0 0 0 0 0 0 0 11 31 0 0 0 0 0 0 0 9 4 0 0 0 23 0 0 0 2 0 20
35 0 32 47 0 0 3 29 0 19 0 0 1 0 5 0 0 0 0 47 27 36 23 0 0 25 0 0 0 26 7 0 0 0 44 43 0 0 0 0 0 45 0 40 0 0 0 0 0 0 0 0 26 0 42 0 45 39 21 0 0 0 0 0 0 37 0 0
0 0 0 25 0 0 0 31 0 0 0 0 21 44 34 16 46 0 0 15 0 0 0 9 0 0 0 0 19 0 0 12
38 43 0 0 21 3 0 43 0 0 0 0 0 0 0 0 0 9 48 0 48 0 0 0 0 0 23 0 8 0 0 4 5 0 40 28 37 28 45 42 0 46 0 0 21 35 12 8 11 0 0 0 0 13 0 44 0 38 0 0 0 45 0 16 0 20
0 0 2 0 35 37 0 0 8 1 47 0 46 0 0 0 0 0 0 0 0 0 0 0 0 0 48 0 30 13 10 11 0 0 0
0 46 0 0 16 0 46 15 4 0 0 0 0 0 0 15 0 0 0 48 0 0 0 0 29 0 1 33 0 48 0 0 43 45 0 46 0 3 39 27 5 0 0 0 49 48 0 0 0 0 2 33 0 0 0 0 0 0 3 0 17 12 23 6 0 28 6 0
0 0 0 0 0 41 0 0 0 46 0 0 31 27 0 0 0 0 0 7 0 36 0 0 0 0 0 0 43 0 0 0
0 8 0 0 0 0 0 18 0 42 0 49 43 23 0 0 36 29 10 15 0 0 0 0 27 0 0 27 31 0 27 0 0 0 0 0 0 0 2 47 0 0 0 40 49 0 15 0 0 20 0 13 21 15 48 42 45 0 3 43 0 17 0 0 0
0 0 7 35 0 0 40 0 0 0 33 36 0 0 0 32 0 0 0 0 0 0 0 36 0 0 0 42 9 0 44 1 0 0 8
0 0 0 0 47 0 37 0 12 15 0 22 0 0 0 0 0 26 0 0 17 33 20 0 47 0 47 17 0 0 0 16 46 0 0 0 0 25 0 0 26 15 37 21 0 5 20 0 47 35 0 0 19 0 0 0 0 0 41 4 0 19 0 3 33
0 25 0 0 0 0 0 12 0 0 0 0 49 0 41 0 31 19 23 0 6 0 0 0 0 0 33 0 0 11 0 0 34 0 49
0 0 0 0 3 15 12 0 0 39 37 37 0 49 0 36 0 25 0 27 0 0 0 39 1 48 0 0 0 17 0 11 0 0 0 0 31 9 19 0 0 0 0 0 0 49 0 0 0 25 0 0 47 10 0 26 0 0 9 0 23 0 39 0 0 0 24
0 0 0 5 41 0 0 0 0 32 31 0 1 0 0 0 0 11 45 0 2 45 34 33 21 0 13 27 0 8 37 31 0
24 0 0 0 0 0 0 0 26 0 0 0 0 7 0 0 34 5 22 8 0 0 4 0 0 0 40 0 7 0 36 44 0 0 0 0 0 0 14 45 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 44 0 0 0 0 0 0 0 0 0 0 0 8 9 0 41
26 0 0 0 0 13 0 0 0 0 0 0 0 0 0 0 26 0 13 48 0 0 0 0 0 0 40
0 0 0 0 1 1 11 0 26 0 4 8 0 0 0 0 23 7 0 0 0 3 0 0 0 45 0 12 5 0 0 0 0 22 0 35 34 0 45 0 0 40 0 44 0 44 32 0 0 0 0 40 0 0 0 0 0 0 8 0 0 0 0 48 27 0 0 0 0 27
0 19 0 0 0 0 0 0 31 11 13 0 0 10 0 0 0 0 1 0 0 24 45 0 0 0 6 0 0 0
49 23 47 0 0 0 0 0 0 0 0 0 28 47 0 10 1 0 0 0 20 46 6 0 42 0 0 0 0 0 0 0 0 0 0 0 0 42 9 0 5 44 0 0 11 31 0 0 0 47 0 0 0 0 0 47 0 0 0 0 0 0 7 0 0 17 0 0 0 0 3
0 42 16 11 0 19 45 0 0 14 22 0 0 0 0 0 0 0 0 0 0 0 0 14 0 0 37 6 49 43 15
0 30 11 43 17 0 15 0 0 28 0 21 0 0 12 0 33 42 0 0 0 17 0 27 0 0 0 45 9 0 0 0 0 0 0 0 15 0 35 0 24 0 0 0 1 1 0 0 8 26 3 0 0 0 17 29 0 0 12 0 0 20 12 31 15 0
0 0 0 0 0 0 6 36 2 0 8 0 25 0 3 0 0 0 0 0 0 0 0 0 0 9 31 0 0 40 0 33 28 0 41
31 0 0 37 0 0 0 0 0 15 0 0 8 0 0 0 34 0 12 0 0 34 0 0 10 29 0 0 0 0 0 30 0 0 0 33 24 0 0 0 41 0 0 0 16 38 22 34 23 32 0 4 0 0 8 0 0 42 0 0 15 0 0 0 48 0 0 0
0 0 33 0 0 0 26 40 0 4 0 22 0 0 5 0 28 0 0 0 0 0 2 31 0 31 0 12 5 0 34 0
4 0 0 3 0 0 42 0 29 17 33 0 31 0 0 14 0 0 37 0 25 0 0 35 0 2 0 0 0 13 0 0 0 0 0 8 2 0 15 0 20 43 3 0 0 0 0 0 2 0 40 31 0 0 0 2 0 13 0 0 18 0 0 15 0 22 0 46
0 0 0 20 0 0 0 32 0 0 0 0 0 0 47 28 0 23 3 0 0 34 0 9 0 17 19 0 0 0 37 0 43
19 0 0 0 0 0 0 0 0 0 0 0 0 44 0 0 20 33 0 0 0 27 34 11 0 10 0 0 0 0 0 18 9 13 24 0 0 0 0 30 3 29 21 0 0 0 0 44 0 0 0 25 39 45 48 16 0 0 0 0 0 16 0 0 28 49 0 0
0 18 44 0 0 43 38 0 0 0 0 0 10 0 0 0 7 0 0 0 14 8 0 7 0 24 29 37 43 13 0 0 0
5 0 0 0 0 0 0 0 0 0 0 0 0 11 21 0 0 0 0 21 0 0 11 0 37 0 0 0 1 0 0 0 27 0 5 5 0 31 4 42 0 0 0 0 0 0 0 13 0 0 20 0 0 0 34 24 12 0 43 0 25 0 4 46 18 0 19 0 0
0 15 0 0 0 0 22 0 0 29 26 48 49 0 1 33 0 0 0 0 30 42 46 37 0 17 0 0 0 0 0
41 0 7 42 34 0 0 0 0 0 0 0 0 17 0 0 25 0 0 38 23 0 0 0 0 0 4 45 38 0 0 0 0 28 0 7 0 0 17 0 12 0 0 47 0 42 0 0 0 0 25 49 0 41 0 0 0 0 18 0 0 20 17 19 42 8 17
16 0 47 34 11 35 0 26 2 0 4 1 31 0 0 0 0 12 19 0 0 0 29 0 0 0 0 30 0 0 0 0 15
24 33 37 47 6 0 0 0 0 25 0 0 31 0 0 40 0 0 0 22 0 26 15 0 0 0 0 0 27 0 0 8 0 0 6 33 0 2 0 49 5 0 25 26 0 0 0 0 4 40 0 0 0 32 0 0 0 0 18 0 47 0 0 0 0 0 0 0 0
13 0 17 30 7 0 0 0 0 14 0 49 8 29 0 0 13 0 0 34 0 0 12 0 0 0 0 30 0 0 7
46 31 26 0 0 4 0 0 16 0 0 0 0 0 0 39 0 31 0 0 0 19 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 39 0 0 9 0 0 0 0 36 42 13 0 0 26 49 0 0 0 26 0 3 0 14 21 0 0 0 0 0 0 0 0 0
7 0 0 0 0 0 0 0 10 12 0 0 0 43 0 0 0 0 0 24 46 28 47 0 37 0 0 0 38 0
0 0 44 0 0 0 8 21 0 0 6 0 0 29 0 0 0 0 11 0 0 0 41 0 18 0 13 0 19 0 0 0 0 39 14 0 0 0 0 0 0 6 0 35 0 40 0 17 1 0 35 46 0 0 8 16 38 0 0 0 0 45 0 0 0 11 0 3 0
0 0 0 0 0 47 6 0 0 0 0 0 46 0 0 0 16 0 0 31 0 21 0 35 0 0 29 0 0 0 0
0 0 0 35 5 0 0 0 0 1 11 0 0 16 0 6 0 0 0 0 0 0 18 22 0 44 0 0 0 0 0 0 0 27 0 0 0 0 44 26 29 0 20 33 0 0 36 38 0 0 0 0 46 38 0 8 28 43 9 0 0 0 0 6 0 49 0 0 0 0 0
0 0 0 0 0 34 0 25 43 31 5 0 0 10 26 0 13 0 0 41 0 0 10 0 0 17 0 0 0 0 0
0 0 0 6 0 39 1 0 0 0 0 0 0 35 0 0 46 0 0 36 0 0 11 0 0 0 21 0 21 20 0 0 13 0 0 2 38 0 0 0 11 24 0 8 0 34 8 0 0 23 0 43 14 0 0 0 37 0 0 27 0 0 0 0 37 0 3 0 0
48 5 42 0 0 25 4 49 0 21 0 0 21 33 0 2 1 17 0 7 17 0 0 0 0 0 0 13 0 21 0

0 0 0 0 18 0 0 0 17 18 40 35 0 0 0 0 7 0 2 0 18 0 14 9 33 0 25 0 34 4 40 45 30 1 4 0 48 0 0 0 0 49 2 38 27 0 27 0 34 16 12 0 1 36 1 0 0 0 0 0 0 44 33 0 28 0 0 18 0 0 48 0 0 24 17 18 29 25 31 4 0 7 0 0 0 0 44 44 0 0 33 0 0 0 0 23 0 39 10 0

0 0 0 0 17 0 0 27 18 0 19 0 0 0 29 41 40 15 21 0 0 10 0 0 46 13 0 0 0 44 0 39 15 0 28 20 0 0 0 30 0 0 0 0 35 0 3 0 40 0 0 0 18 21 0 15 0 31 0 0 20 0 0 0 0 0 0 0 0 8 0 40 15 0 0 0 34 0 0 0 10 0 0 9 0 1 0 0 0 13 29 0 40 45 0 0 0 0 0 0

0 20 0 31 29 0 35 0 0 48 30 0 0 0 0 0 1 0 0 0 0 7 39 0 47 0 1 0 0 0 47 0 0 0 0 0 0 0 8 42 0 19 0 0 37 0 0 0 0 41 0 7 0 0 0 0 0 0 0 0 12 0 28 0 0 46 8 49 46 0 0 41 0 0 38 41 0 13 13 0 0 0 0 0 0 0 0 0 0 33 0 0 0 0 34 0 0 1 0 0 0 28 19 0

0 2 0 0 0 0 23 0 42 34 12 0 0 0 5 33 0 0 0 0 0 0 0 0 0 0 46 48 0 17 0 0 0 27 0 0 0 8 19 0 0 10 0 15 0 0 0 49 0 0 0 0 0 25 28 0 36 43 0 0 0 0 0 33 26 0 0 0 0 19 32 0 0 0 0 41 0 0 2 0 0 0 0 0 42 0 0 39 44 0 0 0 0 0 41 0 0 0 0 37 0 0

0 29 0 12 19 0 0 0 47 39 40 0 0 34 20 0 0 0 0 0 44 0 0 0 42 0 0 25 0 28 0 0 0 0 0 17 22 0 0 0 0 0 0 0 0 21 23 0 18 0 35 0 0 0 0 36 0 0 0 0 0 35 3 18 19 0 9 4 0 3 0 0 0 29 1 0 0 0 0 20 43 29 6 0 0 0 0 44 32 0 14 0 10 0 0 0 0 0 33 20

0 19 0 0 0 24 0 41 0 0 6 0 49 49 0 6 0 0 0 23 29 0 0 19 38 0 0 0 0 7 32 0 0 0 0 0 6 0 45 19 23 0 29 0 7 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0 0 7 0 34 37 0 0 0 0 0 47 5 0 0 0 0 12 2 16 40 5 33 0 0 0 30 0 0 0 0 0 20 46 48 0 0 0 23

0 0 0 9 43 0 0 0 10 0 0 47 41 0 0 0 18 29 0 0 46 0 0 0 0 48 23 27 0 0 0 0 1 0 0 0 21 46 40 16 7 9 20 0 0 0 0 0 10 12 0 10 0 0 0 0 22 0 0 44 0 36 0 8 0 46 0 0 0 0 11 0 0 0 0 0 36 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 39 0 0 0

0 7 20 0 1 0 0 0 0 0 5 17 0 0 0 20 0 0 37 0 0 0 0 0 0 4 27 30 44 34 0 47 0 0 0 0 30 12 0 0 0 0 0 14 0 7 0 0 16 0 0 49 0 0 0 0 46 0 0 0 0 0 0 0 0 0 30 0 0 33 27 7 19 0 23 3 0 0 19 49 30 0 0 0 41 0 31 0 22 0 0 0 0 0 0 37 0 20 46 0 0

16 44 0 0 0 0 0 1 34 0 23 0 0 30 0 0 0 33 0 0 0 0 0 35 0 0 0 0 0 0 0 23 26 39 23 0 0 0 0 5 0 44 0 38 0 0 15 0 0 0 24 0 0 0 0 0 0 0 0 0 27 0 0 0 0 45 46 0 0 16 0 32 0 0 0 0 0 0 0 0 0 0 0 22 7 0 25 0 0 19 27 0 0 33 47 0 48 0 32 0 0 27

0 0 41 0 0 0 0 12 0 0 0 0 35 49 5 0 22 4 0 0 0 0 0 0 32 0 0 0 0 1 49 0 26 34 0 0 0 0 19 48 27 9 0 0 0 31 0 47 0 0 0 28 0 0 2 0 33 8 0 43 0 9 0 0 14 0 24 0 0 0 46 0 35 0 30 15 0 0 35 0 0 0 0 0 8 20 0 0 0 0 43 0 0 0 0 0 0 14 0 0

0 47 0 0 0 41 0 0 9 37 45 0 0 2 43 0 0 0 0 0 11 0 0 0 0 24 33 10 0 12 24 0 48 9 0 2 0 32 49 17 0 38 27 0 26 8 0 0 0 0 0 0 0 5 0 0 27 0 0 0 8 0 42 0 12 0 0 0 41 22 0 44 0 0 0 0 25 10 0 0 0 0 0 27 27 37 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 2 0 0 41 7 2 0 0 23 39 4 48 0 0 0 0 0 32 0 31 0 0 17 0 0 0 43 0 4 0 0 0 29 0 25 0 3 0 19 22 0 0 26 0 0 0 12 46 0 0 0 24 0 31 0 0 0 16 0 27 0 0 27 0 0 0 18 0 0 0 39 14 0 43 0 0 18 0 32 0 0 0 0 0 0 0 14 47 0 17 0 0 0 41

0 27 14 0 42 25 0 35 0 0 0 0 11 0 48 0 38 29 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 0 0 30 0 0 0 0 6 0 0 0 0 37 0 0 34 49 14 29 0 0 0 32 0 0 0 0 0 0 7 0 43 0 0 0 0 0 0 0 0 24 28 0 27 0 37 13 44 0 0 0 0 0 4 0 26 0 28 16 0 0 0 0 0 0 0 0 0

42 37 33 7 0 0 0 0 0 0 0 0 0 35 0 0 0 0 30 36 0 0 1 0 25 40 15 22 46 0 40 32 27 0 0 0 0 0 11 0 41 28 12 32 0 15 0 0 0 0 6 17 37 0 0 0 36 0 0 0 0 0 0 0 0 0 24 0 0 19 26 0 8 0 0 6 0 0 45 0 32 5 14 23 0 0 0 4 0 0 0 0 0 37 0 1 0 43 0 8

0 0 27 0 0 0 8 0 0 11 0 0 0 22 0 46 0 0 31 0 0 19 0 0 31 0 0 0 0 43 5 42 0 36 0 0 0 24 27 0 14 0 0 32 0 0 0 0 45 0 25 46 11 25 0 0 0 0 0 0 0 44 0 10 0 0 0 0 0 0 0 18 0 26 0 0 0 0 10 0 11 0 0 0 0 0 8 30 0 0 0 0 25 0 0 0 26 0 41 26 39

0 0 0 0 0 0 0 0 0 1 0 16 0 0 0 0 12 14 16 0 0 0 0 0 46 0 33 39 0 1 49 1 0 0 0 28 0 10 17 0 0 46 0 0 0 0 0 20 0 0 0 17 25 0 6 0 0 0 18 0 41 0 0 0 1 25 0 0 36 8 0 1 47 24 0 0 0 0 34 0 0 0 38 0 0 0 0 4 24 0 0 23 42 0 1 0 30 29 0 44

0 0 0 0 0 0 20 0 0 0 0 0 0 39 0 27 0 0 0 0 0 27 0 38 0 0 0 15 0 49 6 36 0 40 0 9 0 46 15 24 0 0 0 0 0 8 0 0 47 23 0 35 19 30 0 0 15 0 14 0 0 4 0 0 14 8 0 25 0 25 48 45 0 35 6 0 0 21 34 0 0 0 0 48 33 20 17 0 0 35 0 0 32 0 0 0 47 0 0 0