RIKIAVIMO ALGORITMŲ TYRIMAS

Ingrida Orvidaitė, Marijus Bernotas
Šiaulių universitetas, Technologijos fakultetas

Įvadas

Tyrimo tikslas – išanalizuoti ir palyginti rikiavimo algoritmų kiekybines charakteristikas.

Greitesniam darbui su dideliu kiekiu duomenų, būtina juos surikiuoti. Esant daug duomenų, didelę reikšmę įgauna rikiavimo algoritmo sudėtingumas – atlikimo greičio priklausomybė nuo duomenų kiekio. Algoritmų analizėje duomenų rikiavimo problema laikoma pačia svarbiausia, nes tai viena dažniausiai pasitaikančių operacijų programuojant. Yra daug rikiavimo algoritmų, veikiančių skirtingais principais, ir jie klasifikuojami pagal nevienodus parametrus, pavyzdžiui, pagal naudojamą atmintį, pagal reikiamos atminties kiekį, stabilumą, sudėtingumą. Be to, grupuojami pagal rikiavimo metodus, iš kurių galima išskirti tris pagrindines grupes: išrinkimo (*selection sorts*), įterpimo (*insertion sorts*) ir sukeitimo (*exchange sorts*). Bet yra ir kitų, rečiau vartojamų rikiavimo metodų grupių, tokių kaip sąlajos (*merge sorts*), pasiskirstymo (*distribution sorts*), sutapimo (*concurrent sorts*) ir t. t. Efektyvus rikiavimo algoritmo pasirinkimas gali turėti lemiamos įtakos programos vykdymo spartai dėl duomenų kiekio didėjimo.

Uždaviniai: Sukurti programą, realizuojančią šešis rikiavimo algoritmus; palyginti jų kiekybines charakteristikas (palyginimų kiekis, sukeitimų kiekis, rikiavimo laikas); pateikti rikiavimo algoritmų taikymo rekomendacijas.

Tyrimo metodai: srities modeliavimas, funkcinė analizė, testavimas, loginis apibendrinimas.

Rikiavimo algoritmų klasifikavimo būdai

Klasifikavimas pagal naudojamą atmintį. Rikiavimo algoritmai skirstomi į vidinio rikiavimo ir išorinio rikiavimo algoritmus priklausomai nuo to, ar naudoja tik vidinę kompiuterio atmintį, ar jiems reikia ir išorinės (antrinės). Išorinio rikiavimo algoritmai naudojami tada, kai duomenys vienu metu negali būti išdėstyti atmintyje [1].

Vidinio rikiavimo algoritmai turi dvi pagrindines operacijas: raktų palyginimą (*compare*) ir elementų keitimą vietomis (*swap*) [3].

Klasifikavimas pagal reikiamos atminties kiekį. Rikiavimo algoritmai gali būti skirstomi į grupes [1] pagal papildomos atminties naudojimo kiekį:

- nenaudoja visiškai,
- naudoja papildomai tik atmintį, reikalingą rodyklėms išdėstyti,
- naudoja papildomai tiek pat atminties, kiek yra duomenų.

Papildomą atmintį naudoja algoritmai, kurie remiasi rekursijos principu arba paradigma „skaldyk ir valdyk“ [2]. Netgi algoritmai, naudojantys papildomą atmintį, jos sunaudoja nevienodai, priklausomai, kiek yra rikiuojamų elementų. Pavyzdžiui, *Merge sort* algoritmas naudoja papildomą atmintį, kurios dydis proporcingas duomenų kiekiui, tačiau gali būti pasiektas pilnų duomenų kiekio vietos poreikis, *Heapsort* algoritmas reikalauja tik fiksuoto papildomos atminties kiekio, o *Quicksort* algoritmui papildoma atmintis beveik nereikalinga.

Klasifikavimas pagal stabilumą. Stabilūs algoritmai nekeičia lygių elementų tvarkos, o nestabilūs algoritmai to negarantuoja. Realizuojant rikiavimo algoritmą, kai kurie elementai sukeičiami vietomis. Jeigu elementai vienodi, tai jų tarpusavio išsidėstymas nesvarbus. Tačiau kai kada reikia garantuoti, kad algoritmas nepakeičia tokių elementų tarpusavio padėties. Duomenų rikiavimo algoritmas vadinamas stabiliu, jei jis išsaugo santykines lygių elementų vietas vienas kito atžvilgiu [6].

Algoritmai *Bubble sort*, *Insertion sort*, *Gnome sort*, *Merge sort* yra laikomi stabiliais. Šiame tyrime iš jų naudojami *Bubble sort*, *Insertion sort*, *Gnome sort.* Jie nekeičia lygių elementų tvarkos ir išsaugo santykines lygių elementų vietas vienas kito atžvilgiu. Iš nestabilių algoritmų galima paminėti *Quicksort* (jis veikia „skaldyk ir valdyk“ principu), *Heapsort*, *Selection sort*, *Shell sort*. Šioje analizėje naudojami *Selection sort* ir *Shell sort* algoritmai, kurie negarantuoja stabilios lygių elementų tvarkos ir gali neišsaugoti lygių elementų vietos vienas kito atžvilgiu.

Klasifikavimas pagal sudėtingumą. Esant dideliems duomenų kiekiams, labai svarbu ir paties rikiavimo algoritmo sudėtingumas – atlikimo greičio priklausomybė nuo duomenų kiekio. Paprasčiausių rikiavimo algoritmų (Selection sort, Insertion sort, Bubble sort) sudėtingumas yra kvadratinis (žymima $O(N^2)$, kur N – rikiuojamų duomenų kiekis; „O“ žymuo: funkcija g(N) yra vadinama O(f(N)), jei egzistuoja konstantos c_0 ir N_0, tokios, kad $0 < g(N) < c_0 f(N)$ visiems $N > N_0$) [2]. Dažnai greičiausiu laikomo Quicksort algoritmo sudėtingumas daugeliu

atveju yra O(N lg N), tačiau, rikiuojant beveik surikiuotus duomenis, šio algoritmo sudėtingumas siekia $O(N^2)$ [5].

Algoritmo sudėtingumas svarbus tik esant dideliam duomenų kiekiui. Jei jis nedidelis, dažniausiai visiškai nesvarbu, kiek mikrosekundžių bus vykdomas rikiavimas, tačiau, esant didesniems duomenų kiekiams, šis skirtumas yra milžiniškas. Algoritmų sudėtingumas priklauso ir nuo duomenų savybių. Dažniausiai matuojamas algoritmų sudėtingumas *vidutiniu atveju*, dažnai – *blogiausiu atveju* ir tik kartais – *geriausiu*. Tas pats algoritmas, būdamas labai greitas *geriausiu atveju*, gali būti labai lėtas *vidutiniu* ar *blogiausiu atveju* [5]. Geriausias atvejis toks, kai pradinė elementų aibė jau surikiuota, vidutinis atvejis toks, kai elementų pradinio išsidėstymo visų variantų tikimybė yra vienoda, blogiausias atvejis toks, kai pradiniame masyve elementai yra išdėstyti mažėjančia (atvirkštine nei reikia surikiuoti) tvarka [6].

Rekursiškumas. Daug algoritmų yra pagrįsti didelių uždavinių pasikartojančiu skaidymu į kelis mažesnius, kuriuos išsprendus išsprendžiamas ir visas uždavinys. Tokiam uždavinių skaidymui tinkamos metodikos pirmiausia paremtos rekursyviais arba rekurentiniais metodais. Nagrinėjant rekurentinį skaidymą, tikslinga apžvelgti pagrindinius tokių algoritmų analizės metodus, pagrįstus kelių standartinių formulių sprendimais [2].

Pagrindinę algoritmo skaidymo į dalis schemą apibrėžia principas, kad algoritmo pakartotiniai skaidymai į dalis tiesiogiai atsispindi jo analizėje. Tokių algoritmų vykdymo laikas priklauso nuo paprogramių dydžio ir skaičiaus bei nuo dekompozicijos laiko. Algoritmo su N įvedimo duomenų kiekiu vykdymo laiko priklausomybė nuo mažesnio įvedimo duomenų kiekio vykdymo laiko yra lengvai apskaičiuojama rekurentinėmis formulėmis. Rekursija yra naudinga todėl, kad ji dažnai leidžia sudėtingus algoritmus išreikšti glaustesne forma, išsaugant programos efektyvumą [2].

Pagrindiniai rekursinės programos bruožai: ji kreipiasi pati į save (naudodama mažesnes kintamųjų reikšmes), ji turi užbaigimo sąlygą, kuri tiesiogiai skaičiuoja savo rezultatus. Norint įsitikinti (faktiškai įrodyti matematiškai), kad rekursyvi programa veikia teisingai, dažniausiai taikomas matematinės indukcijos metodas, pavyzdžiui, faktorialas [2].

Klasifikavimas pagal rikiavimo metodą

Rikiavimo algoritmai yra trijų pagrindinių tipų: *išrinkimo*, *įterpimo* bei *sukeitimo*.

Rikiavimas išrinkimu – paprasčiausias būdas. Masyve randamas mažiausias, jei rikiuojama didėjimo tvarka, elementas ir sukeičiamas su pirmuoju masyvo elementu. Toliau vėl ieškomas mažiausias masyvo elementas, pradedant nuo antrojo, ir sukeičiamas vietomis su antruoju. Procesas kartojamas su visais masyvo elementais, kol surikiuojamas visas masyvas [4, 7]. Pagrindiniai rikiavimai: *Selection sort, Double Selection sort, Heapsort.*

Rikiavimas įterpimu. Bet kuriuo rikiavimo įterpimu momentu susidaro dvi dalys: vienoje yra jau surikiuoti elementai, kitoje – ne. Iš pradžių surikiuotą masyvo dalį sudaro tik pirmasis masyvo elementas. Imamas elementas iš nerikiuotos masyvo dalies ir įterpiamas į surikiuotąją. Norint elementą įterpti, reikia surasti jo vietą surikiuotoje dalyje ir visus didesnius už įterpiamąjį elementus pastumti per vieną poziciją. Įterpimo vietos paieška pradedama nuo surikiuotos dalies pabaigos, vienu metu ieškant ir pastumiant [4, 7]. Pagrindiniai rikiavimai: *Insertion sort, Shell sort.*

Rikiavimo sukeitimu algoritmai pagrįsti dviejų masyvo elementų, dažniausiai gretimų, sukeitimu vietomis, kai vien pagal jų reikšmes sprendžiama, ar reikia juos sukeisti. Šios grupės algoritmuose sprendžiami du uždaviniai: kokiu būdu parinkti elementų poras ir kaip nustatyti, ar masyvas jau surikiuotas. Didžiausias rikiavimo sukeitimu algoritmo trūkumas – „maži“ elementai, atliekant šį algoritmą, aukštyn pastumiami per vieną vietą [4, 7]. Pagrindiniai rikiavimai: *Bubble sort, Gnome sort, Quicksort.*

Yra ir šiems tipams nepriskiriamų rikiavimų, tokių kaip: *Merge sort, Bucket sort, Counting sort, Radix sort* ir kitų.

Tyrimo metu sukurtos programos analizei atlikti

Duomenų generavimo programa skirta atsitiktinių duomenų failui (= rinkmenai) generuoti. Programa generuoja trijų tipų duomenų failus (= rinkmenas): INT tipo (sveikųjų skaičių), FLOAT tipo (slankaus kablelio skaičių), ir CHAR tipo (generuojančio tik raidyną). Programoje nurodomas reikiamas reikšmių kiekis. Skaičiams nurodomas ir intervalas (apatinė bei viršutinė intervalo reikšmės), iš kurio atsitiktiniu būdu renkamos reikšmės.

Atskirų algoritmų veikimo demonstravimo programa skirta algoritmams po vieną analizuoti, stebėti, kaip keičiasi kuriuo nors metodu (vienu iš šešių) rikiuojami duomenys. Ši programa ne tik skaičiuoja palyginimų ir sukeitimų kiekius ir rikiavimo laiką, kuris skaičiuojamas mikrosekundėmis, bet ir išrašo elementų sekas po kiekvieno sukeitimo.

Duomenų analizės charakteristikų tyrimo programa. Jai pateikiami INT, FLOAT arba CHAR tipo duomenų failai (= rinkmenos). Programa juos surikiuoja visais šešiais analizuojamais būdais ir grąžina rikiavimo charakteristikas: palyginimų kiekį, sukeitimų kiekį bei rikiavimo laiką. Charakteristikos pateikiamos .txt failais (= rinkmenomis).

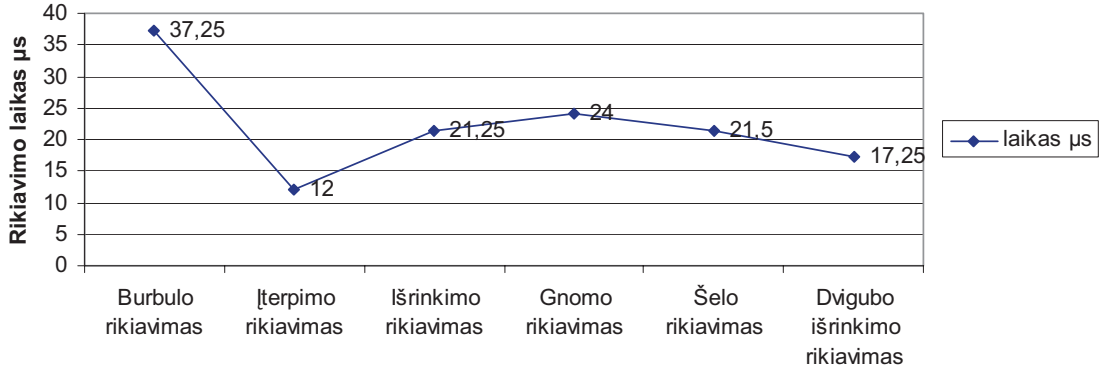
Tyrimo rezultatai

Rikiavimo algoritmų kiekybinių charakteristikų palyginimas

Analizuojant rikiavimo algoritmus su INT tipo duomenimis (50 reikšmių), daugiausia palyginimų – net 1984,5, atliko *Burbulo* rikiavimas, po jo – *Gnomo* rikiavimas, atlikęs 1229 palyginimus, ir *Išrinkimo* rikiavimas, atlikęs 1225 palyginimus. Mažiausia palyginimų atliko *Dvigubo išrinkimo* rikiavimas, tik 625, ir *Įterpimo* rikiavimas – 641 palyginimas.

Daugiausia sukeitimų atliko *Burbulo*, *Įterpimo* ir *Gnomo* rikiavimai – visi trys po 592 sukeitimus. Mažiausia sukeitimų atliko *Išrinkimo* rikiavimas – tik 44 sukeitimus, ir *Dvigubo išrinkimo* rikiavimas – tik 50 sukeitimų.

Rikiavimo laikas pasiskirstęs panašiai kaip palyginimų kiekis, tačiau ne taip, kaip tikėtasi, jis nėra tiesiogiai priklausomas nuo atlikto operacijų *palyginimai* + *sukeitimai* kiekio (1 pav.).

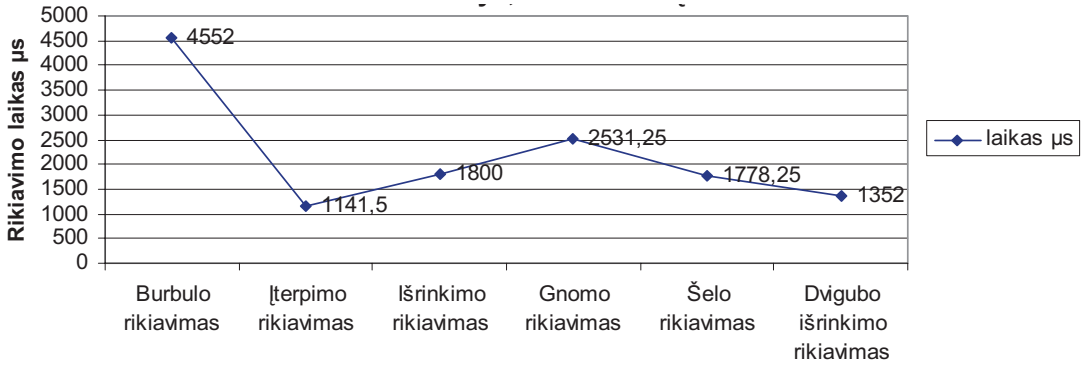


1 pav. Skirtingų algoritmų rikiavimo laiko sklaida. INT tipo duomenys, 50 reikšmių

Analizuojant rikiavimo algoritmus su INT tipo duomenimis (500 reikšmių), galima pastebėti, kad palyginimų ir sukeitimų kiekių kreivės išsidėsčiusios panašiai kaip ir rikiuojant to paties tipo 50 reikšmių. Daugiausia palyginimų – net 238023 – atliko *Burbulo* rikiavimas, po jo – *Gnomo* rikiavimas, atlikęs 125467,5 palyginimo ir *Išrinkimo* rikiavimas, atlikęs 124750 palyginimų. Mažiausia palyginimų atliko *Dvigubo išrinkimo* rikiavimas, tik 62500, ir *Įterpimo* rikiavimas – 62985,75 palyginimo.

Daugiausia sukeitimų atliko *Burbulo*, *Įterpimo* ir *Gnomo* rikiavimai – visi trys po 62486,75 sukeitimo. Mažiausia sukeitimų atliko *Išrinkimo* rikiavimas – tik 494,25 sukeitimo, ir *Dvigubo išrinkimo* rikiavimas – tik 500 sukeitimų.

Rikiavimo laikas pasiskirstęs panašiai kaip ir rikiuojant to paties tipo 50 reikšmių (2 pav.).

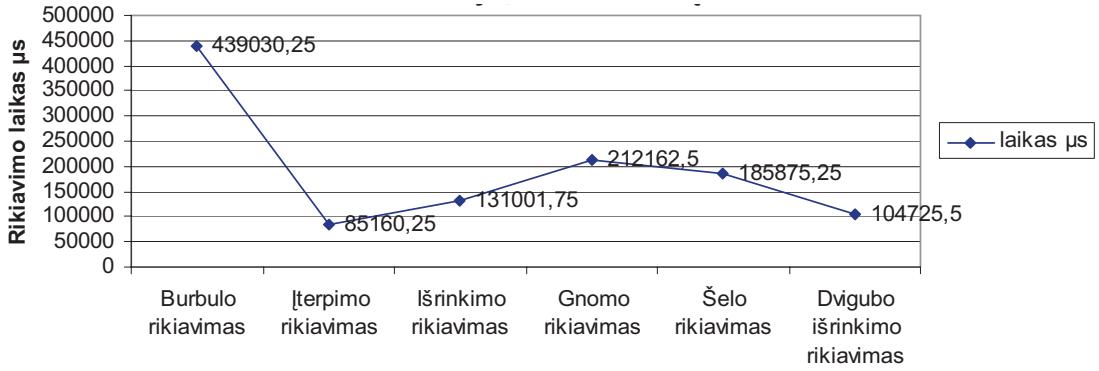


2 pav. Skirtingų algoritmų rikiavimo laiko sklaida. INT tipo duomenys, 500 reikšmių

Analizuojant rikiavimo algoritmus su INT tipo duomenimis (5000 reikšmių), nustatyta, kad palyginimų ir sukeitimų kiekių kreivės išsidėsčiusios panašiai kaip ir ankstesniuose to paties tipo grafikuose, tik *Šelo* rikiavimo palyginimų kreivė šiek tiek kilstelėjusi aukštyn, lyginant su mažesnių duomenų kiekių grafikais. Daugiausia palyginimų – net 24518845,25 – atliko *Burbulo* rikiavimas, po jo – *Gnomo* rikiavimas, atlikęs 12604574 palyginimus, ir *Šelo* rikiavimas (nebe *Išrinkimo*), atlikęs 12563743,75 palyginimo. *Išrinkimo* rikiavimas atliko 12497500 palyginimų. Mažiausia palyginimų atliko *Dvigubo išrinkimo* rikiavimas, tik 6250000, ir *Įterpimo* rikiavimas – 6304789,5 palyginimo.

Daugiausia sukeitimų atliko *Burbulo*, *Įterpimo* ir *Gnomo* rikiavimai – visi trys po 6299790,5 sukeitimo. Mažiausia sukeitimų atliko *Išrinkimo* rikiavimas – tik 4989,75 sukeitimo, ir *Dvigubo išrinkimo* rikiavimas – tik 5000 sukeitimų.

Rikiavimo laikas pasiskirstęs panašiai kaip ir ankstesniuose to paties tipo grafikuose, tik *Šelo* rikiavimo laiko kreivė šiek tiek kilstelėjusi aukštyn, lyginant su mažesnių duomenų kiekių grafikais (3 pav.).

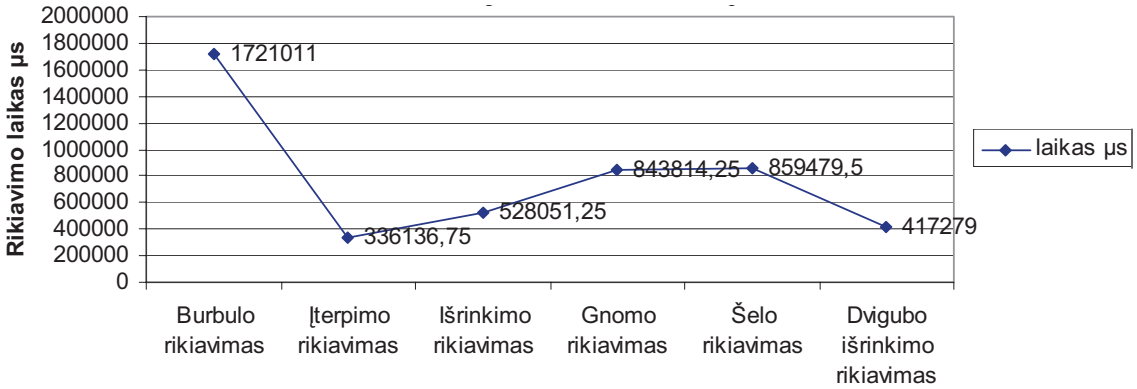


3 pav. Skirtingų algoritmų rikiavimo laiko sklaida. INT tipo duomenys, 5000 reikšmių

Analizuojant rikiavimo algoritmus su INT tipo duomenimis (10000 reikšmių), išaiškinta, kad *Šelo* rikiavimo palyginimų kreivė dar labiau kilstelėjusi aukštyn, lyginant su praeitu palyginimų bei sukeitimų kiekių pasiskirstymo grafiku (to paties tipo duomenų 5000 reikšmių). Daugiausia palyginimų – net 98915107,5 – atliko *Burbulo* rikiavimas, po jo – *Šelo* rikiavimas (nebe *Gnomo*), atlikęs 57839223,5 palyginimo ir *Gnomo* rikiavimas, atlikęs 50115988 palyginimus. *Išrinkimo* rikiavimas atliko 49995000 palyginimų. Mažiausia palyginimų atliko *Dvigubo išrinkimo* rikiavimas, tik 25000000, ir *Įterpimo* rikiavimas – 25062998,75 palyginimo.

Daugiausia sukeitimų atliko *Burbulo*, *Įterpimo* ir *Gnomo* rikiavimai – visi trys po 25052999,75 sukeitimo. Mažiausia sukeitimų atliko *Išrinkimo* rikiavimas – tik 9990 sukeitimų, ir *Dvigubo išrinkimo* rikiavimas – tik 10000 sukeitimų.

Rikiavimo laikas pasiskirstęs panašiai kaip ir ankstesniuose to paties tipo grafikuose, tik *Šelo* rikiavimo laiko kreivė dar labiau kilstelėjusi aukštyn, lyginant su praeitu rikiavimo laiko pasiskirstymo grafiku (to paties tipo duomenų 5000 reikšmių) (4 pav.).



4 pav. Skirtingų algoritmų rikiavimo laiko sklaida. INT tipo duomenys, 10000 reikšmių

Analizuojant rikiavimo algoritmus su FLOAT tipo duomenimis (50 reikšmių), daugiausia palyginimų – net 1947,75 – atliko *Burbulo* rikiavimas, po jo – *Išrinkimo* rikiavimas, atlikęs 1225 palyginimus ir *Gnomo* rikiavimas, atlikęs 1194,75 palyginimo. Mažiausia palyginimų atliko *Įterpimo* rikiavimas, tik 619,25, ir *Dvigubo išrinkimo* rikiavimas – 625 palyginimus. Pastebėtina, kad rikiavimų išsidėstymas nesutampa su INT tipo duomenų analize, kai naudojamas tas pats reikšmių kiekis. INT tipo duomenų analizėje, kai analizuojama su 50 reikšmių duomenimis, mažiausiai palyginimų atliko *Dvigubo išrinkimo* rikiavimas; *Įterpimo* rikiavimas buvo antras.

Daugiausia sukeitimų atliko *Burbulo*, *Įterpimo* ir *Gnomo* rikiavimai – visi trys po 573,75 sukeitimo. Mažiausiai sukeitimų atliko *Išrinkimo* rikiavimas – tik 45,5 sukeitimo, ir *Dvigubo išrinkimo* rikiavimas – tik 50 sukeitimų. Pastebėta, kad *Dvigubo išrinkimo* rikiavimo palyginimų ir sukeitimų kiekis sutampa su INT tipo to paties reikšmių kiekio palyginimų ir sukeitimų kiekiu.

Rikiavimo laikas pasiskirstęs panašiai kaip palyginimų kiekis. Kaip ir INT tipo duomenų analizėje, rikiavimo laikas nėra tiesiogiai priklausomas nuo atlikto operacijų *palyginimai + sukeitimai* kiekio. Ilgiausiai 50 reikšmių FLOAT tipo duomenis rikiavo

Burbulo algoritmas, net 39,5 mikrosekundės. Trumpiausiai – *Įterpimo* rikiavimas, surikiavęs duomenis per 14 mikrosekundžių. *Šelo* duomenis surikiavo per 19,75 mikrosekundės, *Dvigubo išrinkimo* algoritmas – per 20 mikrosekundžių, *Išrinkimo* rikiavimas – per 22 mikrosekundes.

Analizuojant rikiavimo algoritmus su FLOAT tipo duomenimis (500 reikšmių), buvo nustatyta, kad palyginimų ir sukeitimų kiekių kreivės išsidėsčiusios panašiai kaip ir rikiuojant to paties reikšmių kiekio INT tipo duomenis. Daugiausia palyginimų – net 236900,25, atliko *Burbulo* rikiavimas, po jo – *Gnomo* rikiavimas, atlikęs 127011,5 palyginimo, *Išrinkimo* rikiavimas, atlikęs 124750 palyginimų, ir *Šelo* rikiavimas, atlikęs 119642,25 palyginimo. Mažiausiai palyginimų, kaip ir INT tipo duomenų analizėje, atliko *Dvigubo išrinkimo* rikiavimas, tik 62500, ir *Įterpimo* rikiavimas – 63688,25 palyginimo (*Įterpimo* rikiavimo palyginimų kiekis truputį skiriasi nuo INT tipo duomenų).

Daugiausia sukeitimų atliko *Burbulo*, *Įterpimo* ir *Gnomo* rikiavimai – visi trys po 63259 sukeitimo. Mažiausia sukeitimų atliko *Išrinkimo* rikiavimas – tik 494,5 sukeitimo, ir *Dvigubo išrinkimo* rikiavimas – tik 500 sukeitimų.

Rikiavimo laikas pasiskirstęs panašiai kaip ir rikiuojant to paties reikšmių kiekio INT tipo duomenis. Ilgiausiai 500 reikšmių FLOAT tipo duomenis rikiavo *Burbulo* algoritmas, net 6481,25 mikrosekundes. Trumpiausiai – *Įterpimo* rikiavimas, išdėstęs duomenis per 909,5 mikrosekundės, po jo – *Dvigubo išrinkimo* algoritmas, surikiavęs per 1228 mikrosekundes. Nebe taip, kaip ankstesniuose šio duomenų tipo grafikuose, *Šelo* algoritmo rikiavimo laikas atsilieka nuo *Išrinkimo* algoritmo rikiavimo laiko. *Šelo* rikiavimo laikas yra net 2096,25 mikrosekundės, o *Išrinkimo* rikiavimo laikas yra 1416,25 mikrosekundės. Skirtumas net 680 mikrosekundžių. INT tipo analizėje su tokiu pat duomenų kiekiu, priešingai nei šioje FLOAT tipo duomenų analizėje, *Šelo* algoritmas rikiuoja duomenis greičiau negu *Išrinkimo* algoritmas.

Analizuojant rikiavimo algoritmus su FLOAT tipo duomenimis (5000 reikšmių), pastebėta, kad palyginimų ir sukeitimų kiekių kreivės išsidėsčiusios panašiai kaip ir ankstesniuose to paties tipo grafikuose, tik *Šelo* rikiavimo palyginimų kreivė, kaip ir INT tipo duomenų grafike, šiek tiek kilstelėjusi aukštyn, lyginant su mažesnių duomenų kiekių grafikais. Daugiausia palyginimų – net 24761296,75, atliko *Burbulo* rikiavimas, po jo – *Šelo* (nebe *Gnomo*) rikiavimas, atlikęs 13313593,75 palyginimo ir *Gnomo* rikiavimas, atlikęs 12636950,5 palyginimo. *Išrinkimo* rikiavimas atliko 12497500 palyginimų. Mažiausiai palyginimų atliko *Dvigubo išrinkimo* rikiavimas, tik 6250000, ir *Įterpimo* rikiavimas – 6320385 palyginimus.

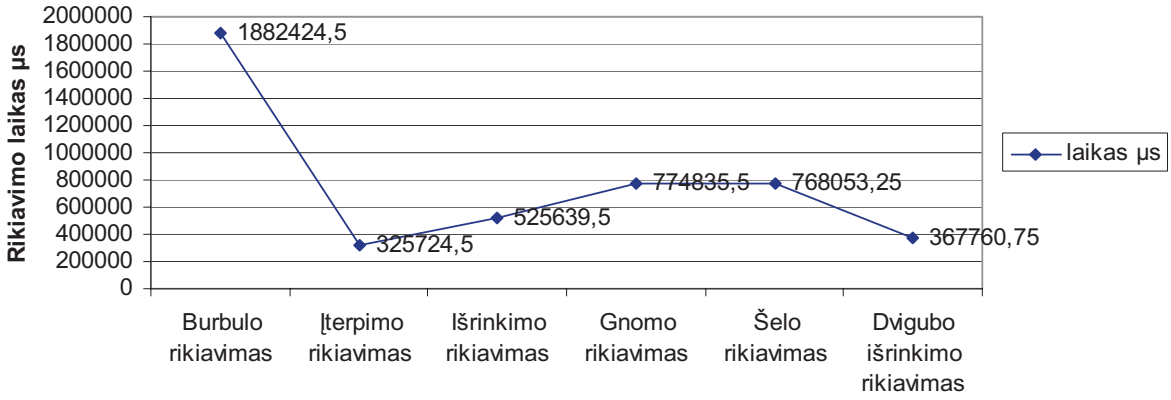
Daugiausia sukeitimų atliko *Burbulo*, *Įterpimo* ir *Gnomo* rikiavimai – visi trys po 6315980,25 sukeitimo. Mažiausia sukeitimų atliko *Išrinkimo* rikiavimas – tik 4988,75 sukeitimo, ir *Dvigubo išrinkimo* rikiavimas – tik 5000 sukeitimų.

Rikiavimo laikas pasiskirstęs panašiai kaip ir ankstesniuose to paties tipo grafikuose, tik *Šelo* rikiavimo laiko kreivė šiek tiek kilstelėjusi aukštyn, lyginant su mažesnių duomenų kiekių grafikais, kaip ir INT tipo to paties kiekio duomenų analizėje. Ilgiausiai 5000 reikšmių FLOAT tipo duomenis rikiavo *Burbulo* algoritmas, net 480257 mikrosekundes. Trumpiausiai – *Įterpimo* rikiavimas, surikiavęs duomenis per 84664,5 mikrosekundės, antroje vietoje – *Dvigubo išrinkimo* algoritmas, surikiavęs duomenis per 103153,75 mikrosekundės. Galima pastebėti, kad *Šelo* algoritmo rikiavimo laikas dar labiau atsilieka nuo *Išrinkimo* algoritmo rikiavimo laiko. *Šelo* rikiavimo laikas yra net 204243,25 mikrosekundės, o *Išrinkimo* rikiavimo laikas yra 132431,25 mikrosekundės. Skirtumas net 71812 mikrosekundžių.

Analizuojant rikiavimo algoritmus su FLOAT tipo duomenimis (10000 reikšmių), išaiškinta, kad *Šelo* rikiavimo palyginimų kreivė dar labiau kilstelėjusi aukštyn, lyginant su praeitu palyginimų bei sukeitimų kiekių pasiskirstymo grafiku, to paties tipo duomenų 5000 reikšmių, lygiai taip kaip ir INT tipo duomenų analizėje. Daugiausia palyginimų – net 99050094, atliko *Burbulo* rikiavimas, po jo – *Šelo* rikiavimas (nebe *Gnomo*), atlikęs 54352072,25 palyginimo ir *Išrinkimo* rikiavimas, atlikęs 49995000 palyginimų. *Gnomo* rikiavimas atliko 49783716,75 palyginimo. Mažiausia palyginimų atliko *Įterpimo* rikiavimas, tik 24896705,75, ir *Dvigubo išrinkimo* rikiavimas –25000000 palyginimų. INT tipo duomenų to paties duomenų kiekio analizėje mažiausia palyginimų atliko *Dvigubo išrinkimo* rikiavimas, o antroje vietoje liko *Įterpimo* rikiavimas. Bet reikia pastebėti, kad tiek INT tipo duomenų tiek FLOAT tipo duomenų analizėje *Dvigubo išrinkimo* ir *Įterpimo* rikiavimų analizėse palyginimų kiekiai skiriasi nežymiai.

Daugiausia sukeitimų atliko *Burbulo*, *Įterpimo* ir *Gnomo* rikiavimai – visi trys po 24886863,25 sukeitimo. Mažiausia sukeitimų atliko *Išrinkimo* rikiavimas – tik 9990,75 sukeitimų, ir *Dvigubo išrinkimo* rikiavimas – tik 10000 sukeitimų.

Rikiavimo laikas pasiskirstęs panašiai kaip ir ankstesniuose to paties tipo grafikuose, tik *Šelo* rikiavimo laiko kreivė dar labiau kilstelėjusi aukštyn lyginant su praeitu rikiavimo laiko pasiskirstymo grafiku (to paties tipo duomenų 5000 reikšmių), lygiai taip kaip ir INT tipo duomenų analizėje. (5 pav.).



5 pav. Skirtingų algoritmų rikiavimo laiko sklaida. FLOAT tipo duomenys, 10000 reikšmių

Išanalizavus CHAR tipo duomenų apdorojimo charakteristikas, gauti panašūs rezultatai – greičiausiai rikiavimus atliko *Įterpimo* rikiavimas, ilgiausiai – *Burbulo*.

1 lentelė. ***Agoritmų kiekybinės charakteristikos. CHAR tipo duomenys, 50 reikšmių***

CHAR tipo duomenys, 50 reikšmių			
Vidutiniai duomenys	**palyginimai**	**sukeitimai**	**laikas µs**
Burbulo rikiavimas	1898,75	571,50	34,75
Įterpimo rikiavimas	620,50	571,50	12,00
Išrinkimo rikiavimas	1225,00	46,50	21,00
Gnomo rikiavimas	1188,25	571,50	24,25
Šelo rikiavimas	823,75	149,25	18,25
Dvigubo išrinkimo rikiavimas	625,00	50,00	17,25

2 lentelė. ***Agoritmų kiekybinės charakteristikos. CHAR tipo duomenys, 500 reikšmių***

CHAR tipo duomenys, 500 reikšmių			
Vidutiniai duomenys	**palyginimai**	**sukeitimai**	**laikas µs**
Burbulo rikiavimas	238272,50	61245,25	4662,5
Įterpimo rikiavimas	61744,25	61245,25	1218,75
Išrinkimo rikiavimas	124750,00	486,00	1397,00
Gnomo rikiavimas	122984,50	61245,25	2611,00
Šelo rikiavimas	104797,00	8950,75	1612,50
Dvigubo išrinkimo rikiavimas	62500,00	500,00	1291,00

3 lentelė. ***Agoritmų kiekybinės charakteristikos. CHAR tipo duomenys, 5000 reikšmių***

CHAR tipo duomenys, 5000 reikšmių			
Vidutiniai duomenys	**palyginimai**	**sukeitimai**	**laikas µs**
Burbulo rikiavimas	24177663,50	6076581,00	431134,00
Įterpimo rikiavimas	6081580,00	6076581,00	81855,25

3 lentelės tęsinys

Išrinkimo rikiavimas	12497500,00	4902,00	127710,00
Gnomo rikiavimas	12158157,50	6076581,00	205968,50
Šelo rikiavimas	12658724,75	728210,50	177234,25
Dvigubo išrinkimo rikiavimas	6250000,00	5000,00	103787,25

4 lentelė. ***Agoritmų kiekybinės charakteristikos. CHAR tipo duomenys,10000 reikšmių***

CHAR tipo duomenys, 10000 reikšmių			
Vidutiniai duomenys	**palyginimai**	**sukeitimai**	**laikas µs**
Burbulo rikiavimas	97577741,25	24470499,5	1733829,75
Įterpimo rikiavimas	24480498,50	24470499,5	338982,75
Išrinkimo rikiavimas	49995000,00	9811,5	533421,50
Gnomo rikiavimas	48950995,50	24470499,5	827251,25
Šelo rikiavimas	58189188,50	3300541,0	810415,25
Dvigubo išrinkimo rikiavimas	25000000,00	10000,0	414149,50

Išvados

1. Pagrindinėmis rikiavimo algoritmų charakteristikomis reikia laikyti palyginimų kiekį, sukeitimų kiekį bei rikiavimo laiką. Kadangi analizė atliekama ne tik su dideliais duomenų kiekiais (pvz., 10000 reikšmių), bet ir su mažais (100 reikšmių), tai rikiavimo laikas užtrunka sekundės dalimis. Dėl šios priežasties dažniausiai rikiavimo laikas skaičiuojamas mikrosekundėmis.
2. Rikiavimo laikas priklauso nuo atliktų palyginimo operacijų kiekio, nuo sukeitimų operacijų

kiekio ir nuo programos kodo sudėtingumo. Duomenų tipai – INT, FLOAT ir CHAR – didelės įtakos rikiavimų charakteristikoms nedaro. Tačiau svarbu, ar tarp rikiuojamų duomenų visi duomenys skirtingi, ar yra vienodų duomenų. Dėl šios priežasties gali skirtis stabilių ir nestabilių algoritmų efektyvumas.

3. Rikiuojant atsitiktine tvarka pasiskirsčiusius duomenis, kurių kiekis yra nuo 50 iki 10000 reikšmių, visi algoritmai vienas kito atžvilgiu išlaiko panašias kitimo tendencijas. Pagal rikiavimo metodą lėčiausi yra sukeitimo grupės algoritmai.
4. Kadangi, rikiuojant iki 50 reikšmių, visi analizuojami algoritmai užima iki 40 μs rikiavimo laiko, tai mažiems duomenų kiekiams rikiuoti geriau tinka paprasto realizavimo algoritmai, pvz., *Įterpimo, Burbulo* arba *Gnomo*.
5. Didesniems duomenų kiekiams geriau naudoti *Įterpimo* rikiavimą, *Dvigubo išrinkimo* rikiavimą arba *Išrinkimo* rikiavimą, kadangi šie trys algoritmai, rikiuojant bet kurios apimties duomenis, buvo greičiausi.
6. Pats lėčiausias algoritmas atsitiktiniams duomenims visais atvejais buvo *Burbulo* algoritmas, pats greičiausias – *Įterpimo* algoritmas.
7. *Šelo* algoritmas nepatartinas naudoti dideliems atsitiktinių duomenų kiekiams rikiuoti, kadangi, didėjant rikiuojamų reikšmių kiekiui, didėja ir *Šelo* algoritmo rikiavimo laikas kitų rikiavimo algoritmų darbo atžvilgiu.

Literatūra

1. Juozapavičius A., 1997, *Duomenų struktūros ir algoritmai*. Mokomoji priemonė. Vilniaus universiteto leidykla. P. 19–42.
2. Juozapavičiaus A. Paskaitų konspektas, Algoritmų sudėtingumo analizė. < http://www.mif.vu.lt/~algis/dsax/dsanalize.pdf>. [2011-05-10].
3. Juozapavičiaus A. Paskaitų konspektas, Vidinio ir išorinio rūšiavimo algoritmai. <http://ututi.com/subject/VU/MIF/duomenu-strukturos-ir-algoritmai>. Šio dokumento tiesioginė prieiga: <http://mif.vu.lt/cs2/lt/kursai/algoritm/files/DSA-rusiavimas.pdf.> [2011-05-10]
4. Kompiuterija, 2000. *Mokymosi knyga studentams, moksleiviams, entuziastams*. Naujasis lankas. Kaunas. P. 445–454.
5. Šiaulių universiteto Informatikos katedros lekt. dr. G. Felinsko paskaitų moduliui „Euristinių algoritmų tyrimas ir taikymai“ konspektas, Algoritmų sudėtingumas <ik.su.lt/~grazvis/ea/LT%20-%20algoritmu%20sudetingumas.doc> [2011-05-07].
6. VGTU konspektai <www.techmat.vgtu.lt> serveris, <http://www.techmat.vgtu.lt/konspektai/Algoritmai/Paskaita61.pdf> [2011-05-11].
7. VKA mokomoji medžiaga. <vka.puslapiai.lt/Cpp/c++6.doc> p. 52–54. [2011-05-09].

ANALYSIS OF SORTING ALGORITHMS

Ingrida Orvidaitė, Marijus Bernotas

Summary

Six most used sorting algorithms (two from each of the three main groups: *selection*, *insertion*, and *exchange*) were analysed. Specifically, in the selection group – *Selection Sort* and *Double Selection Sort*, in the insertion group – *Insertion Sort* and *Shell Sort*, in the exchange group – *Bubble Sort* and *Gnome Sort* were analysed. These algorithms were compared by the amount of compares and swaps as well as sorting time. By using C++ Builder three programs necessary for analysis were created: one for data generation, another for examination of separate algorithms, and the main program that returns sorting parameters.

The data are generated in Int type (Integer number), Float type (Floating-point number) and Char type (alphabet). The data are generated by the number of values, for the generation of numbers we can appoint the range and so we can make files with different levels of data such as an almost sorted row or, on the contrary, a very scattered one. The program for examination of separate algorithms sorts data and after every swap writes a string of values in the text box, so we can follow the work of separate algorithms and we can see how different they are. And the main program sorts values by data type and returns only amount of compares, swaps and sorting time. Tables with these results were used to build diagrams and charts for comparisons of algorithms.

Keywords: algorithms, sorting.

RIKIAVIMO ALGORITMŲ TYRIMAS

Ingrida Orvidaitė, Marijus Bernotas

Santrauka

Greitesniam darbui su dideliu kiekiu duomenų, būtina juos surikiuoti. Esant dideliems duomenų kiekiams, ypatingą reikšmę įgauna rikiavimo algoritmo sudėtingumas – atlikimo greičio priklausomybė nuo duomenų kiekio.

Nagrinėjami 6 rikiavimo algoritmai, po du iš trijų pagrindinių grupių: iš išrinkimo grupės – išrinkimo rikiavimas (*selection sort*) ir dvigubo išrinkimo rikiavimas (*double selection sort*), iš įterpimo grupės – įterpimo rikiavimas (*insertion sort*) ir šelo rikiavimas (*shell sort*), iš sukeitimo grupės – burbulo rikiavimas (*bubble sort*) ir gnomo rikiavimas (*gnome sort*). Algoritmai gretinami pagal sukeitimų ir palyginimų kiekius, rikiavimo laiką. Analizė atliekama su skirtingo kiekio duomenimis.

Naudojant *C++ Builder* programą buvo sukurtos trys būtinos analizei atlikti programos duomenų generavimui, atskirų algoritmų analizės demonstravimui ir duomenų analizės charakteristikų tyrimui.

Prasminiai žodžiai: algoritmai, rikiavimas.

Įteikta 2011-11-15