

**VILNIAUS KOLEGIJA**

ELEKTRONIKOS IR INFOMATIKOS FAKULTETAS

**Programinės įrangos katedra**

Programu sistemos (vals. kodas 6531BX028)

Gabrielė Tamaševičiūtė

PI23S **grupė**

RIKIAVIMO ALGORITMŲ LYGINAMOJI ANALIZĖ

Kursinis darbas

Darbo vadovas (-ė) Dr. Romanas Tumasonis

Pateikimo data: 2024-03-15

Vilnius

2024

**Ivadas**

Duomenų rikiavimas yra esminis programavimo mokslų elementas, kuris leidžia manipuliuoti didelius duomenų kiekius pagal įvairius parametrus, pavyzdžiui, reikšmių dydį ar abėcėlinę tvarką. Duomenų rikiavimas programavime yra būtinas daugybėje kasdienių užduočių. Duomenų mokslinikai ir analitikai dažniausiai rikiuoja duomenenų rinkinius prieš tolesnę analizę, prekių sąrašai internetinese parduotuvėse rikiuojamos pagal kainą, populiarumą, vartotojo vertinimus arba daugelį kitų kriterijų, socialinių tinklų platformose įrašai, komentarai, pranešimai – ir tie yra rikiuojami pagal tam tikrus kriterijus. Galime suvokti, kad rikiavimo algoritmų tyrimas ir optimizacija yra svarbus uždavinys, nes net ir nedidelis efektyvumo pagerėjimas gali turėti reikšmingą poveikį sistemų veikimo greičiui ir resursų naudojimui. Šio kursinio darbo tema yra "Rikiavimo algoritmų lyginamoji analizė", kurioje dėmesys skiriamas dviem skirtingų tipų algoritmams: paprastajam Burbulo rikiavimo algoritmui ir sparčiajam Suliejimo rikiavimo algoritmui. Ši tema yra itin aktuali, nes rikiavimo algoritmų pasirinkimas ir jų veikimo efektyvumas gali turėti esminę įtaką programų, kurios juos naudoja, veikimo spartai ir efektyvumui. Kursinio darbo tikslas – išanalizuoti ir palyginti Burbulo ir Suliejimo rikiavimo algoritmų efektyvumą, taikant juos skirtingų dydžių ir išdėstymo duomenų rinkiniams. Šis tikslas yra pasirinktas siekiant suprasti, kaip šie du algoritmai veikia įvairiose situacijose ir kuris iš jų yra efektyvesnis konkrečiomis sąlygomis. Kursinio darbo uždaviniai:

1. Apibrėžti ir aprašyti Burbulo ir Suliejimo rikiavimo algoritmų teorinius principus ir jų veikimo principus, realizuoti algoritmus C++ programavimo kalba.
2. Burbulo ir Suliejimo rikiavimo algoritmų veikimo rezultatų analizė ir palyginimas.
3. Pateikti išsamias išvadas apie kiekvieno algoritmo privalumus ir trūkumus, remiantis atlikta analize.

Šio darbo tyrimo metodai apima teorinę mokslinės literatūros analizę, duomenų rikiavimo algoritmų realizacija C++ programavimo kalba ir eksperimentinį tyrimą, naudojant sukurtas programas.

**Burbulo rikiavimo algoritmo apibrėžimas ir aprašymas**

Burbulo rikiavimo algoritmas yra vienas iš paprasčiausių rikiavimo algoritmų, kurį naudojame, kai rūšiuojame mažą kiekį elementų. Šis algoritmas veikia iteruodamas per masyvą ir keisdamas vietomis gretimus elementus, jei pirmasis elementas yra didesnis už antrąjį, jie keičiami vietomis.Šis procesas kartojamas kiekvienai porai masyvo elementų, pradedant nuo pirmojo iki paskutiniojo ir tesiasi kol nebus išrikiuotas visas masyvas. Pagal Raimondą Čiegį (2007) “kadangi didžiausio elemento judėjimas panašus į vandens burbulo kilimą į paviršių, tai toks rūšiavimo algoritmas vadinamas burbulo metodu” (p.133). Burbulo rūšiavimo didėjančia tvarka algoritmo pavyzdinė programa:

#include <iostream>

using namespace std;

void bubbleSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main() {

int arr[] = { 64, 34, 25, 12 };

int n = sizeof(arr) / sizeof(arr[0]);

cout << "Nesurusiuotas masyvas: \n";

for (int i = 0; i < n; i++)

cout << arr[i] << " ";

cout << endl;

bubbleSort(arr, n);

cout << "Surusiuotas masyvas: \n";

for (int i = 0; i < n; i++)

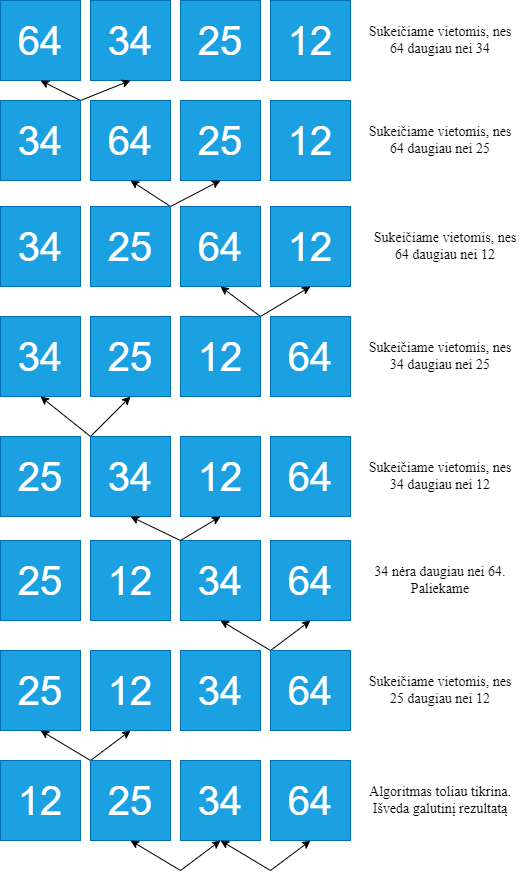
cout << arr[i] << " ";

cout << endl;

return 0;

}

Algoritmo veikimo principas:

1. Deklaruota „bubbleSort“ funkcija priima du argumentus: sveikųjų skaičių masyvą „int arr[]“ (žr 1 Pav.) ir masyvo ilgį „int n“. Funkcijos tikslas - surūšiuoti masyvą didėjimo tvarka naudojant burbulo rūšiavimo metodą.
2. Vykdome dvigubą ciklą:
   1. Išorinis ciklas („for (int i = 0; i < n - 1; i++)“) nustato, kiek kartų reikia peržiūrėti visą masyvą. Kadangi didžiausias elementas kiekvieno peržiūros metu "nuskęsta" į masyvo galą, kiekvieną kartą užtenka peržiūrėti vienu elementu mažesnę masyvo dalį, todėl ir yra „i < n - 1“.
   2. Vidinis ciklas („for (int j = 0; j < n - i - 1; j++)“) eina per masyvo elementus, kuriuos reikia palyginti ir, jei reikia, apkeisti vietomis (žr. 2 Pav.).
3. Vidiniame cikle kiekvienas elementas „arr[j]“ yra palyginamas su kitu šalimai esančiu elementu „arr[j + 1]“. Jei elementas „arr[j]“ yra didesnis už elementą „arr[j + 1]“, elementai apkeičiami vietomis naudojant kintamąjį „temp“. Tai užtirkina, kad po kiekvieno ciklo didžiausias elementas iš nesurūšiuotu nukeliautų į masyvo galą.
4. Prieš iškviečiant „bubbleSort“ funkciją atspausdiname pradinį masyvą „arr[]“, kad lengviau galėtume palyginti surušiuotą ir nesurušiuotą masyvą.
5. Iškviečiame „bubbleSort“ funkciją perduodant „arr[]“ kaip parametrą, kad funkcija surušiuotų masyvą.
6. Po rūšiavimo funkcijos iškvietimo, masyvas „arr[]“ yra vėl atspausdinamas, bet šį kartą kaip „Surūšiuotas masyvas“, demonstruodamas, kaip elementai buvo išrikiuoti didėjimo tvarka.
7. Rezultatas, toks, kad konsolėje gauname išvestus du masyvus, pirminį bei išrikiuotą (žr. 3).

**Suliejimo Rikiavimo Algoritmo Apibrėžimas ir Aprašymas**

Suliejimo rikiavimo algoritmas yra efektyvus algoritmas, naudingas didelių duomenų rinkinių rikiavimui, kuris naudoja „sakldyk ir valdyk“ principą padalindamas problemą į mažesnes, lengviau sprendžiamas dalis. Pagal Raimondą Čiegį (2007) „nesunku sujungti su surūšiuotus poaibius į vieną surūšiuotą aibę. Todėl metodas ir vadinamas *suliejimo* algoritmu“. Suliejimo rūšiavimo didėjančia tvarka algoritmo pavyzdinė programa:

#include <iostream>

#include <vector>

using namespace std;

void merge(vector<int>& arr, int l, int m, int r) {

int n1 = m - l + 1;

int n2 = r - m;

vector<int> L(n1), R(n2);

for (int i = 0; i < n1; i++)

L[i] = arr[l + i];

for (int j = 0; j < n2; j++)

R[j] = arr[m + 1 + j];

int i = 0;

int j = 0;

int k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(vector<int>& arr, int l, int r) {

if (l >= r) {

return;

}

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

int main() {

vector<int> arr = { 12, 11, 13, 5 };

int arr\_size = arr.size();

cout << "Nesurusiuotas masyvas: \n";

for (int i = 0; i < arr\_size; i++)

cout << arr[i] << " ";

cout << endl;

mergeSort(arr, 0, arr\_size - 1);

cout << "Surusiuotas masyvas: \n";

for (int i = 0; i < arr\_size; i++)

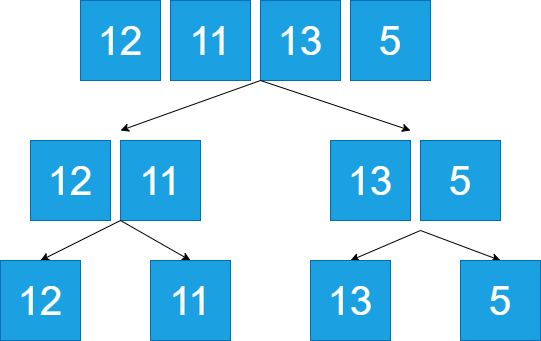
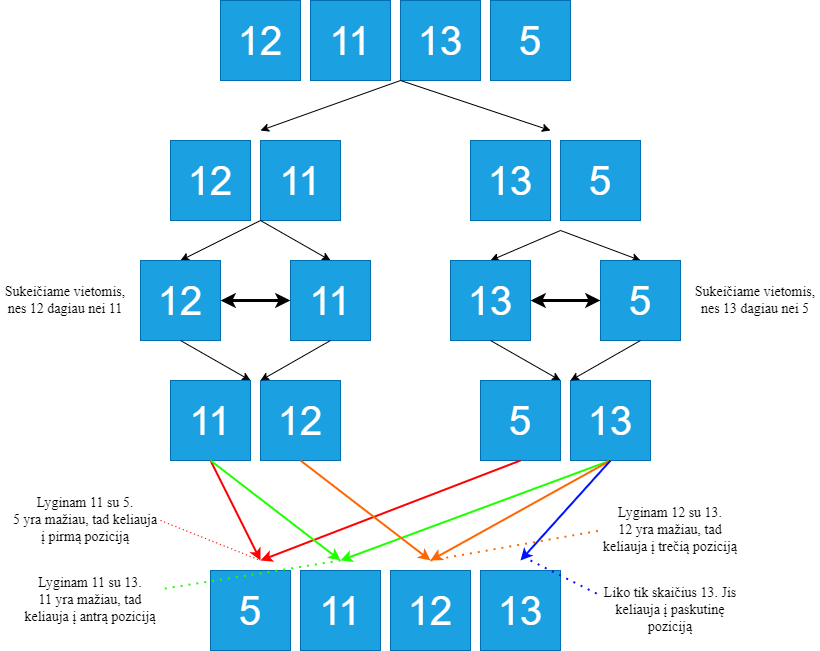
cout << arr[i] << " ";

cout << endl;

return 0;

}

Algoritmo veikimo principas:

1. Programa apibrėžia dvi funkcijas: „merge“, kuri atsakinga už sub-masyvų sujungimą ir rūšiavimą, ir „mergeSort“, kuri atsakinga už masyvo skaidymą ir rekursyvų algoritmo kvietimą.
2. Funkcija „mergeSort“ rekursyviai skaido masyvą į mažesnes dalis, naudodama vidurio tašką „m“. Skaidymas tęsiamas, kol sub-masyvuose lieka vienas elementas (žr. 3 Pav.) 
3. Funkcija „merge“ sujungia du surūšiuotus sub-masyvus į vieną surūšiuotą masyvą. Ji naudoja laikinus masyvus „L“ ir „R“, į kuriuos kopijuoja sub-masyvų duomenis. Tada, palyginus elementus, ji juos įterpia atgal į pagrindinį masyvą „arr“ jau išrikiuotai.
4. Funkcijoje „merge“, kai du sub-masyvai sujungiami, kiekvieno sub-masyvo elementai palyginami ir mažesnis iš jų įterpiamas į pagrindinį masyvą (žr. 4 Pav.).
5. Prieš kviečiant „mergeSort“ funkciją, pradinis masyvas „arr“ atspausdinamas konsolėje, kad būtų galima pamatyti pradinę masyvo būklę.
6. Funkcija „mergeSort“ iškviečiama pradiniam masyvui „arr“, pradedant nuo nulinės pozicijos 0 ir baigiant masyvo ilgiu minus vienas.
7. Baigus rūšiavimą, masyvas „arr“ vėl atspausdinamas, bet šį kartą kaip "Surūšiuotas masyvas". Tai parodo, kaip algoritmas išrikiavo elementus didėjimo tvarka.
8. Galiausiai, konsolėje rodomi du masyvai: pradinis nesurūšiuotas ir galutinis surūšiuotas, leidžiantis vizualiai įvertinti algoritmo atliktą rūšiavimą.