

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKLSŲ FAKULTETAS

INFORMACIJNIŲ SISTEMŲ KATEDRA

**ALGORITMAI IR DUOMENŲ STRUKTŪROS**

Savarankiškas darbas Nr.3

Darbą atliko: PRIf-17/2

Andrius Radzivonas

Gabrielius Šamšonas

Julius Lungys

Priėmė: Lekt. Nijolė Čeikienė

Vilnius, 2018

**Turinys**

[Uždavinys 3](#_Toc514276224)

[Apibendrintas algoritmas 3](#_Toc514276225)

[Testavimas ir jo rezultatai 3](#_Toc514276226)

[Programos kodas 4](#_Toc514276227)

Lentelių sąrašas

1 lentelė Efektyvumas skaičių aibėse nuo 10 iki 500 4

2 lentelė Efektyvumas skaičių aibėse nuo 1000 iki 50000 5

3 lentelė Efektyvumas skaičių aibėse nuo 100000 iki 2000000 5

Uždavinys

Realizuoti duomenų rūšiavimą greito rūšiavimo (Quicksort) metodu, suteikiant galimybę naudotojui dialogo metu pasirinkti rūšiavimą reikšmių didėjimo arba mažėjimo tvarka. Pradinius duomenis nuskaityti iš failų. Rūšiavimų rezultatus išvesti į failus. Realizuota algoritmą ištestuoti su įvairiais duomenų kiekiais. Išanalizuoti testavimo rezultatus įvertinant duomenų kiekio įtaka metodo vykdymo greičiui. Pateikti išvadas.

Apibendrintas algoritmas

Algoritmas iš „input.txt“ failo nuskaito skaičių aibę ir įdeda ją į vektorių. Sukuria dialogo langą kuriame naudotojas gali pasirinkti rikiuoti skaičius mažėjimo ar didėjimo tvarka. Surikiuoja aibę pagal pasirinkimą. Ekrane parašo kiek laiko užtruko rikiavimo funkcija. Surikiuotą aibę surašo į failą „output.txt“.

Testavimas ir jo rezultatai

Algoritmas buvo ištestuotas su skirtingo ilgumo skaičių aibėmis. Jos buvo sudarytos iš atsitiktinių skaičių nuo **-X** iki **X**. Testavimo rezultatai matomi lentelėse 1-3. Iš jų akivaizdu, kad kuo ilgesnė skaičių aibė tuo ilgesnis algoritmo veikimo greitis.

1 lentelė Efektyvumas skaičių aibėse nuo 10 iki 500

2 lentelė Efektyvumas skaičių aibėse nuo 1000 iki 50000

3 lentelė Efektyvumas skaičių aibėse nuo 100000 iki 2000000

Programos kodas

rez

Rezultatas

#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

#include <chrono>

using namespace std;

void Reverse(vector <int> &vect);

void Swap(vector <int> &vect, int i, int j);

void QuickSort(vector <int> &vect, int begin, int end);

bool isCorrect(vector <int> inputData, vector <int> outputData, int type);

int main() {

clock\_t time;

int number, type;

bool isRunning = true;

vector <int> inputData, outputData;

ifstream input;

input.open("input.txt");

while (input >> number) {

inputData.push\_back(number);

}

input.close();

while (isRunning) {

cout << "Ka norite atlitki?" << endl;

cout << "0 - Surusiuoti skaiciu seka nuo maziausio iki didziausio." << endl;

cout << "1 - Surusiuoti skaiciu seka nuo didziausio iki maziausio." << endl;

cin >> type;

system("cls");

switch (type) {

case 0:

isRunning = false;

break;

case 1:

isRunning = false;

break;

default:

cout << "Jusu pasirinkimas yra klaidingas." << endl;

}

}

outputData = inputData;

auto start = chrono::system\_clock::now(); //laiko pradzia

QuickSort(outputData, 0, outputData.size() - 1);

auto end = chrono::system\_clock::now(); //laiko pradzia

if (type == 1) {

Reverse(outputData);

}

ofstream output;

output.open("output.txt");

for (int a = 0; a < outputData.size(); a++) {

output << outputData[a] << " ";

}

output.close();

if (isCorrect(inputData, outputData, type)) { //Tiesiog checkinimas

cout << "Rusiavimas yra sekmingas." << endl;

}

else {

cout << "Ivyko klaida." << endl;

}

chrono::duration<double> elapsed\_seconds = end - start; //gl boyz

cout << "Operacijos atlikimo laikas: " << elapsed\_seconds.count();

system("pause");

return 0;

}

void Reverse(vector <int> &vect) {

reverse(vect.begin(), vect.end());

}

void Swap(vector <int> &vect, int i, int j) {

int temp = vect[j];

vect[j] = vect[i];

vect[i] = temp;

}

void QuickSort(vector <int> &vect, int begin, int end) {

int i = begin;

for (int a = begin; a < end; a++) {

if (vect[a] < vect[end]) {// jeigu yra mazesnis, tada swapinam

Swap(vect, a, i); // su keiciame vietomis ok

i++;

}

}

Swap(vect, i, end);

/\*for (int a = 0; a < vect.size(); a++) { //test

cout << vect[a] << " ";

}

cout << endl;\*/

if (i - 1 > begin) { //for cheking

QuickSort(vect, begin, i - 1);

}

if (i < end - 1) {

QuickSort(vect, i + 1, end);

}

}

bool isCorrect(vector <int> inputData, vector <int> outputData, int type) { //tiesiog checkinimas

sort(inputData.begin(), inputData.end());

if (type == 1) {

Reverse(inputData);

}

if (inputData == outputData) {

return true;

}

else return false;

}