Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Алгоритми та методи обчислення

Лабораторна робота №2

**«Обчислювальна складність алгоритмів сортування»**

Виконала:

студентка групи ІВ-71

Молчанова В.С.

Перевірив:

ст.вик. Порєв В. М.

Київ

2019 р.

**Мета:** Закріплення навичок практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритмів сортування.

**Завдання:** Використовуючи відповідний до варіанту алгоритм сортування

написати програму сортування масиву даних. Застосовуючи дану програму,

дослідити часову складність алгоритму сортування та порівняти її з

теоретичною алгоритмічною складністю.

**Алгоритм за варіантом:** прискорене бульбашкове сортування з початку до кінця.

**Роздруківка тексту програми:**

#include "appcore.h"

#include <iostream>

#include <QFile>

#define MAX\_N 20

#define DEFAULT\_ANSWER "надто великий для відображення"

AppCore::**AppCore**(QObject \*parent) : QObject(parent)

{

}

void AppCore::**receiveArray**(QString arrayString)

{

stringToArray(arrayString);

sort();

emit sendToQML(time, (n <= MAX\_N ? arrayString : DEFAULT\_ANSWER), arrayToString());

}

void AppCore::**receiveFromFile**(QString filePath){

QFile file(filePath);

QString arrayString = "";

if (file.*open*(QIODevice::ReadOnly|QIODevice::Text))

{

arrayString = QString::fromUtf8(file.readLine());

}

file.*close*();

stringToArray(arrayString);

sort();

emit sendToQML(time, (n <= MAX\_N ? arrayString : DEFAULT\_ANSWER) , arrayToString());

}

void AppCore::**receiveNumber**(int my\_n){

n = my\_n;

for (unsigned long long int i = 0; i < n;i++) {

array[i] = rand() - 10000;

}

QString input = arrayToString();

sort();

emit sendToQML(time, input, arrayToString());

}

void AppCore::**stringToArray**(QString arrayString){

QStringList list = arrayString.split(' ');

n = list.length();

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

array[i] = list[i].toInt();

}

}

QString AppCore::**arrayToString**(){

QString output = "";

if ( n <= 20){

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

output += QString::number(array[i])+" ";

}

}

else {

output = DEFAULT\_ANSWER;

}

return output;

}

void AppCore::**sort**(){

auto start = std::chrono::steady\_clock::now();

int k = 0;

bool flag = true;

while(flag){

flag = false;

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

if(array[i] > array[i+1]){

int temp = array[i];

array[i] = array[i+1];

array[i+1] =temp;

flag = true;

}

}

k++;

}

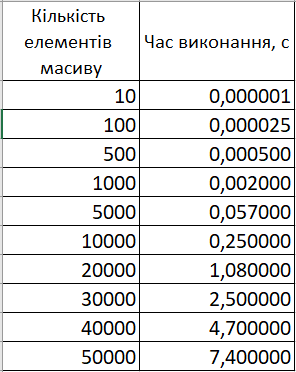
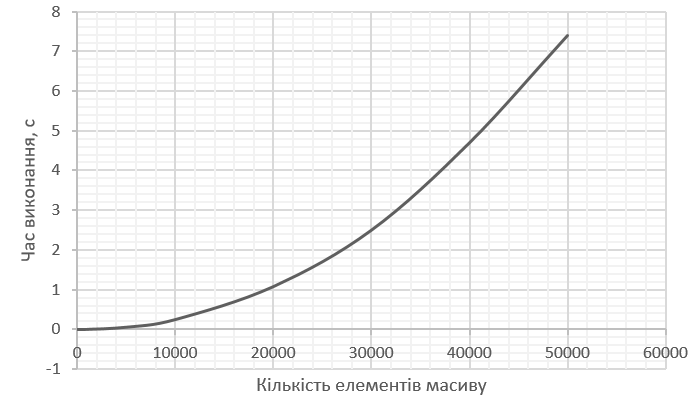
auto finish = std::chrono::steady\_clock::now();

time = std::chrono::duration\_cast<

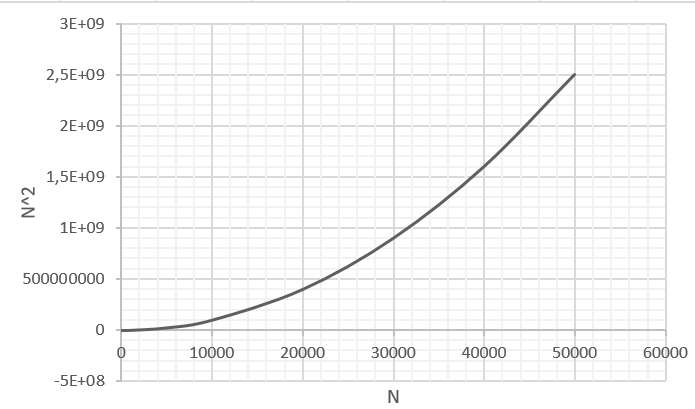
std::chrono::duration<double> >(finish - start).count();

}

**Аналіз результатів:**

1. Графік залежності часу виконання алгоритму від розміру вхідного масиву даних

2. Графік теоретично відомої обчислювальної складності (*O(N*2))



**Висновки:** У ході виконання лабораторної роботи я закріпила знання з базових понять алгоритмів, навички практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритмів сортування. Графік залежності часу виконання алгоритму збігається з графіком теоретичної обчислювальної складності, отже програма працює правильно.