Министерствонауки и высшего образования РФ

Севастопольский государственный университет

Кафедра информатики и управления в технических системах

ОТЧЕТ

по лабораторной работе №6

АНАЛИЗ АЛГОРИТМОВ СОРТИРОВКИ

по дисциплине «Основы теории алгоритмов»

Выполнил:

Студент группы ИВТ/б 22-о

Черняев Н.Г.

Проверил:

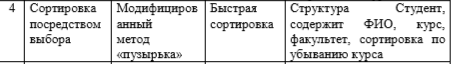
Абрамович А.Ю.

г. Севастополь 2019

**Цель работы**

Научиться оценивать сложность и количество операций для алгоритмов сортировки

**Вариант №4**



Создать структуру и реализуйте алгоритм сортировки согласно варианту задания и критерию сортировки. Реализоватб более эффективный алгоритм сортировки согласно варианту задания. Сравнить производительность различных алгоритмов.

**Текст программы**

#include "pch.h"

#include <iostream>

#include <fstream>

#include <algorithm>

#include <time.h>

#include <chrono>

#include <string>

using namespace std;

const int N = 1000;

struct Student {

string FIO;

int course;

string facult;

};

void SortSelect(Student \*a, int size) {

int i, j, k;

Student x;

for (i = 0; i < size; i++) {

k = i;

x = a[i];

for (j = i + 1; j < size; j++)

if (a[j].course < x.course) {

k = j;

x = a[j];

}

a[k] = a[i];

a[i] = x;

}

}

void SortBubbleModif(Student \*a, int size) {

int i = size;

int f;

Student t;

do {

f = 0;

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

if (a[k].course > a[k + 1].course)

{

t = a[k];

a[k] = a[k + 1];

a[k + 1] = t;

f = 1;

}

i--;

} while (f && i > 1);

}

void SortQuick(Student \*a, int size) {

long i = 0, j = size - 1;

Student temp, p;

p.course = a[size >> 1].course;

do {

while (a[i].course < p.course) i++;

while (a[j].course > p.course) j--;

if (i <= j) {

temp = a[i];

a[i] = a[j];

a[j] = temp;

i++; j--;

}

} while (i <= j);

if (j > 0) SortQuick(a, j);

if (size > i) SortQuick(a + 1, size - i);

}

int main()

{

int n;

Student mas[N];

cout << "Enter amount of elements in massive" << endl;

cin >> n;

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

{

cout << "Enter " << i + 1 << " elem" << endl;

cin >> mas[i].FIO;

cin >> mas[i].course;

cin >> mas[i].facult;

}

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

cout << "FIO: "

<< mas[i].FIO << " "

<< "Course: "

<< mas[i].course << " "

<< "Facult: "

<< mas[i].facult << endl;

}

cout << endl;

auto start1 = chrono::steady\_clock::now();

SortSelect(mas, n);

auto end1 = chrono::steady\_clock::now();

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

cout << "FIO: "

<< mas[i].FIO << " "

<< "Course: "

<< mas[i].course << " "

<< "Facult: "

<< mas[i].facult << endl;

}

cout << endl

<< "SORT SELECT Time taken in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end1 - start1).count()

<< endl;

auto start2 = chrono::steady\_clock::now();

SortBubbleModif(mas, n);

auto end2 = chrono::steady\_clock::now();

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

cout << "FIO: "

<< mas[i].FIO << " "

<< "Course: "

<< mas[i].course << " "

<< "Facult: "

<< mas[i].facult << endl;

}

cout << endl

<< "SORT BUBBLE MODIF Time taken in nanoseconds: "

<< chrono::duration\_cast<chrono::nanoseconds>(end2 - start2).count()

<< endl;

auto start3 = chrono::steady\_clock::now();

SortQuick(mas, n);

auto end3 = chrono::steady\_clock::now();

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

cout << "FIO: "

<< mas[i].FIO << " "

<< "Course: "

<< mas[i].course << " "

<< "Facult: "

<< mas[i].facult << endl;

}

cout << endl

<< "SORT QUICK Time taken in nanoseconds: "

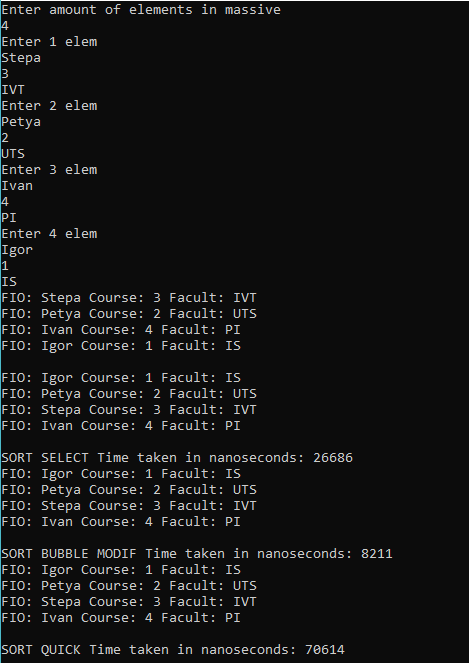
<< chrono::duration\_cast<chrono::nanoseconds>(end3 - start3).count()

<< endl;

}

**Результаты**

Пример работы программы:



Как видно из результатов работы программы сортировка методом модифицированного «пузырька» наиболее быстрая.

**Вывод**

В данной лабораторной работе были изучены алгоритмы поиска кратчайших путей на графах на примере метода динамического программирования.