Лабораторная работа №6

“Анализ алгоритмов сортировки”

1 ЦЕЛЬ РАБОТЫ

Научиться оценивать сложность и количество операций для алгоритмов сортировки.

2 ПОСТАНОВКА ЗАДАЧИ

1. Для каждого из приведённых алгоритмов найти оценку для количества шагов и количества требуемой памяти.

2. Создать структуру и реализовать алгоритм сортировки согласно варианту задания.

3. Реализовать более эффективные алгоритмы сортировки согласно варианту задания.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | Простейший алгоритм 1 | Простейший алгоритм 2 | Быстрый алгоритм | Структура и критерий сортировки |
| 4 | Сортировка посредством выбора | Модифицированный метод пузырька | Быстрая сортировка | Структура Студент, содержит ФИО, курс, факультет, сортировка по убыванию курса |

4. Сравнить производительность различных алгоритмов.

3 ТЕКСТ ПРОГРАММЫ

#include <conio.h>

#include <time.h>

#include <ctime>

#include <windows.h>

#include <iostream>

using namespace std;

const int N = 10000;

struct student {

char fio[10];

int faculty;

int kurs;

};

// сортировка модифицированным пузырьком

void modBubble(student \*a, int n)

{

int i=n,t;

int f;

do {

f=0;

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

if (a[j].kurs < a[j+1].kurs)

{

swap(a[j], a[j + 1]);

f=1;

}

i--;

} while (f && i>1);

}

// сортировка выбором

void selectSort(student \*a, int n)

{

int i, j, imax;

for(i = n; i > 1; i--)

{

imax = 0;

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

{

if(a[j].kurs < a[imax].kurs)

imax = j;

swap(a[i], a[imax]);

}

}

}

// быстрая сортировка

void quickSort(student \*a, int n)

{

int i = 0, j = n;

student p;

p.kurs = a[n / 2].kurs;

do

{

while (a[i].kurs > p.kurs) i++;

while (a[j].kurs < p.kurs) j--;

if (i <= j)

{

swap(a[i], a[j]);

i++;

j--;

}

} while (i <= j);

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

if (n > i) quickSort(a + i, n - i);

}

int main()

{

FILE \*stream;

stream = fopen("Result.txt", "w+");

student a[N];

int n = N;

int key;

for (int i = 0; i < n; i++) a[i].kurs = rand()%100;

cout << "Выбор сортировки: " << endl;

cout << "1 - Пузырьком" << endl;

cout << "2 - Выбором" << endl;

cout << "3 - Быстрая" << endl;

cout << "Ваш выбор: "; cin >> key;

double start\_time, end\_time;

switch (key)

{

case 1:

start\_time = clock();

modBubble(a, N);

end\_time = clock();

break;

case 2:

start\_time = clock();

selectSort(a, N);

end\_time = clock();

break;

case 3:

start\_time = clock();

quickSort(a, N);

end\_time = clock();

break;

}

double time = end\_time - start\_time;

cout << "Результаты сортировки занесены в файл Result.txt" << endl;

for (int i = 0; i < n; i++) fprintf(stream, "%i ", a[i].kurs);

cout << "Время работы программы (MS) = " << time/(double)CLOCKS\_PER\_SEC << endl;

fclose(stream);

return 0;

}

4 РЕЗУЛЬТАТЫ

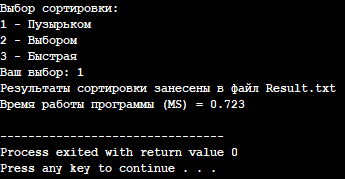


Рисунок 1 – Результат сортировки пузырьком

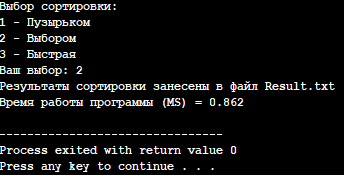


Рисунок 2 – Результат сортировки выбором

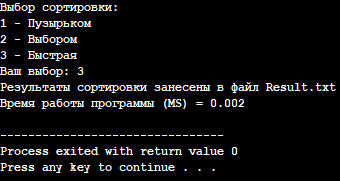


Рисунок 3 – Результат быстрой сортировки

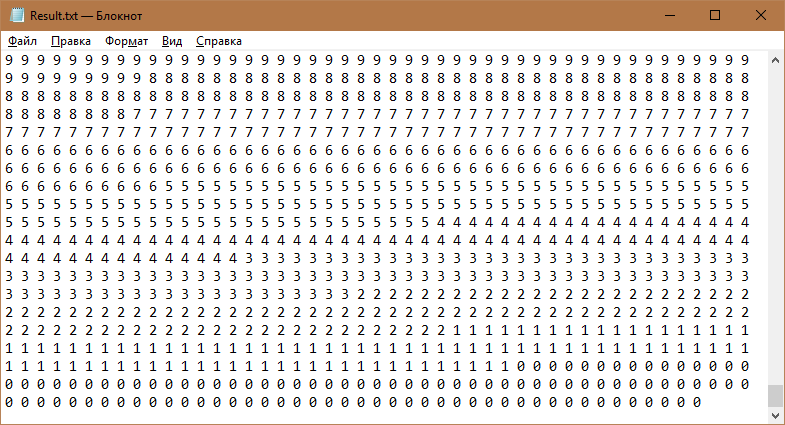


Рисунок 4 – Результат выполнения сортировок

ВЫВОД

В ходе выполнения лабораторной работы были приобретены навыки оценки сложности и эффективности различных алгоритмов сортировки. Согласно варианту, были составлены три алгоритма: сортировка пузырьком, сортировка выбором, “быстрая сортировка”. Результаты эффективности показали, что наиболее быстрым является алгоритм “быстрой” сортировки, а наиболее долгим – сортировка выбором.