Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №2

по курсу «Логика и основы алгоритмизации в инженерных задачах»

Выполнили:

студенты группы 21ВВ2

Сорокина Е.А.

Нефедова Е.Д.

Принял:

Юрова О.В.

Пенза 2022

**Задание 1:**

1. Вычислить порядок сложности программы (*О*-символику).

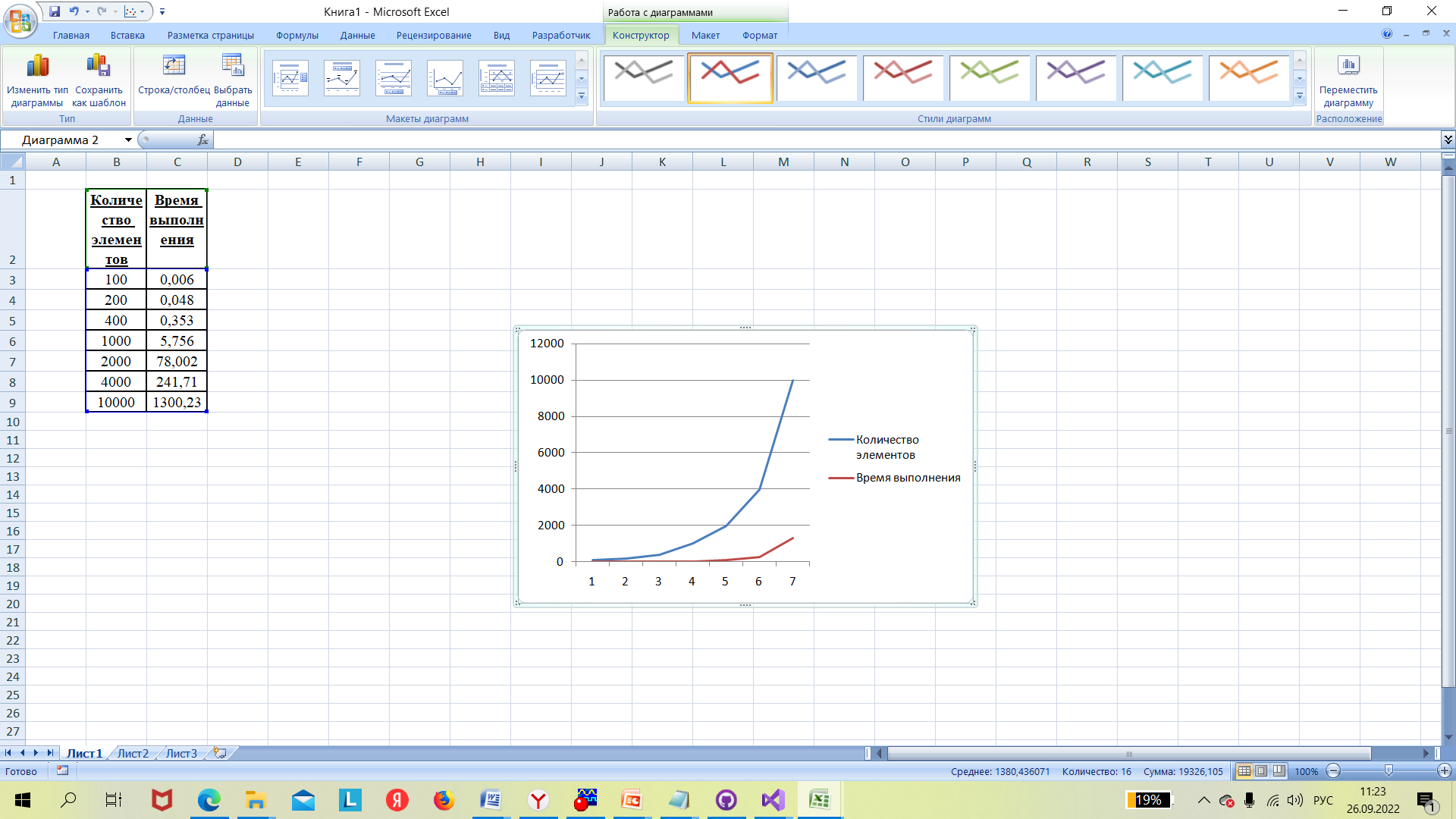
Порядок сложности программы:

O(n^3)

1. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.

|  |  |  |
| --- | --- | --- |
|  | **Количество элементов** | **Время выполнения** |
| 1 | 100 | 0,006000 |
| 2 | 200 | 0,048000 |
| 3 | 400 | 0,353000 |
| 4 | 1000 | 5,756000 |
| 5 | 2000 | 78,00200 |
| 6 | 4000 | 241,7100 |
| 7 | 10000 | 1300,2300 |

1. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.



**Задание 2**:

1. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.
2. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.
3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.
4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.
5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Количество элементов** | **1 массив** | | | **2 массив** | | | **3 массив** | | | **4 массив** | | |
|  | **shell** | **qs** | **qsort** | **shell** | **qs** | **qsort** | **shell** | **qs** | **qsort** | **shell** | **qs** | **qsort** |
| 100 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 |
| 200 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,001000 |
| 400 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 |
| 1000 | 0,000000 | 0,000000 | 0,001000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,001000 | 0,000000 |
| 2000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,001000 | 0,000000 | 0,000000 | 0,001000 | 0,000000 | 0,001000 | 0,001000 |
| 4000 | 0,001000 | 0,000000 | 0,001000 | 0,000000 | 0,000000 | 0,001000 | 0,001000 | 0,000000 | 0,001000 | 0,002000 | 0,001000 | 0,004000 |
| 10000 | 0,006000 | 0,001000 | 0,003000 | 0,002000 | 0,001000 | 0,005000 | 0,001000 | 0,001000 | 0,004000 | 0,001000 | 0,001000 | 0,007000 |

**Листинг**

**Задание 1**

#include <iostream>

#include <time.h>

void Program(int n)

{

double TIME1 = clock();

int\*\* A = (int\*\*)malloc(n \* sizeof(int\*));

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

A[i] = (int\*)malloc(n \* sizeof(int));

}

int\*\* B = (int\*\*)malloc(n \* sizeof(int\*));

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

B[i] = (int\*)malloc(n \* sizeof(int));

}

int\*\* C = (int\*\*)malloc(n \* sizeof(int\*));

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

C[i] = (int\*)malloc(n \* sizeof(int));

}

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

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

A[i][j] = 10 + rand() % 90;

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

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

B[i][j] = 10 + rand() % 90;

double time1 = clock();

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

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

C[i][j] = 0;

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

C[i][j] += A[i][k] \* B[k][j];

}

}

double time2 = clock();

double TIME2 = clock();

printf("Кол-во элементов: %d\n", n);

printf("Время умножения: %lf \n", (time2 - time1) / CLOCKS\_PER\_SEC);

printf("Время выполнение програмы: %lf \n\n\n", (TIME2 - TIME1) / CLOCKS\_PER\_SEC);

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

free(A[i]);

}

free(A);

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

free(B[i]);

}

free(B);

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

free(C[i]);

}

free(C);

}

int main(void)

{

setlocale(0, "");

srand(time(NULL));

Program(100);

Program(200);

Program(400);

Program(1000);

Program(2000);

Program(4000);

Program(10000);

system("pause");

return 0;

}

**Задание 2**

#include <iostream>

#include <cstdlib>

#include <iomanip>

#include <stdio.h>

FILE\* file;

void shell(int\* items, int count)

{

int i, j, gap, k;

int x, a[5];

a[0] = 9; a[1] = 5; a[2] = 3; a[3] = 2; a[4] = 1;

for (k = 0; k < 5; k++) {

gap = a[k];

for (i = gap; i < count; ++i) {

x = items[i];

for (j = i - gap; (x < items[j]) && (j >= 0); j = j - gap)

items[j + gap] = items[j];

items[j + gap] = x;

}

}

}

void qs(int\* items, long int left, long int right) //вызов функции: qs(items, 0, count-1);

{

long int i, j;

long int x, y;

i = left; j = right;

/\* выбор компаранда \*/

x = items[(left + right) / 2];

do {

while ((items[i] < x) && (i < right)) i++;

while ((x < items[j]) && (j > left)) j--;

if (i <= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) qs(items, left, j);

if (i < right) qs(items, i, right);

}

int compare(const void\* x1, const void\* x2)

{

return (\*(int\*)x1 - \*(int\*)x2);

}

void Program(int n) {

int\* arr11 = (int\*)malloc(n \* sizeof(int));

int\* arr12 = (int\*)malloc(n \* sizeof(int));

int\* arr13 = (int\*)malloc(n \* sizeof(int));

int\* arr21 = (int\*)malloc(n \* sizeof(int));

int\* arr22 = (int\*)malloc(n \* sizeof(int));

int\* arr23 = (int\*)malloc(n \* sizeof(int));

int\* arr31 = (int\*)malloc(n \* sizeof(int));

int\* arr32 = (int\*)malloc(n \* sizeof(int));

int\* arr33 = (int\*)malloc(n \* sizeof(int));

int\* arr41 = (int\*)malloc(n \* sizeof(int));

int\* arr42 = (int\*)malloc(n \* sizeof(int));

int\* arr43 = (int\*)malloc(n \* sizeof(int));

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

{

arr11[i] = 10 + rand() % 90;

arr12[i] = 10 + rand() % 90;

arr13[i] = 10 + rand() % 90;

}

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

{

arr21[i] = i + 1;

arr22[i] = i + 1;

arr23[i] = i + 1;

}

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

{

arr31[i] = n - i;

arr32[i] = n - i;

arr33[i] = n - i;

}

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

{

if (i < n / 2) {

arr41[i] = i + 1;

arr42[i] = i + 1;

arr43[i] = i + 1;

}

else {

arr41[i] = n - i;

arr42[i] = n - i;

arr43[i] = n - i;

}

}

fprintf(file, "Кол-во элементов: %d", n);

///////////////////////////////////

fprintf(file, "\n\nЗаполение массива 1 сопособом: рандом\n");

double time1 = clock();

shell(arr11, n);

double time2 = clock();

fprintf(file, "\nСортировка Shell: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

time1 = clock();

qs(arr12, 0, n - 1);

time2 = clock();

fprintf(file, "\nСортировка qs: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

time1 = clock();

qsort(arr13, n, sizeof(int), compare);

time2 = clock();

fprintf(file, "\nСортировка qsors: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

//////////////////////////////////////

fprintf(file, "\n\nЗаполение массива 2 сопособом: по возрастанию\n");

time1 = clock();

shell(arr21, n);

time2 = clock();

fprintf(file, "\nСортировка Shell: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

time1 = clock();

qs(arr22, 0, n - 1);

time2 = clock();

fprintf(file, "\nСортировка qs: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

time1 = clock();

qsort(arr23, n, sizeof(int), compare);

time2 = clock();

fprintf(file, "\nСортировка qsors: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

//////////////////////////////////////

fprintf(file, "\n\nЗаполение массива 3 сопособом: по убыванию\n");

time1 = clock();

shell(arr31, n);

time2 = clock();

fprintf(file, "\nСортировка Shell: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

time1 = clock();

qs(arr32, 0, n - 1);

time2 = clock();

fprintf(file, "\nСортировка qs: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

time1 = clock();

qsort(arr33, n, sizeof(int), compare);

time2 = clock();

fprintf(file, "\nСортировка qsors: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

//////////////////////////////////////

fprintf(file, "\n\nЗаполение массива 4 сопособом: возрастие/убывание\n");

time1 = clock();

shell(arr41, n);

time2 = clock();

fprintf(file, "\nСортировка Shell: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

time1 = clock();

qs(arr42, 0, n - 1);

time2 = clock();

fprintf(file, "\nСортировка qs: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

time1 = clock();

qsort(arr43, n, sizeof(int), compare);

time2 = clock();

fprintf(file, "\nСортировка qsors: %lf", (time2 - time1) / CLOCKS\_PER\_SEC);

//////////////////////////////////////

fprintf(file, "\n\n\n");

free(arr11); free(arr12); free(arr13);

free(arr21); free(arr22); free(arr23);

free(arr31); free(arr32); free(arr33);

free(arr41); free(arr42); free(arr43);

}

int main(void)

{

fopen\_s(&file, "txt.txt", "w");

setlocale(0, "");

Program(100);

Program(200);

Program(400);

Program(1000);

Program(2000);

Program(4000);

printf("Результат записан в файл\n");

fclose(file);

system("pause");

return 0;

}

**Вывод:** в ходе данной лабораторной работы мы приобрели и закрепили навыки оценивания времени выполнения алгоритмов. В результате заполнения таблицы, было выяснено, что самой быстрой является сортировка, реализованная нами (qs – быстрая сортировка), а самой медленной оказалась – qsort.