Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Оценка времени выполнения программы»

**Выполнил студент группы 19ВВ3:**

Филатова Д.С.

**Принял:**

Митрохин М.А.

Пенза 2020

**Цель работы:**

Оценка времени выполнения программы.

**Ход работы:**

**Задание 1.**

1. Вычислили порядок сложности программы.

O(n2) + O(n2) + O(n3) = O(n3)

1. Оценили 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 |
| 100 | 0.002 |
| 200 | 0.03 |
| 400 | 0.268 |
| 1000 | 5.684 |
| 2000 | 75.117 |

1. Построили график зависимости времени выполнения программы от размера матриц

и сравнили полученный результат с теоретической оценкой.

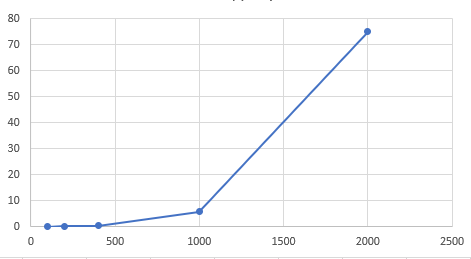


Рисунок 6. график зависимости

**Листинг (Задание 1):**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Header.h"

#include <stdio.h>

#include <stdlib.h>

#include<time.h>

#pragma comment(linker, "/STACK:100000000000000000")

int main(void)

{

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

int i = 0, j = 0, r;

int a[2000][2000], b[2000][2000], c[2000][2000], elem\_c;

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

while (i < 2000)

{

while (j < 2000)

{

a[i][j] = rand() % 100 + 1; // заполняем массив случайными числами

j++;

}

i++;

}

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

i = 0; j = 0;

while (i < 2000)

{

while (j < 2000)

{

b[i][j] = rand() % 100 + 1; // заполняем массив случайными числами

j++;

}

i++;

}

double start\_time = clock();

for (i = 0; i < 2000; i++)

{

for (j = 0; j < 2000; j++)

{

elem\_c = 0;

for (r = 0; r < 2000; r++)

{

elem\_c = elem\_c + a[i][r] \* b[r][j];

c[i][j] = elem\_c;

}

}

}

double end\_time = clock();

double search\_time = end\_time - start\_time;

printf("%lf", search\_time/CLK\_TCK);

\_getch();

}

**Задание 2.**

**Результат работы программы:**

NUMBER OF ELEMENTS = 10000

Time shell random = 0.012000

Time qsort random = 0.001000

Standart qsort random = 0.004000

Time shell vozrast = 0.000000

Time qsort vozrast = 0.002000

Standart qsort vozrast = 0.000000

Time shell pila = 0.011000

Time qsort pila = 0.006000

Standart qsort pila = 0.012000

Time shell ubiva = 0.023000

Time qsort ubiva = 0.002000

Standart qsort ubiva = 0.005000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NUMBER OF ELEMENTS = 20000

Time shell random = 0.049000

Time qsort random = 0.004000

Standart qsort random = 0.009000

Time shell vozrast = 0.001000

Time qsort vozrast = 0.004000

Standart qsort vozrast = -0.001000

Time shell pila = 0.074000

Time qsort pila = 0.031000

Standart qsort pila = 0.041000

Time shell ubiva = 0.121000

Time qsort ubiva = 0.004000

Standart qsort ubiva = 0.015000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NUMBER OF ELEMENTS = 50000

Time shell random = 0.377000

Time qsort random = 0.008000

Standart qsort random = 0.021000

Time shell vozrast = 0.001000

Time qsort vozrast = 0.007000

Standart qsort vozrast = 0.000000

Time shell pila = 0.325000

Time qsort pila = 0.115000

Standart qsort pila = 0.100000

Time shell ubiva = 0.603000

Time qsort ubiva = 0.007000

Standart qsort ubiva = 0.024000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### NUMBER OF ELEMENTS = 100000

### Time shell random = 1.351000

### Time qsort random = 0.017000

### Standart qsort random = 0.035000

### Time shell vozrast = 0.003000

### Time qsort vozrast = 0.015000

### Standart qsort vozrast = 0.000000

### Time shell pila = 1.602000

### Time qsort pila = 0.574000

### Standart qsort pila = 0.252000

### Time shell ubiva = 2.760000

### Time qsort ubiva = 0.018000

### Standart qsort ubiva = 0.052000

### Листинг:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <random>

#include <time.h>

#include <stdlib.h>

#include <stdio.h>

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, int left, int right) {

int i, j;

int x, y;

i = left; j = right;

x = items[(left + right) / 2 - 20];

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 cmpfunc(const int\* i, const int\* j) {

return \*i - \*j;

}

int main() {

FILE\* sort;

sort = fopen("C:\\sortt.txt", "a");

int n;

srand(time(NULL));

printf("Enter lenght:");

scanf("%d", &n);

fprintf(sort, "Number of elements = %d\n", n);

int\* arr\_random = new int[n];

int\* arr\_vozrast = new int[n];

int\* arr\_ubiva = new int[n];

int\* arr\_pila = new int[n];

int\* arr\_random\_3 = new int[n];

int\* arr\_random\_2 = new int[n];

int\* arr\_vozrast\_2 = new int[n];

int\* arr\_vozrast\_3 = new int[n];

int\* arr\_ubiva\_3 = new int[n];

int\* arr\_ubiva\_2 = new int[n];

int\* arr\_pila\_2 = new int[n];

int\* arr\_pila\_3 = new int[n];

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

arr\_random[i] = rand() % 100;

arr\_random\_2[i] = arr\_random[i];

arr\_random\_3[i] = arr\_random[i];

}

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

arr\_vozrast[i] = i;

arr\_vozrast\_2[i] = arr\_vozrast[i];

arr\_vozrast\_3[i] = arr\_vozrast[i];

}

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

arr\_ubiva[i] = n - i;

arr\_ubiva\_2[i] = arr\_ubiva[i];

arr\_ubiva\_3[i] = arr\_ubiva[i];

}

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

if (i <= n / 2) {

arr\_pila[i] = i;

arr\_pila\_2[i] = arr\_pila[i];

arr\_pila\_3[i] = arr\_pila[i];

}

if (i > n / 2) {

arr\_pila[i] = n - i;

arr\_pila\_2[i] = arr\_pila[i];

arr\_pila\_3[i] = arr\_pila[i];

}

}

double start = clock();

shell(arr\_random, n);

double end = clock();

fprintf(sort, "Time shell random = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

int left = 0;

double start\_new = clock();

qs(arr\_random\_2, left, n - 1);

double end\_new = clock();

fprintf(sort, "Time qsort random = %lf\n", (end\_new - start\_new) / CLOCKS\_PER\_SEC);

start = clock();

qsort(arr\_random\_3, n, sizeof(int), (int(\*)(const void\*, const void\*))cmpfunc);

end = clock();

fprintf(sort, "Standart qsort random = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

start = clock();

shell(arr\_vozrast, n);

end = clock();

fprintf(sort, "Time shell vozrast = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

start = clock();

qs(arr\_vozrast\_2, left, n - 1);

end = clock();

fprintf(sort, "Time qsort vozrast = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

start = clock();

qsort(arr\_vozrast\_3, n, sizeof(int), (int(\*)(const void\*, const void\*))cmpfunc);

fprintf(sort, "Standart qsort vozrast = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

end = clock();

start = clock();

shell(arr\_pila, n);

end = clock();

fprintf(sort, "Time shell pila = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

start = clock();

qs(arr\_pila\_2, left, n - 1);

end = clock();

fprintf(sort, "Time qsort pila = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

start = clock();

qsort(arr\_pila\_3, n, sizeof(int), (int(\*)(const void\*, const void\*))cmpfunc);

end = clock();

fprintf(sort, "Standart qsort pila = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

start = clock();

shell(arr\_ubiva, n);

end = clock();

fprintf(sort, "Time shell ubiva = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

start = clock();

qs(arr\_ubiva\_2, left, n - 1);

end = clock();

fprintf(sort, "Time qsort ubiva = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

start = clock();

qsort(arr\_ubiva\_3, n, sizeof(int), (int(\*)(const void\*, const

void\*))cmpfunc);

end = clock();

fprintf(sort, "Standart qsort ubiva = %lf\n", (end - start) / CLOCKS\_PER\_SEC);

return 0;

### }

### Выводы

Оценили время выполнения программы.