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 программ»

Выполнил:

Студент группы 23ВВВ2

Кузин М. Е.

Приняли:

Юрова О. В.

Митрохин М. А.

Пенза 2024

**Цель работы**

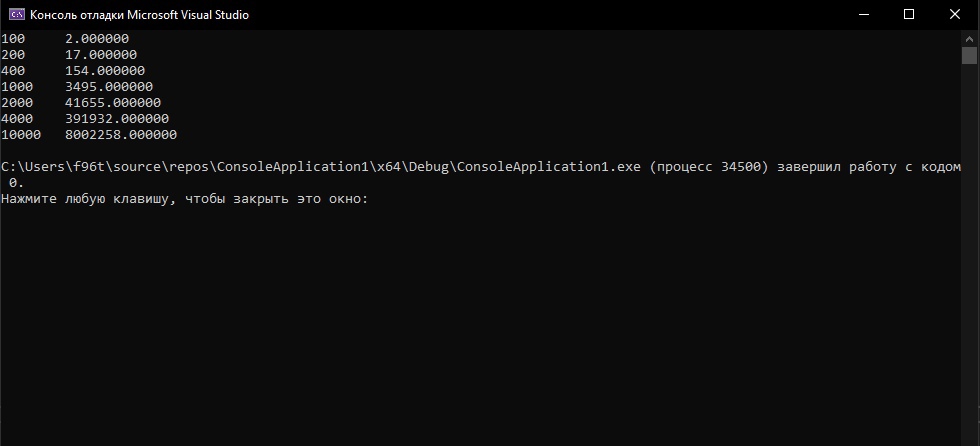
Изучить и измерить производительность алгоритмов, сравнить с предполагаемой сложностью программы.

**Ход работы**

Задание 1:

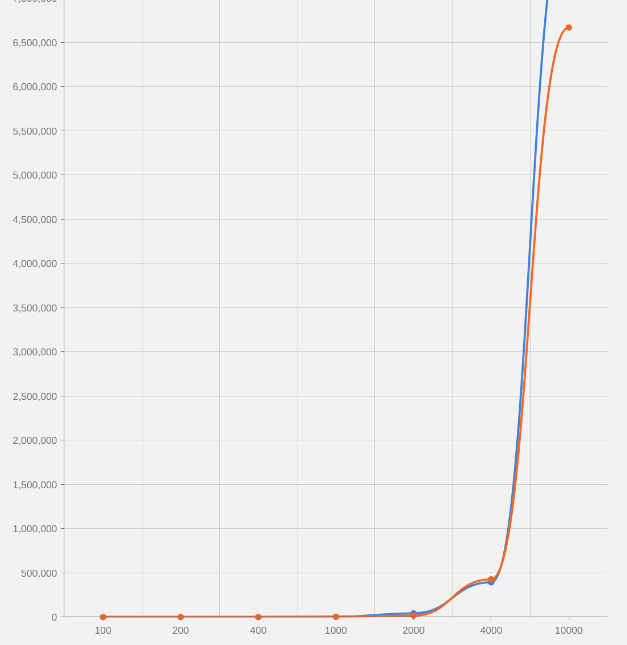
1. Сложность программы составляет O(n3).
2. Оценил время выполнения программы и кода.

Левый столбец – размер перемножаемых матриц, правый столбец – время в миллисекундах, нужное для их перемножения.



**Рисунок 1 – Результат работы программы**

1. Построил график зависимости времени выполнения программы от размера матриц (рисунок 2) и сравнил полученный результат с теоретической оценкой.



**Рисунок 2 – График зависимости времени выполнения программы от размера матриц**

Проанализировав данный рисунок, можно сказать, что график демонстрирует прямую зависимость: с увеличением размеров массивов увеличивается время выполнения программы. Фактический результат на больших объемах данных практически в 2 раза превосходит теоретическую оценку.Это может быть связано с производительностью компьютера или с оптимизацией компилятора VisualStudio 2019.

**Задание 2.**

Qsort сортировка на всех наборахчисел показала наибольший по времени результат от 330 мс до428мс, что показало низкую эффективность.

Представленный в условиях лабораторной работы алгоритм QS, показал лучшие результаты на всех видах массивов. Алгоритм сортировки Shell уступил по скорости сортировки всех видов массивов алгоритму QS, но показал хороший результат на последних трех массивах.

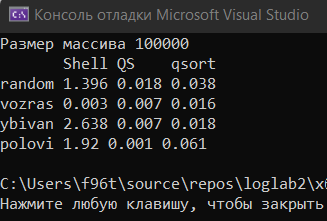
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Случайный | Возрастающий | Убывающий | Половинчатый |
| Shell | 1.396 | 0.003 | 2.638 | 1.92 |
| QS | 0.018 | 0.007 | 0.007 | 0.001 |
| QSort | 0.038 | 0.016 | 0.018 | 0.061 |

Программа создает измеряет время выполнения сортировки тремя различными способами на 4 видах массивов: случайный, возрастающий, убывающий и массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

Первое время – Алгоритм сортировки Шелла

Второе время – Алгоритм сортировки QS

Третье время – Алгоритм сортировки Qsort



**Рисунок 3 – результаты работы программы**

**Вывод**

В ходе выполнения лабораторной работы была разработана программа, выполняющая задания Лабораторной работы №2. В процессе выполнения работы были использованы знания о расчёте сложности алгоритмов и затратах времени в зависимости от величины входных данных.

**Листинг**

**Задание 1**

//Задание 1

constint NUM\_SIZES = 7;

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<malloc.h>

#include<conio.h>

int main(void) {

constint SIZES[NUM\_SIZES] = { 100, 200, 400, 1000, 2000, 4000, 10000 };

inti = 0, j = 0, r = 0, elem\_c = 0;

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

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

for (intsID = 0; sID< NUM\_SIZES; sID++) {

int SIZE = SIZES[sID];

int\*\* a = (int\*\*)malloc(SIZE \* sizeof(int\*));

int\*\* b = (int\*\*)malloc(SIZE \* sizeof(int\*));

int\*\* c = (int\*\*)malloc(SIZE \* sizeof(int\*));

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

a[i] = (int\*)malloc(SIZE \* sizeof(int));

b[i] = (int\*)malloc(SIZE \* sizeof(int));

c[i] = (int\*)malloc(SIZE \* sizeof(int));

}

clock\_tstart, end; // объявляем переменные для определения времени выполнения

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

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

for (j = 0; j < SIZE; j++) {

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

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

}

}

start = clock();

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

for (j = 0; j < SIZE; j++) {

elem\_c = 0;

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

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

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

}

}

end = clock();

printf("%d\t%f\n", SIZE, (double)(end - start) \* 1000.0 / CLOCKS\_PER\_SEC);

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

free(a[i]);

free(b[i]);

free(c[i]);

}

free(a);

free(b);

free(c);

}

return 0;

}

**Задание 2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <iomanip>

using namespace std;

const int size\_m = 100000;

int a[size\_m], b[size\_m], c[size\_m], d[size\_m], e[size\_m];

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) //вызов функции: qs(items, 0, count-1);

{

int i, j;

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);

}

void random() {

int i = 0;

while (i < size\_m)

{

a[i] = rand() % size\_m + 1; // заполняем массив случайными числами

i++;

}

}

void vozr()

{

int i = 0;

while (i < size\_m)

{

a[i] = i; // заполняем массив случайными числами

i++;

}

}

void ybiv()

{

int i = 0;

while (i < size\_m)

{

a[i] = size\_m - i; // заполняем массив случайными числами

i++;

}

}

void polov() {

int i = 0;

while (i < size\_m / 2)

{

a[i] = size\_m - i; // заполняем массив случайными числами

i++;

}

while (i < size\_m)

{

a[i] = size\_m - (i - size\_m / 2); // заполняем массив случайными числами

i++;

}

}

int compare(const void\* a, const void\* b)

{

return (\*(int\*)a - \*(int\*)b);

}

int main(void)

{

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

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

setlocale(LC\_ALL, "RUS");

int i = 0, j = 0, r;

cout << "Размер массива " << size\_m << endl;

clock\_t start, end;

cout << setw(12) << "Shell" << setw(3) << "QS" << setw(9) << "qsort" << endl;

double time = 0.0;

random();

start = clock();

shell(a, size\_m);

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "random " << setw(2) << time;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

random();

start = clock();

qs(a, 0, size\_m);

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << setw(6) << time;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

random();

int n = sizeof(a) / sizeof(a[0]);

start = clock();

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

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << setw(6) << time << endl;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

vozr();

start = clock();

shell(a, size\_m);

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "vozras " << setw(2) << time;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

vozr();

start = clock();

qs(a, 0, size\_m);

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << setw(6) << time;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

vozr();

n = sizeof(a) / sizeof(a[0]);

start = clock();

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

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << setw(6) << time << endl;

time = 0.0;

start, end = 0;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

ybiv();

start = clock();

shell(a, size\_m);

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "ybivan " << setw(2) << time;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

ybiv();

start = clock();

qs(a, 0, size\_m);

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << setw(6) << time;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

ybiv();

n = sizeof(a) / sizeof(a[0]);

start = clock();

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

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << setw(6) << time << endl;

time = 0.0;

start, end = 0;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

polov();

start = clock();

shell(a, size\_m);

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "polovi " << setw(2) << time;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

polov();

start = clock();

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << setw(6) << 0.001;

time = 0.0;

start, end = 0;

memset(a, 0, size\_m);

polov();

n = sizeof(a) / sizeof(a[0]);

start = clock();

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

end = clock();

time = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << setw(6) << time << endl;

time = 0.0;

start, end = 0;

return(0);}