Министерство образования РоссийскойФедерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по теме «Оценка времени выполнения программ»

Выполнили:

Студенты группы 21ВВ2

Назаров Е.А.

Макаров И.С.

Принял:

Митрохин М.А.

Юрова О.В.

Пенза 2021

**Цель работы:**

Приобретение и закрепление навыков оценивания времени выполнения алгоритмов.

**Лабораторные работы:**

**Задание 1:**

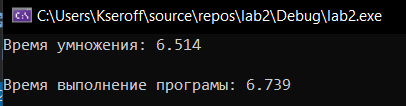
1. Вычислить порядок сложности программы (*О*-символику).
2. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.
3. Построить график зависимости времени выполнения программы от 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.

**Результат работы программы:**

**Задание 1:**

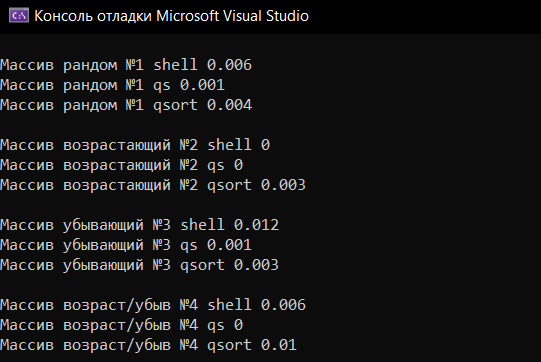


Программа на выходе выдает время выполнения программы и кода, выполняющего перемножение матриц.

Порядок сложности программы: O(n) + O(n\*logn) + O(2\*n^3) есть сложность О (n^3)

|  |  |  |
| --- | --- | --- |
| № теста. | Количество элементов. | Время выполнения сортировки в секундах. |
| 1 | 100 | 0.065 |
| 2 | 200 | 0.083 |
| 3 | 400 | 0.203 |
| 4 | 1 000 | 6.703 |
| 5 | 2 000 | 40.491 |
| 6 | 4 000 | 123.184 |
| 7 | 10 000 | 660.223 |

**Задание 2:**



Программа на выходе выдает время выполнения сортировки 3 видов: shellsort, quicksort, qsort (стандартная функция).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество элементов | 1 массив | 2 массив | 3 массив | 4 массив |
| |  |  |  | | --- | --- | --- | | shell | qs | qsort | | |  |  |  | | --- | --- | --- | | shell | qs | qsort | | |  |  |  | | --- | --- | --- | | shell | qs | qsort | | |  |  |  | | --- | --- | --- | | shell | qs | qsort | |
| 100 | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | |
| 200 | |  |  |  | | --- | --- | --- | | 0 | 0 | 0.001 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | |
| 400 | |  |  |  | | --- | --- | --- | | 0.001 | 0 | 0/001 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0.001 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | | |  |  |  | | --- | --- | --- | | 0 | 0.001 | 0.001 | |
| 1 000 | |  |  |  | | --- | --- | --- | | 0.001 | 0.001 | 0.001 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0.001 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | | |  |  |  | | --- | --- | --- | | 0 | 0.001 | 0.001 | |
| 2 000 | |  |  |  | | --- | --- | --- | | 0.001 | 0.001 | 0.003 | | |  |  |  | | --- | --- | --- | | 0 | 0 | 0.001 | | |  |  |  | | --- | --- | --- | | 0.001 | 0 | 0 | | |  |  |  | | --- | --- | --- | | 0.001 | 0.003 | 0.002 | |
| 4 000 | |  |  |  | | --- | --- | --- | | 0.005 | 0.002 | 0.005 | | |  |  |  | | --- | --- | --- | | 0.001 | 0 | 0.002 | | |  |  |  | | --- | --- | --- | | 0.003 | 0.001 | 0.002 | | |  |  |  | | --- | --- | --- | | 0.002 | 0.014 | 0.005 | |
| 10 000 | |  |  |  | | --- | --- | --- | | 0.006 | 0.003 | 0.001 | | |  |  |  | | --- | --- | --- | | 0 | 0.001 | 0.005 | | |  |  |  | | --- | --- | --- | | 0.002 | 0.001 | 0.004 | | |  |  |  | | --- | --- | --- | | 0.001 | 0.020 | 0.007 | |

По итогам измерений мы приходим к выводу, что самой быстрой сортировкой является qs(быстрая сортировка) реализованная нами, а самой медленной сортировкой оказалась стандартная функция qsort.

**Листинг:**

Lab2.cpp (1задание)

#include <iostream>

#include <ctime>

using namespace std;

int main(void)

{

setlocale(0, "");

srand(time(NULL));

int n=1000, m = n;

int\*\* A = new int\* [n];

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

{

A[i] = new int[m];

}

int\*\* B = new int\* [n];

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

{

B[i] = new int[m];

}

int\*\* C = new int\* [n];

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

{

C[i] = new int[m];

}

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

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

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

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

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

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

double time1 = clock();

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

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

C[i][j] = 0;

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

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

}

}

double time2 = clock();

cout << "Время умножения: " << (time2 - time1) / 1000.0 << endl;

cout << endl << "Время выполнение програмы: " << clock() / 1000.0;

cin.get();

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

delete[] A[i];

delete[] A;

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

delete[] B[i];

delete[] B;

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

delete[] C[i];

delete[] C;

return 0;

}

Lab2.1.cpp (2 задание)

// lab2.1.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include <cstdlib>

#include <iomanip>

#include <fstream>

using namespace std;

ofstream fout("txt.txt");

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 osnova(int n) {

int\* arr11 = new int[n];

int\* arr12 = new int[n];

int\* arr13 = new int[n];

int\* arr21 = new int[n];

int\* arr22 = new int[n];

int\* arr23 = new int[n];

int\* arr31 = new int[n];

int\* arr32 = new int[n];

int\* arr33 = new int[n];

int\* arr41 = new int[n];

int\* arr42 = new int[n];

int\* arr43 = new int[n];

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;

}

}

fout << "кол-во элмементов: " << n << endl;

fout << setw(15) << " " << setw(15) << "shell" << setw(15) << "qs" << setw(15) << "qsort";

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

double time1 = clock();

shell(arr11, n);

double time2 = clock();

fout << endl << setw(15) << "Рандом" << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

time1 = clock();

qs(arr12, 0, n - 1);

time2 = clock();

fout << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

time1 = clock();

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

time2 = clock();

fout << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

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

time1 = clock();

shell(arr21, n);

time2 = clock();

fout << endl << setw(15) << "Вызростание" << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

time1 = clock();

qs(arr22, 0, n - 1);

time2 = clock();

fout << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

time1 = clock();

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

time2 = clock();

fout << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

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

time1 = clock();

shell(arr31, n);

time2 = clock();

fout << endl << setw(15) << "Убывание" << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

time1 = clock();

qs(arr32, 0, n - 1);

time2 = clock();

fout << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

time1 = clock();

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

time2 = clock();

fout << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

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

time1 = clock();

shell(arr41, n);

time2 = clock();

fout << endl << setw(15) << "Возраст/убыван" << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

time1 = clock();

qs(arr42, 0, n - 1);

time2 = clock();

fout << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

time1 = clock();

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

time2 = clock();

fout << setw(15) << (time2 - time1) / CLOCKS\_PER\_SEC;

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

}

int main(void)

{

setlocale(0, "");

osnova(100);

fout << endl << endl << endl;

osnova(200);

fout << endl << endl << endl;

osnova(400);

fout << endl << endl << endl;

osnova(1000);

fout << endl << endl << endl;

osnova(2000);

fout << endl << endl << endl;

osnova(4000);

fout << endl << endl << endl;

cout << "Результат записан в файл";

fout.close();

cin.get();

return 0;

}

**Вывод:** Мы приобрели и закрепили навыки оценивания времени выполнения алгоритмов.