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»

**Выполнили студенты группы 21вв1:**

Кирьянов В.Е.

Аляев А.О.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

**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.

**Листинг:**

**Задание 1:**

#include"StdAfx.h"

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include"iostream"

#pragmacomment(linker, "/STACK:1468006400")

usingnamespace std;

int main(void)

{

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

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

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

int i=0, j=0, r;

int a[12000][12000], b[12000][12000], c[12000][12000], elem\_c;

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

while(i<200)

{

while(j<200)

{

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<200)

{

while(j<200)

{

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

j++;

}

i++;

}

start = clock();

for(i=0;i<200;i++)

{

for(j=0;j<200;j++)

{

elem\_c=0;

for(r=0;r<200;r++)

{

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

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

}

}

}

end = clock();

double time = (end-start) / 1000.0;

printf("Time: %lf", time);

}

**Задание 2:**

**Сортировка Шелла:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include"StdAfx.h"

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include"Windows.h"

#include"iostream"

clock\_t start, end;

void shellSort(int \*num, int size)

{

start = clock();

int increment = 3;

while (increment > 0)

{

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

{

int j = i;

int temp = num[i];

while ((j >= increment) && (num[j - increment] > temp))

{

num[j] = num[j - increment];

j = j - increment;

}

num[j] = temp;

}

if (increment > 1)

increment = increment / 2;

elseif (increment == 1)

break;

}

end = clock();

}

int main()

{

int m[10];

for (int i = 0; i<10; i++)

{

m[i]=rand()%1000+10000;

}

shellSort(m, 10);

double time = (end-start) / 1000.0;

printf("\nTime: %.50lf", time);

getchar();

return 0;

}

#include"StdAfx.h"

**Быстрая сортировка:**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define SIZE 50000

time\_t start, end;

void quickSort(int \*numbers, int left, int right)

{

int pivot;

int l\_hold = left;

int r\_hold = right;

pivot = numbers[left];

while (left < right)

{

while ((numbers[right] >= pivot) && (left < right))

right--;

if (left != right)

{

numbers[left] = numbers[right];

left++;

}

while ((numbers[left] <= pivot) && (left < right))

left++;

if (left != right)

{

numbers[right] = numbers[left];

right--;

}

}

numbers[left] = pivot;

pivot = left;

left = l\_hold;

right = r\_hold;

if (left < pivot)

quickSort(numbers, left, pivot - 1);

if (right > pivot)

quickSort(numbers, pivot + 1, right);

}

int main()

{

int a[SIZE];

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

a[i] = rand() % 201 - 100 + i;

//a[i] = rand() % 201 - 100 - i;

printf(" %d ", a[i]);

}

start = clock();

quickSort(a, 0, SIZE-1);

end = clock();

double time = (end-start) / 1000.0;

printf("Time: %lf", time);

getchar();

return 0;

}

**Ход работы:**

**Задание 1:**

1. Так как программа имеет 3 вложенных цикла for для перемножения матриц, то сложность алгоритма оценивается как O().
2. Оценили время выполнения программы.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Колл-во элементов | **100** | **200** | **400** | **1000** | **2000** | **4000** | **10000** |
| Время, с | **0,001** | **0,014** | **0,138** | **2,87** | **31,994** | **301,884** | **12583,798** |
| Теоретическая оценка, с | **0,002** | **0,018** | **0,14** | **2,325** | **18,604** | **148,837** | **232,558** |

1. Построили график зависимости времени выполнения программы от размера матриц.

**Задание 2:**

1. Оценили время работы сортировок, реализующих алгоритмы быстрой сортировки и сортировки Шелла на указанных наборах данных.

|  |  |  |  |
| --- | --- | --- | --- |
| Исходные данные | Сортировка Шелла, с | Быстрая сортировка, с | Функция qsort(), с |
| Случайный набор значений | 0,025 | 0,001 | 0,002 |
| Возрастающая последовательность | 0,002 | 0,0015 | 0,0026 |
| Убывающая последовательность | 0,034 | 0,006 | 0,0045 |
| Половина возрастающая, половина убывающая | 0,002 | 0,031 | 0,025 |

**Вывод:** в ходе выполнения лабораторной работы были разработаны программы, выполняющие работу алгоритмами сортировки и перемножения матриц: вычисляли сложность алгоритма, рассчитывали время его выполнения. После тестов было выяснено, что для случайного набора значений, возрастающей и убывающей последовательности наиболее выгодно использовать быструю сортировку, а для возрастающе-убывающей последовательности – сортировка Шелла.

Получили опыт в создании проектов в среде Microsoft Visual Studio, научились писать и отлаживать программы с использованием структур и массивов на языке Си.