Министерство науки и высшего 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 24ВВВ3

Агапов И.А.

Любченко В.К.

Азаров М.С.

Приняли:

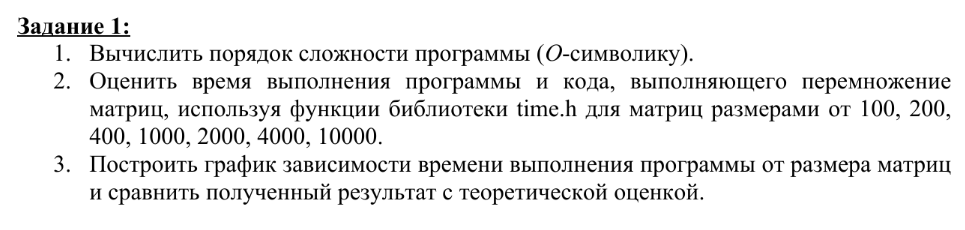
к.т.н., доцент Юрова О.В.

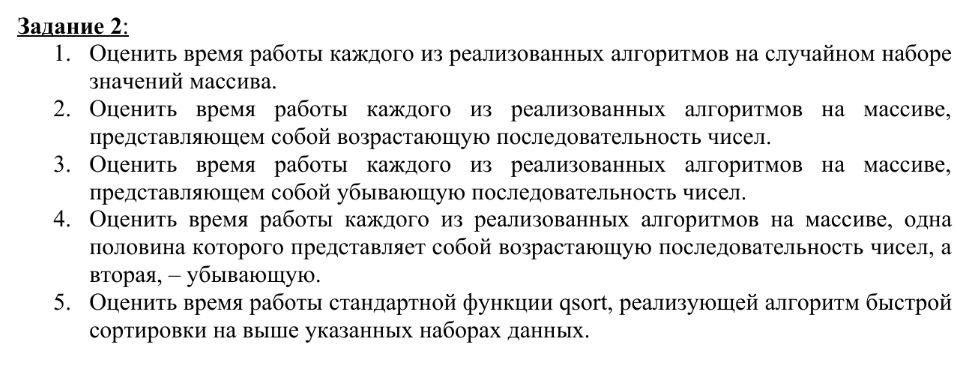
к.т.н., Деев М.В.

Пенза 2025

**Цель работы** – Освоение методов экспериментальной оценки времени выполнения программ на языке С# на примере алгоритма перемножения матриц.

**Лабораторное задание:**

****

****

Метод Random\_matrix:

Два вложенных цикла, каждый до size

Количество операций: size \* size = size²

Сложность: O(n²)

Метод multiplication: Три вложенных цикла, каждый до size

Количество операций: size \* size \* size = size³

Сложность: O(n³)

Общая сложность программы:

Два вызова Random\_matrix: 2 \* O(n²)

Один вызов multiplication: O(n³)

При больших n доминирует кубическая сложность

Итоговая сложность: O(n³)

Для программы с размером матрицы 200:

Количество операций умножения: 200³ = 8,000,000

Количество операций заполнения: 2 \* 200² = 80,000

Таким образом, асимптотическая сложность программы определяется операцией умножения матриц и составляет O(n³).

|  |  |  |
| --- | --- | --- |
| **Размер матрицы** | **Время выполнения(мс)** | **Предполагаемое время(мс)** |
| 100 | 5 | 5 |
| 200 | 36 | 40 |
| 400 | 303 | 320 |
| 1000 | 4679 | 5000 |
| 2000 | 39875 | 40000 |
| 4000 | 360745 | 320000 |
| 10000 | 5206892 | 5000000 |

**Код программы**

**C#**

using System;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Net.NetworkInformation;

using System.Collections.Generic;

namespace second

{

class Program

{

static void PrintTableHeader()

{

Console.WriteLine("┌────────────┬─────────────┬─────────────┬─────────────┬─────────────┬─────────────┐");

Console.WriteLine("│ Размер │ Тип данных │ Shell Sort │ Quick Sort │ Array.Sort │ Отношение │");

Console.WriteLine("│ массива │ │ (мс) │ (мс) │ (мс) │ QS/Array │");

Console.WriteLine("├────────────┼─────────────┼─────────────┼─────────────┼─────────────┼─────────────┤");

}

static void PrintTableRow(int size, string dataType, double shellTime, double qsTime, double arrayTime)

{

double ratio = arrayTime > 0 ? qsTime / arrayTime : 0;

Console.WriteLine($"│ {size,-10} │ {dataType,-11} │ {shellTime,-11:F4} │ {qsTime,-11:F4} │ {arrayTime,-11:F4} │ {ratio,-11:F2} │");

}

static void PrintTableFooter()

{

Console.WriteLine("└────────────┴─────────────┴─────────────┴─────────────┴─────────────┴─────────────┘");

}

static int[,] Random\_matrix(int[,] a, int size)

{

Random rnd = new Random();

int i = 0, j = 0;

while (i < size)

{

while (j < size)

{

a[i, j] = rnd.Next(0, 99999);

j++;

}

i++;

}

return a;

}

static int[,] multiplication(int[,] a, int[,] b, int[,] c, int size)

{

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

{

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

{

int element\_c = 0;

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

{

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

}

}

}

return c;

}

static int[] random(int[] a, int size)

{

Random rnd = new Random();

for (int i = 0; i < size - 1; i++)

{

a[i] = rnd.Next(0, 99999);

}

return a;

}

static void shell(int[] a)

{

Stopwatch sw = new Stopwatch();

sw.Start();

Random rand = new Random();

int gap = 1, k, x;

while (gap < a.Length / 3)

{

//gap = (int)(Math.Pow(3, gap) + 1);

gap = 3 \* gap + 1;

}

while (gap >= 1)

{

// Сортировка вставками с текущим шагом

for (int i = gap; i < a.Length; i++)

{

int temp = a[i];

int j;

for (j = i; j >= gap && a[j - gap] > temp; j -= gap)

{

a[j] = a[j - gap];

}

a[j] = temp;

}

gap /= 3;

}

sw.Stop();

}

static int[] IncreasingRand(int[] a)

{

Random rand = new Random();

a[0] = rand.Next(0, 10);

for (int i = 1; i < a.Length - 1; i++)

{

a[i] = rand.Next(a[i - 1] + 1, 99999 - a.Length + i + 1);

}

return a;

}

static int[] DecreaseRand(int[] a)

{

Random rand = new Random();

a[0] = rand.Next(90000, 99999);

for (int i = 1; i < a.Length - 1; i++)

{

a[i] = rand.Next(a.Length - i, a[i - 1]);

}

return a;

}

static int[] RandHalf(int[] a)

{

Random rand = new Random();

a[0] = rand.Next(0, 10);

int mid = a.Length / 2;

for (int i = 1; i < mid; i++)

{

a[i] = rand.Next(a[i - 1] + 1, 99999 - a.Length + i + 1);

}

for (int i = mid; i < a.Length; i++)

{

a[i] = rand.Next(a.Length - i, a[i - 1]);

}

return a;

}

static void qs(int[] a, int left, int right)

{

int i = left;

int j = right;

int pivot = a[(left + right) / 2]; // Выбор опорного элемента (середина массива)

do

{

while (a[i] < pivot && i < right) i++; // Поиск элемента больше опорного слева

while (pivot < a[j] && j > left) j--; // Поиск элемента меньше опорного справа

if (i <= j)

{

// Обмен элементов

int temp = a[i];

a[i] = a[j];

a[j] = temp;

i++;

j--;

}

} while (i <= j);

// Рекурсивные вызовы для подмассивов

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

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

}

static void print(int[] a)

{

Console.WriteLine("Массив по shell: ");

for (int i = 0; i < a.Length; i++)

{

Console.Write(a[i] + " ");

}

Console.WriteLine();

}

static void Main()

{

for (int i = 1; i < 7; i++)

{

if (i == 1) Console.WriteLine("Введите размер матрицы:");

else Console.WriteLine($"Набор данных: {i}, Введите размер матрицы:");

int size\_for\_matrix = Convert.ToInt32(Console.ReadLine());

int[,] a = new int[size\_for\_matrix, size\_for\_matrix];

int[,] b = new int[size\_for\_matrix, size\_for\_matrix];

int[,] c = new int[size\_for\_matrix, size\_for\_matrix];

Stopwatch sw\_for\_matrix = new Stopwatch();

sw\_for\_matrix.Start();

Random\_matrix(a, size\_for\_matrix);

Random\_matrix(b, size\_for\_matrix);

multiplication(a, b, c, size\_for\_matrix);

sw\_for\_matrix.Stop();

Console.WriteLine($"Время: {sw\_for\_matrix.ElapsedMilliseconds} мс, размер матрицы: {size\_for\_matrix}");

}

// Вторая часть с сортировками в виде таблицы

List<int> sizes = new List<int>();

Console.WriteLine("\nВведите 7 размеров массивов для второй части:");

for (int i = 0; i < 7; i++)

{

sizes.Add(Convert.ToInt32(Console.ReadLine()));

}

PrintTableHeader();

foreach (int size in sizes)

{

int[] originalArray = new int[size];

int[] testArray = new int[size];

Stopwatch sw = new Stopwatch();

// Тест 1: Случайные данные

random(originalArray, size);

Array.Copy(originalArray, testArray, size);

sw.Restart();

shell(testArray);

sw.Stop();

double shellTime1 = sw.Elapsed.TotalMilliseconds;

Array.Copy(originalArray, testArray, size);

sw.Restart();

qs(testArray, 0, testArray.Length - 1);

sw.Stop();

double qsTime1 = sw.Elapsed.TotalMilliseconds;

Array.Copy(originalArray, testArray, size);

sw.Restart();

Array.Sort(testArray);

sw.Stop();

double arrayTime1 = sw.Elapsed.TotalMilliseconds;

PrintTableRow(size, "Случайный", shellTime1, qsTime1, arrayTime1);

// Тест 2: Возрастающие данные

IncreasingRand(originalArray);

Array.Copy(originalArray, testArray, size);

sw.Restart();

shell(testArray);

sw.Stop();

double shellTime2 = sw.Elapsed.TotalMilliseconds;

Array.Copy(originalArray, testArray, size);

sw.Restart();

qs(testArray, 0, testArray.Length - 1);

sw.Stop();

double qsTime2 = sw.Elapsed.TotalMilliseconds;

Array.Copy(originalArray, testArray, size);

sw.Restart();

Array.Sort(testArray);

sw.Stop();

double arrayTime2 = sw.Elapsed.TotalMilliseconds;

PrintTableRow(size, "Возрастающий", shellTime2, qsTime2, arrayTime2);

// Тест 3: Убывающие данные

DecreaseRand(originalArray);

Array.Copy(originalArray, testArray, size);

sw.Restart();

shell(testArray);

sw.Stop();

double shellTime3 = sw.Elapsed.TotalMilliseconds;

Array.Copy(originalArray, testArray, size);

sw.Restart();

qs(testArray, 0, testArray.Length - 1);

sw.Stop();

double qsTime3 = sw.Elapsed.TotalMilliseconds;

Array.Copy(originalArray, testArray, size);

sw.Restart();

Array.Sort(testArray);

sw.Stop();

double arrayTime3 = sw.Elapsed.TotalMilliseconds;

PrintTableRow(size, "Убывающий", shellTime3, qsTime3, arrayTime3);

// Тест 4: Половина отсортирована

RandHalf(originalArray);

Array.Copy(originalArray, testArray, size);

sw.Restart();

shell(testArray);

sw.Stop();

double shellTime4 = sw.Elapsed.TotalMilliseconds;

Array.Copy(originalArray, testArray, size);

sw.Restart();

qs(testArray, 0, testArray.Length - 1);

sw.Stop();

double qsTime4 = sw.Elapsed.TotalMilliseconds;

Array.Copy(originalArray, testArray, size);

sw.Restart();

Array.Sort(testArray);

sw.Stop();

double arrayTime4 = sw.Elapsed.TotalMilliseconds;

PrintTableRow(size, "Половина", shellTime4, qsTime4, arrayTime4);

Console.WriteLine("├────────────┼─────────────┼─────────────┼─────────────┼─────────────┼─────────────┤");

}

PrintTableFooter();

Console.ReadKey();

}

}

}

**Результаты работы программы**

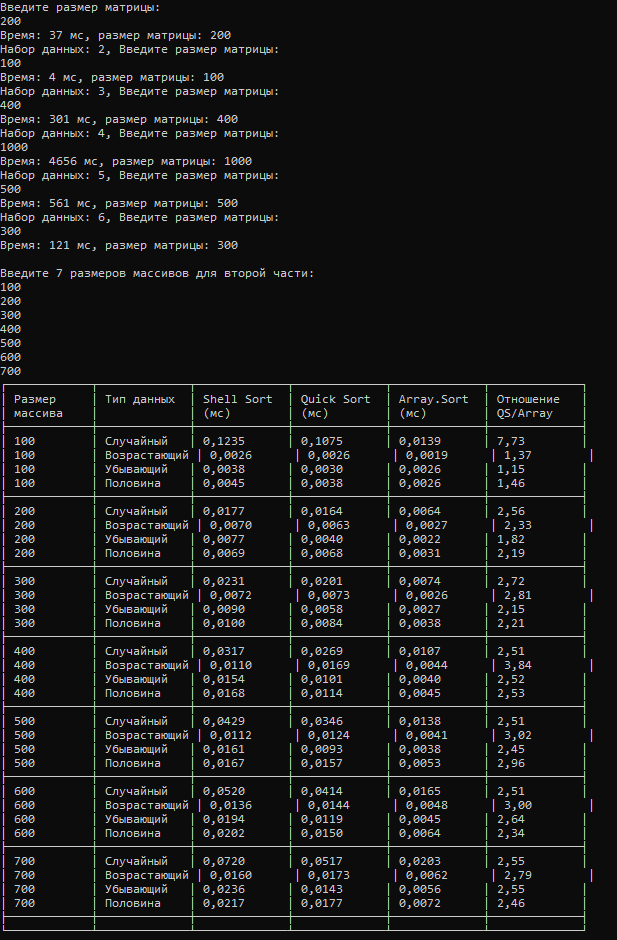


Рисунок 1 - Результат работы программы

**Вывод:** В ходе данной лабораторной работы были успешно освоены методы экспериментальной оценки времени выполнения программ на языке С#. Была реализована программа, выполняющая перемножение квадратных матриц.