Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Факультет безопасности (ФБ)

Кафедра безопасности информационных систем (БИС)

Отчет по практической работе №3

По дисциплине «Численные Методы»

|  |  |  |
| --- | --- | --- |
|  |  | Студент гр. 731-1 |
|  | \_\_\_\_\_\_\_\_\_ | Брода И.А. |
|  | \_\_\_\_\_\_\_\_\_ |  |
|  |  | Преподаватель |
| \_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_ | Якимук А.Ю. |
|  | \_\_\_\_\_\_\_\_\_ |  |

Томск 2022

Оглавление

[1 Введение 3](#_Toc118522751)

[2 Ход работы 4](#_Toc118522752)

[2.1 Вычисление точек 4](#_Toc118522753)

[2.2 Быстрая сортировка 6](#_Toc118522754)

[2.3 Сортировка выбором 7](#_Toc118522755)

[3 Худший и лучший случай 10](#_Toc118522756)

[Заключение 12](#_Toc118522757)

[Приложение А 13](#_Toc118522758)

[Приложение Б 14](#_Toc118522759)

[Приложение В 16](#_Toc118522760)

# 1 Введение

Целью работы является получение навыков вычисления интерполяционного полинома Лагранжа и аппроксимирующей функции.

# 2 Ход работы

## 2.1 Вычисление точек

Необходимо вычислить 14 точек с координатами (xi; yi).

xi = 0; 2 i

i = 0; 13

yi = f(xi)

f(xi) – индивидуальная функция. Индивидуальная функция имеет вид:

y = sin(1,8x2 + 2,12)

## 2.2 Вычисление полинома Лагранжа

Листинг программы представлен в Приложении А.

На рисунке 2.1 представлен результат работы программы

Рисунок 2.1 – Результат быстрой сортировки

В приложение Б, представлен листинг быстрой сортировки.

На рисунках представлены графики зависимости количества тактов от размерности массива (Рисунок 2.5) и количества операций от размерности массива (Рисунок 2.6).

Рисунок 2.5 - Зависимость тактов от размерности массива

Рисунок 2.6 – Зависимость количество операций от размерности массива

## 2.3 Сортировка выбором

На рисунке 2.7 представлен результат работы программы по вычислению алгоритмической сложности алгоритма сортировки выбором.

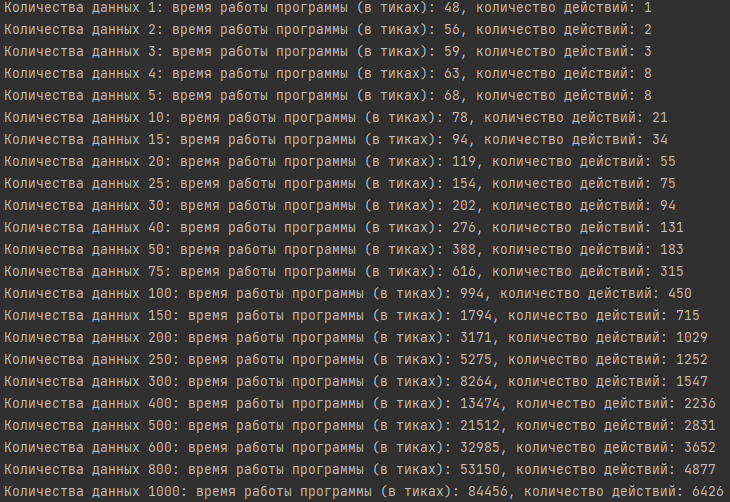


Рисунок 2.7 – Результат сортировки выбором

В приложение В представлен листинг сортировки выбором.

На рисунках представлены графики зависимости количества тактов от размерности массива (Рисунок 2.8) и количества операций от размерности массива (Рисунок 2.9).

Рисунок 2.8 - Зависимость тактов от размерности массива

Рисунок 2.9 – Зависимость количество операций от размерности массива

# 3 Худший и лучший случай

После постройки графиков были проведены исследования на худший и лучший случай для каждой сортировки на массиве из 100 элементов, было проведено по 5 экспериментов для каждой.

Результаты для сортировки расческой представлены в Таблице 3.1.

Таблица 3.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 3 | 4 | 5 |
| Худший | Кол-во операций | 1339 | 1339 | 1339 | 1339 | 1339 |
| Время в тактах | 220 | 190 | 176 | 165 | 178 |
| Лучший | Кол-во операций | 1233 | 1233 | 1233 | 1233 | 1233 |
| Время в тактах | 210 | 192 | 187 | 178 | 215 |

0,6340498884916622, -1,816262939025907, 1,0009575413553733

Результаты для быстрой сортировки представлены в Таблице 3.2.

Таблица 3.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 3 | 4 | 5 |
| Худший | Кол-во операций | 1255 | 1222 | 1317 | 1206 | 1190 |
| Время в тактах | 4200 | 3320 | 3152 | 2314 | 2700 |
| Лучший | Кол-во операций | 1100 | 1107 | 1092 | 1125 | 1089 |
| Время в тактах | 3456 | 12275 | 3659 | 3439 | 3256 |

Результаты для сортировки выбором представлены в Таблице 3.3.

Таблица 3.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 3 | 4 | 5 |
| Худший | Кол-во операций | 2305 | 2325 | 2331 | 2257 | 2317 |
| Время в тактах | 629 | 583 | 427 | 494 | 430 |
| Лучший | Кол-во операций | 2695 | 2456 | 2405 | 2660 | 2732 |
| Время в тактах | 442 | 696 | 426 | 649 | 600 |

Если смотреть на результаты сортировок можно увидеть, что сортировка расческой оказалась эффективней, чем другие.

# Заключение

В ходе выполнения практической работы были приобретены навыки анализа результатов работы алгоритмов сортировки на основе использования различных входных данных, а также использования графиков для отображения результатов и анализа их построения. Была проделана работа по написанию автоматического создания массивов разной длинны, подсчетов времени работы сортировок. Были проведены эксперименты с худшим и лучшим случаем сортировки. Исходя из результатов экспериментов можно сказать, что практические показания совпали с теоретическими.

# Приложение А

Сортировка расческой

using System;

using System.Diagnostics;

namespace ConsoleApp

{

class Program

{

public static Stopwatch Timer = new Stopwatch();

public static Random rand = new Random();

static void Main(string[] args)

{

int[] array = new int[] { 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 50, 75, 100, 150, 200, 250, 300, 400, 500, 600, 800, 1000 };

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

{

comb(array[i]);

}

}

static void comb(int n)

{

int[] array = new int[n];

Timer.Start();

double factor = 1.2473309;

int step = n - 1;

int compare = 0;

while (step >= 1)

{

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

{

if (array[i] > array[i + step])

{

int temp = array[i]; array[i] = array[i + step]; array[i + step] =

temp;

compare = compare + 1;

}

compare++;

}

step = (int)(step / factor);

}

Timer.Stop();

Console.WriteLine("Количества данных {0}: время работы программы (в тиках): {1}, количество действий: {2}", n, Timer.ElapsedTicks, compare);

}

}

}

# Приложение Б

Быстрая сортировка

using System;

using System.Diagnostics;

namespace QuickSortAlgorithm

{

class Program

{

static public Stopwatch Timer = new Stopwatch();

static int count = 0;

static void Main()

{

int[] array1 = new int[] { 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 50, 75, 100, 150, 200, 250, 300, 400, 500, 600, 800, 1000 };

Random rand = new Random();

int position = 2;

for (int j = 0; j < array1.Length; j++)

{

int n = array1[j];

int[] array = new int[n];

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

{

array[i] = rand.Next(0, 5000);

array[i] = array[i] \* 2 + 1;

if (position % 2 == 0)

{

array[i] = array[i] \* -1;

}

position++;

}

Timer.Start();

int[] sortedArray = QuickSort(array, 0, array.Length - 1);

Timer.Stop();

Console.WriteLine("Количество данных: {0}: время работы (в тактах): {1}, количество действий: {2}", n, Timer.ElapsedTicks, count);

}

}

private static int[] QuickSort(int[] array, int minIndex, int maxIndex)

{

if (minIndex >= maxIndex)

{

return array;

}

count++;

int pivotIndex = GetPivotIndex(array, minIndex, maxIndex);

QuickSort(array, minIndex, pivotIndex - 1);

QuickSort(array, pivotIndex + 1, maxIndex);

return array;

}

private static int GetPivotIndex(int[] array, int minIndex, int maxIndex)

{

int pivot = minIndex - 1;

for (int i = minIndex; i <= maxIndex; i++)

{

if (array[i] < array[maxIndex])

{

pivot++;

count++;

Swap(ref array[pivot], ref array[i]);

}

count++;

}

pivot++;

count++;

Swap(ref array[pivot], ref array[maxIndex]);

return pivot;

}

private static void Swap(ref int leftItem, ref int rightItem)

{

int temp = leftItem;

leftItem = rightItem;

rightItem = temp;

}

}

}

# Приложение В

Сортировка выбором

**using** **System.Diagnostics**;

**namespace** **ConsoleApp**

{

**class** **Program**

{

**public** **static** Stopwatch Timer = **new** Stopwatch();

**public** **static** Random rand = **new** Random();

**static** **void** **Main**(**string**[] args)

{

**int**[] array = **new** **int**[]

{ **1**, **2**, **3**, **4**, **5**, **10**, **15**, **20**, **25**, **30**, **40**, **50**, **75**, **100**, **150**, **200**, **250**, **300**, **400**, **500**, **600**, **800**, **1000** };

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

{

SelectionSort(array[i]);

}

Console.ReadLine();

**static** **void** **SelectionSort**(**int** n)

{

**int**[] mas = **new** **int**[n];

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

{

mas[i] = rand.Next(**0**, **99999**);

}

**int** compare = **0**;

Timer.Start();

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

{

**int** min = i;

**for** (**int** j = i + **1**; j < mas.Length; j++)

{

**if** (mas[j] < mas[min])

{

min = j;

compare++;

}

}

**int** temp = mas[min];

mas[min] = mas[i];

mas[i] = temp;

compare++;

}

Timer.Stop();

Console.WriteLine("Количества данных {0}: время работы программы (в тиках): {1}, количество действий: {2}",n, Timer.ElapsedTicks, compare);

}

}