Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа №16
«Вычисление метрических характеристик реализаций алгоритмов»

Выполнил:
Студент группы ИП-013
Копытина Т.А.
Работу проверил:
ассистент кафедры ПМиК
Агалаков А.А.

Новосибирск 2023 г.

Содержание

1. Задание

1. Написать подпрограммы на двух языках программирования для решения следующих задач:

ЗАДАЧА
1. Отыскать минимальный элемент одномерного массива целых, его значение и значение его индекса.
2. Сортировка одномерного массива в порядке возрастания методом пузырька.
3. Бинарный поиск элемента в упорядоченном одномерном массиве.
4. Отыскать минимальный элемент двумерного массива целых, его значение и значение его индексов.
5. Осуществить перестановку значений элементов одномерного массива в обратном порядке.
6. Осуществлять циклический сдвиг элементов одномерного массива на заданное число позиций влево.
7. Заменить все вхождения целочисленного значения в целочисленный массив.

2. Для каждой подпрограммы вычислить следующие метрические характеристики:

- η^*_2 – число единых по смыслу входных и выходных параметров, представленных в сжатой без избыточной форме;
- η_1 - число отдельных операторов;
- η_2 - число отдельных операндов;
- η - длина словаря реализации;
- N_1 - общее число вхождений всех операторов в реализацию;
- N_2 - общее число вхождений всех операндов в реализацию;
- N - длина реализации;
- $\hat{N}$ - предсказанная длина реализации по соотношению Холстеда;
- V^* - потенциальный объем реализации:

$$V^* = (2 + \eta_2^*) * \log_2(2 + \eta_2^*).$$

- V - объем реализации:

$$V = N * \log_2 \eta.$$

- L - уровень программы через потенциальный объем:

$$L = V^* / V.$$

- $\hat{L}$ - уровень программы по реализации:

$$\hat{L} = (2/\eta_1) * (\eta_2 / N_2).$$

- I - интеллектуальное содержание программы:

$$I = (2/\eta_1) * (\eta_2 / N_2) * (N_1 + N_2) * \log_2(\eta_1 + \eta_2).$$

- $\hat{T}_1$ - прогнозируемое время написания программы, выраженное через потенциальный объем:

$$\hat{T} = \frac{V^2}{S * V^*}.$$

- $\hat{T}_2$ - прогнозируемое время написания программы, выраженное через длину реализации, найденную по Холстеду (т.е. в предположении, что программа совершенна):

$$\hat{T} = \frac{\eta_1 \times N_2 \times (\eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2) \times \log_2 \eta}{2 \times S \times \eta_2}.$$

- $\hat{T}_3$ - прогнозируемое время написания программы, выраженное через метрические характеристики реализации:

$$\hat{T} = \frac{\eta_1 \times N_2 \times N \times \log_2 \eta}{2 \times S \times \eta_2}.$$

3. По всем реализациям алгоритмов определить средние значения уровней языков программирования λ:

$$\lambda_1 = \hat{L}^2 \times V,$$

$$\lambda_2 = \frac{V^{*2}}{V}.$$

2. Исходный код программы
2.1. Код программы

Program.cs

```
using System;
using System.Linq;
using System.Collections.Generic;


namespace lab16
{


    class Program
    {
        public static (int, int) FindMinimum(int[] array)
        {
            int minValue = int.MaxValue;
            int minIndex = -1;


            for (int i = 0; i < array.Length; i++)
            {
                if (array[i] < minValue)
                {
                    minValue = array[i];
                    minIndex = i;
                }
            }


            return (minValue, minIndex);
        }


        public static void BubbleSort(int[] array)
        {
            for (int i = 0; i < array.Length - 1; i++)
            {
                for (int j = 0; j < array.Length - 1 - i; j++)
                {
                    if (array[j] > array[j + 1])
                    {
                        int temp = array[j];
                        array[j] = array[j + 1];
                        array[j + 1] = temp;
                    }
                }
            }
        }


        public static int BinarySearch(int[] sortedArray, int target)
        {
            int left = 0;
            int right = sortedArray.Length - 1;


            while (left <= right)
            {
                int mid = left + (right - left) / 2;


                if (sortedArray[mid] == target)
                    return mid;


                if (sortedArray[mid] < target)
                    left = mid + 1;
```

```
                else
                    right = mid - 1;
            }

            return -1;
        }

        public static (int, int, int)
FindMinimum2D(int[,] array)
        {
            int minValue = int.MaxValue;
            int minRow = -1;
            int minCol = -1;

            for (int row = 0; row <
array.GetLength(0); row++)
            {
                for (int col = 0; col <
array.GetLength(1); col++)
                {
                    if (array[row, col] <
minValue)
                    {
                        minValue =
array[row, col];
                        minRow = row;
                        minCol = col;
                    }
                }
            }

            return (minValue, minRow,
minCol);
        }

        public static void
ReverseArray(int[] array)
        {
            Array.Reverse(array);
        }

        public static void
RotateArrayLeft(int[] array, int
positions)
        {
            int n = array.Length;
            positions %= n;

            int[] temp = new int[n];

            for (int i = 0; i < n; i++)
            {
                int newPosition = (i -
positions + n) % n;
                temp[newPosition] =
array[i];
            }

            for (int i = 0; i < n; i++)
            {
                array[i] = temp[i];
            }
        }

        public static void
ReplaceAll(int[] array, int oldValue, int
newValue)
        {
            for (int i = 0; i <
array.Length; i++)
            {
                if (array[i] == oldValue)
                {
                    array[i] = newValue;
                }
            }
        }
```

```
        public static void
CalculateMetrics(string taskName, int[]
array, string[] operands, string[]
operators)
        {
            int etaStar2 =
operands.Length + operators.Length;
            int eta1 = operators.Length;
            int eta2 = operands.Length;

            var vocabulary = new
Dictionary<int, int>();
            foreach (int item in array)
            {
                if
(!vocabulary.ContainsKey(item))
                {
                    vocabulary[item] = 1;
                }
                else
                {
                    vocabulary[item] +=
1;
                }
            }

            int N1 = 0;
            int N2 = array.Length;
            foreach (int operand in
vocabulary.Keys)
            {
                N1 +=
vocabulary[operand];
            }

            int N = eta1 + eta2;
            double NHat = etaStar2 *
Math.Log(2 + etaStar2, 2);
            double VStar = (2 + etaStar2)
* Math.Log(2 + etaStar2, 2);
            double V = VStar * N *
Math.Log(2, Math.E);
            double L = V / VStar;
            double LHat = (2 / 3.0) * (N1
+ N2) * Math.Log(2 + (N1 + N2) / 2, 2);
            double I = (2 / 3.0) * (N1 /
N2) * (eta1 + eta2) * Math.Log(2 + (N1 /
N2) * (eta1 + eta2), 2);

            double T1Hat = V / VStar;
            double T2Hat =
(Math.Log(Math.Log(2 + eta1, 2) *
Math.Log(2 + eta2, 2), 2) / 2);
            double T3Hat = Math.Log(2 +
eta1, 2) * Math.Log(2 + eta2, 2) * N1 *
N2 / 2;

            Console.WriteLine("Метрики
для " + taskName + ":");
            Console.WriteLine($"n*2:
{etaStar2}");
            Console.WriteLine($"n1:
{eta1}");
            Console.WriteLine($"n2:
{eta2}");
            Console.WriteLine($"Объем
словарного запаса (n):
{vocabulary.Count}");
            Console.WriteLine($"N1:
{N1}");
            Console.WriteLine($"N2:
{N2}");
            Console.WriteLine($"N: {N}");
            Console.WriteLine($"N^:
{NHat}");
            Console.WriteLine($"V*:
{VStar}");
            Console.WriteLine($"V: {V}");
            Console.WriteLine($"L: {L}");
            Console.WriteLine($"L^:
{LHat}");
            Console.WriteLine($"I: {I}");
            Console.WriteLine($"T^1:
{T1Hat}");
```

```
            Console.WriteLine($"T^2: {T2Hat}");

            Console.WriteLine($"T^3: {T3Hat}");

            Console.WriteLine();


            // Вычисление уровня программы через потенциальный объем (λ1)

            double lambda1 = L * V;


            Console.WriteLine($"(lam)1 (Уровень через потенциальный объем) для {taskName}: {lambda1}");

        }


        public static void Main()

        {

            int[] oneDimensionalArray = { 5, 3, 9, 1, 7 };

            int[,] twoDimensionalArray = { { 5, 3, 9 }, { 1, 7, 2 } };

            int targetValue = 1;


            // Выполнение задачи 1

            var minResult = FindMinimum(oneDimensionalArray);


Console.WriteLine($"Минимальное значение в одномерном массиве: {minResult.Item1}");

            Console.WriteLine($"Индекс минимального значения: {minResult.Item2}");


            string[] task1Operands = { "minValue", "minIndex", "array", "i" };

            string[] task1Operators = { "for", "if" };


            CalculateMetrics("Задачи 1", oneDimensionalArray, task1Operands, task1Operators);


            // Выполнение задачи 2

BubbleSort(oneDimensionalArray);

            Console.WriteLine("Сортировка одномерного массива:");

            foreach (var num in oneDimensionalArray)

            {

                Console.Write(num + " ");

            }

            Console.WriteLine();


            string[] task2Operands = { "temp", "i", "j" };

            string[] task2Operators = { "for" };


            CalculateMetrics("Задачи 2", oneDimensionalArray, task2Operands, task2Operators);


            // Выполнение задачи 3

            int binarySearchResult = BinarySearch(oneDimensionalArray, targetValue);

            if (binarySearchResult != -1)

            {

                Console.WriteLine($"Нашел {targetValue} по индексу {binarySearchResult}");

            }

            else

            {


Console.WriteLine($"{targetValue} не найден в одномерном массиве");

            }


            string[] task3Operands = { "left", "right", "mid" };

            string[] task3Operators = { "if", "while" };
```

```
            CalculateMetrics("Задачи 3",
oneDimensionalArray, task3Operands,
task3Operators);


            // Выполнение задачи 4

            var minResult2D =
FindMinimum2D(twoDimensionalArray);


Console.WriteLine($"Минимальное значение
в двумерном массиве:
{minResult2D.Item1}");

            Console.WriteLine($"Строка
минимального значения:
{minResult2D.Item2}");

            Console.WriteLine($"Столбец с
минимальным значением:
{minResult2D.Item3}");


            string[] task4Operands = {
"minValue", "minRow", "minCol", "array",
"row", "col" };

            string[] task4Operators = {
"if", "for", "nested for" };


            CalculateMetrics("Задачи 4",
oneDimensionalArray, task4Operands,
task4Operators);


            // Выполнение задачи 5


ReverseArray(oneDimensionalArray);

            Console.WriteLine("Обратный
одномерный массив:");

            foreach (var num in
oneDimensionalArray)

            {

                Console.Write(num + " ");

            }

            Console.WriteLine();


            string[] task5Operands = {
"temp", "i", "j" };

            string[] task5Operators = {
"for" };


            CalculateMetrics("Задачи 5",
oneDimensionalArray, task5Operands,
task5Operators);


            // Выполнение задачи 6

            int positions = 2;


RotateArrayLeft(oneDimensionalArray,
positions);


Console.WriteLine($"Вращающийся
одномерный массив {positions} позиции
слева:");

            foreach (var num in
oneDimensionalArray)

            {

                Console.Write(num + " ");

            }

            Console.WriteLine();


            string[] task6Operands = {
"positions", "n", "temp", "i",
"newPosition" };

            string[] task6Operators = {
"for" };


            CalculateMetrics("Задачи 6",
oneDimensionalArray, task6Operands,
task6Operators);


            // Выполнение задачи 7

            int oldValue = 3;

            int newValue = 8;


ReplaceAll(oneDimensionalArray, oldValue,
newValue);


Console.WriteLine($"Одномерный массив со
всеми вхождениями {oldValue} заменен на
{newValue}:");
```

```
            foreach (var num in
oneDimensionalArray)
            {
                Console.Write(num + " ");
            }
            Console.WriteLine();


            string[] task7Operands = {
"oldValue", "newValue", "i" };
            string[] task7Operators = {
"for", "if" };


            CalculateMetrics("Задачи 7",
oneDimensionalArray, task7Operands,
task7Operators);
        }
    }
}
```

3. Результаты работы программы

```
Минимальное значение в одномерном массиве: 1
Индекс минимального значения: 3
Метрики для Задачи 1:
n*2: 6
n1: 2
n2: 4
Объем словарного запаса (n): 5
N1: 5
N2: 5
N: 6
N^: 18
V*: 24
V: 99,81319400063212
L: 4,1588830833596715
L^: 18,715699480384025
I: 12
T^1: 4,1588830833596715
T^2: 1,1850716759730007
T^3: 64,62406251802891

(lam)1 (Уровень через потенциальный объем) для Задачи 1: 415,111404025326
Сортировка одномерного массива:
1 3 5 7 9
Метрики для Задачи 2:
n*2: 4
n1: 1
n2: 3
Объем словарного запаса (n): 5
N1: 5
N2: 5
N: 4
N^: 10,339850002884624
V*: 15,509775004326936
V: 43,002227261473315
L: 2,772588722239781
L^: 18,715699480384025
I: 6,893233335256416
T^1: 2,772588722239781
T^2: 0,9398860015953385
T^3: 46,0021119970923
```

```
(lam)1 (Уровень через потенциальный объем) для Задачи 2: 119,22749033635299
Нашел 1 по индексу 0
Метрики для Задачи 3:
n*2: 5
n1: 2
n2: 3
Объем словарного запаса (n): 5
N1: 5
N2: 5
N: 5
N^: 14,03677461028802
V*: 19,651484454403228
V: 68,10685521693597
L: 3,465735902799727
L^: 18,715699480384025
I: 9,357849740192012
T^1: 3,465735902799727
T^2: 1,1076616478683938
T^3: 58,04820237218405

(lam)1 (Уровень через потенциальный объем) для Задачи 3: 236,04037335211788
Минимальное значение в двумерном массиве: 1
Строка минимального значения: 1
Столбец с минимальным значением: 0
Метрики для Задачи 4:
n*2: 9
n1: 3
n2: 6
Объем словарного запаса (n): 5
N1: 5
N2: 5
N: 9
N^: 31,13488456773568
V*: 38,053747805010275
V: 237,3916320070387
L: 6,238324625039508
L^: 18,715699480384025
I: 20,756589711823786
T^1: 6,238324625039508
T^2: 1,4001428982289719
T^3: 87,07230355827608
```

```
(lam)1 (Уровень через потенциальный объем) для Задачи 4: 1480,9260637278264
Обратный одномерный массив:
9 7 5 3 1
Метрики для Задачи 5:
n*2: 4
n1: 1
n2: 3
Объем словарного запаса (n): 5
N1: 5
N2: 5
N: 4
N^: 10,339850002884624
V*: 15,509775004326936
V: 43,002227261473315
L: 2,772588722239781
L^: 18,715699480384025
I: 6,893233335256416
T^1: 2,772588722239781
T^2: 0,9398860015953385
T^3: 46,0021119970923

(lam)1 (Уровень через потенциальный объем) для Задачи 5: 119,22749033635299
Вращающийся одномерный массив 2 позиции слева:
5 3 1 9 7
Метрики для Задачи 6:
n*2: 6
n1: 1
n2: 5
Объем словарного запаса (n): 5
N1: 5
N2: 5
N: 6
N^: 18
V*: 24
V: 99,81319400063212
L: 4,1588830833596715
L^: 18,715699480384025
I: 12
T^1: 4,1588830833596715
T^2: 1,0768300883460078
T^3: 55,619403470953344
```

```
(lam)1 (Уровень через потенциальный объем) для Задачи 6: 415,111404025326
Одномерный массив со всеми вхождениями 3 заменен на 8:
5 8 1 9 7
Метрики для Задачи 7:
n*2: 5
n1: 2
n2: 3
Объем словарного запаса (n): 5
N1: 5
N2: 5
N: 5
N^: 14,03677461028802
V*: 19,6514844544032 28
V: 68,10685521693597
L: 3,465735902799727
L^: 18,715699480384025
I: 9,357849740192012
T^1: 3,465735902799727
T^2: 1,1076616478683938
T^3: 58,04820237218405

(lam)1 (Уровень через потенциальный объем) для Задачи 7: 236,04037335211788
```

4. Вывод

По итогам данной лабораторной работе были сформированы практические навыки реализации программы для разработки алгоритмов метрических характеристик полученных программ на языке C#.