Министерство науки и высшего образования Российской Федерации

Муромский институт (филиал)

Федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

*ЛАБОРАТОРНАЯ*

*РАБОТА №7*

По Дискретная математика

Тема Исследование алгоритмов поиска

Руководитель

Кульков Я.Ю.

(фамилия, инициалы)

(подпись) (дата)

Студент ПИН - 121

(группа)

Ермилов М.В.

(фамилия, инициалы)

(подпись) (дата)

Муром 2022

**Лабораторная работа №7**

Тема: Исследование алгоритмов поиска

Цель работы: исследовать алгоритмы поиска, практически их реализовать, проанализировать результат выполнения всех алгоритмов.

**Ход работы:**

**Задание 1**

1. Реализовать методы поиска данных в массиве:

a. Реализовать метод прямого поиска.

c. Реализовать метод бинарного поиска с использованием

бинарного дерева поиска

2. Реализовать управляющую программу(ы), включающую:

a. ввод исходных данных: из файла, с консоли или генерацией случайных чисел;

b. ввод на экран исходных данных;

c. вывод на экран результата работы;

d. замер времени выполнения сортировки.

3. Выполнить исследование реализованных трех алгоритмах на разных

коллекциях:

a. Выполнить замеры времени для коллекций размера 10, 100,

1 000, 10 000, 10 000, 1 000 000 000 элементов.

b. Заполнить таблицы 1 для каждого анализируемого алгоритма

(всего три таблицы).

c. Сделать выводы по таблицам и графику.

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

Console.Write("Введите размер массива ");

int n = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите искомый элемент ");

int search = Convert.ToInt32(Console.ReadLine());

InitialArray IA = new InitialArray(n);

SortArray SA = new SortArray(IA);

Console.WriteLine($"Search element {search}:");

int count;

if (InitialArray.linearSearchUnsorted(IA.arr, search, out count) == true)

{

Console.WriteLine($"Element {search} found.\n" +

$"Number of search iterations in the original array {count}");

}

else

{

Console.WriteLine($"Element {search} not found.");

}

if (SortArray.BinarySearch(SA.arr, search, out count) == true)

{

Console.WriteLine($"Number of search iterations in the sorted array {count}");

}

BST tree = new BST();

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

{

tree.Add(IA.arr[i], i);

}

List<int> indices;

if (tree.Search(search, out indices, out count) == true)

{

Console.WriteLine($"Number of search iterations in a binary tree {count}");

}

Console.ReadLine();

}

}

}

using System;

namespace ConsoleApp1

{

internal class InitialArray

{

public int[] arr;

public InitialArray(int value)

{

arr = new int[value];

Random rand = new Random();

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

{

arr[i] = rand.Next(1, 100);

}

}

public static bool linearSearchUnsorted(int[] arr, int search, out int interCount)

{

interCount = 0;

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

{

interCount++;

if (search == arr[i])

{

return true;

}

}

return false;

}

}

}

using System;

namespace ConsoleApp1

{

internal class SortArray

{

public int[] arr;

public SortArray(InitialArray a)

{

arr = a.arr;

Array.Sort(arr);

}

public static bool BinarySearch(int[] arr, int value, out int interCount)

{

interCount = 0;

int low = 0;

int high = arr.Length - 1;

int mid;

while (low <= high)

{

mid = low + (high - low) / 2;

interCount++;

if (arr[mid] == value)

{

return true;

}

else if (arr[mid] < value)

{

low = mid + 1;

}

else

{

high = mid - 1;

}

}

return false;

}

}

}

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

internal class BST

{

BSTItem Root;

public void Clear()

{

Root = null;

}

private void AddRecur(BSTItem cur, int value, int index)

{

if (value < cur.Value)

{

if (cur.Left == null)

{

cur.Left = new BSTItem(value, index);

}

else

{

AddRecur(cur.Left, value, index);

}

}

else if (value > cur.Value)

{

if (cur.Rigth == null)

{

cur.Rigth = new BSTItem(value, index);

}

else

{

AddRecur(cur.Rigth, value, index);

}

}

else

{

cur.ArrayIndices.Add(index);

}

}

public void Add(int value, int index)

{

if (Root == null)

{

Root = new BSTItem(value, index);

}

else

{

AddRecur(Root, value, index);

}

}

public bool Search(int value, out List<int> indices, out int interCount)

{

BSTItem cur = Root;

interCount = 0;

while (cur != null)

{

interCount++;

if (cur.Value == value)

{

indices = cur.ArrayIndices;

return true;

}

else if (value < cur.Value)

{

cur = cur.Left;

}

else if (value > cur.Value)

{

cur = cur.Rigth;

}

}

indices = null;

return false;

}

}

}

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

internal class BSTItem

{

public int Value;

public List<int> ArrayIndices;

public BSTItem Left;

public BSTItem Rigth;

public BSTItem(int value, int arrayIndex)

{

Value = value;

ArrayIndices = new List<int>();

ArrayIndices.Add(arrayIndex);

}

}

}

Таблица 1 – проверка программы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 10 | 100 | 1 000 | 10 000 | 1 000 000 000 |
| Алгоритм 1 | 1,05 | 2,8 | 32,6 | 303,3 | 302946,5 |
| Алгоритм 2 | 2,05 | 5,6 | 5,6 | 5,5 | 4,7 |
| Алгоритм 3 | 1,05 | 2,1 | 3,8 | 3,8 | 3,8 |
| Мин. значение | 1,05 | 2,1 | 3,8 | 3,8 | 3,8 |
| Макс. значение | 2,05 | 5,6 | 32,5 | 303,3 | 302948,5 |
| Ср. значение | 1,55 | 3,85 | 18,15 | 153,55 | 151478,15 |

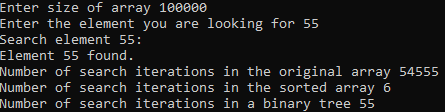


Рисунок 1 – результат выполнения программы