**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

**IКНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 10

**З дисципліни:** *“Алгоритми та структури даних”*

**На тему:** *“Бінарний пошук в упорядкованому масиві”*

**Лектор:**

доц. каф. ПЗ

Коротєєва Т.О.

**Виконала:**

ст. групи ПЗ-26

Матолінець Л. А.

**Прийняв:**

асистент каф. ПЗ

Симець І.І.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема роботи:** Бінарний пошук в упорядкованому масиві.

**Мета роботи**:  Навчитися застосовувати алгоритм бінарного пошуку при розв’язуванні задач та перевірити його ефективність на різних масивах даних. Експериментально визначити складність алгоритму.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

***Покроковий опис алгоритму:***

**Алгоритм BS** Задано одновимірний масив елементів *R1 ,R2 ,…,Rn.* Даний алгоритм дозволяє знайти перше входження шуканого елемента x та повертає його позицію. Тут l – крайній лівий елемент, r – крайній правий елемент, mid = середній елемент між l та r.

**BS1.** Цикл за індексом проходження i = 1…n.

**BS2.** mid = l + (r - l + 1) / 2.

**BS3.** Цикл поки l <= r

**BS4.** Якщо R[mid] < x, то l = mid + 1

**BS5.** Якщо R[mid] > x, то r = mid – 1

**BS6.** Якщо R[mid] = x -> повертнути значення mid

**BS7.** Кінець. Вихід.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

**Варіант 9**

Використовуючи алгоритм бінарного пошуку, знайдіть елемент *b* у масиві *А* з кількістю елементів від 10 до 1000, розташованих за зростанням.

1.               Програма повинна забезпечувати автоматичну генерацію масиву цілих чисел (кількість елементів масиву вказується користувачем) та виведення його на екран;

2.               Визначте кількість порівнянь та порівняйте ефективність на декількох масивах різної розмірності заповнивши табл. 1.

3.               Представте покрокове виконання алгоритму пошуку.

4.               Побудуйте графік залежності кількості порівнянь від кількості елементів масиву у Excel. Побудуйте у тій же системі координат графіки функцій y=n та y=log2(n). Дослідивши графіки, зробіть оцінку кількості С(n) порівнянь алгоритму бінарного пошуку.

5.               З переліку завдань виконайте індивідуальне завдання запропоноване викладачем.

   Дано одновимірний масив цілих чисел A[і], де і =1,2,…,n. Знайти перший елемент що є більшим або рівним значенню к-того числа Фібоначчі.

**ХІД ВИКОНАННЯ**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Diagnostics;

using System.Threading;

class CountingSort

{

static void countSort(int[] arr)

{

int maxValue = arr.Max();

int minValue = arr.Min();

int rangeOfValues = maxValue - minValue + 1;

int []countArray = new int[rangeOfValues];

int []endArray = new int[arr.Length];

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

countArray[i] = 0;

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

countArray[arr[i] - minValue]++;

}

for (int i = 1; i < countArray.Length; i++) {

countArray[i] += countArray[i - 1];

}

for (int i = arr.Length - 1; i >= 0; i--) {

endArray[countArray[arr[i] - minValue] - 1] = arr[i];

countArray[arr[i] - minValue]--;

}

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

arr[i] = endArray[i];

}

}

static int binarySearch(int[] arr, int x)

{

int count = 0;

int l = 0, r = arr.Length - 1;

while (l <= r) {

int m = l + (r - l) / 2;

if (arr[m] == x){

return m;

}

if (arr[m] < x)

l = m + 1;

else

r = m - 1;

count++;

Console.WriteLine("The number of comparisons: {0}\n", count);

}

return -1;

}

public static void Main(string[] args)

{

int Min = -1000;

int Max = 1000;

Console.WriteLine("Enter the size of your array: ");

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

int[] arr = new int[length];

Random randNum = new Random();

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

{

arr[i] = randNum.Next(Min, Max);

}

countSort(arr);

int x = 10;

int n = arr.Length;

int result = binarySearch(arr, x);

if (result == -1)

Console.WriteLine("Element not present");

else

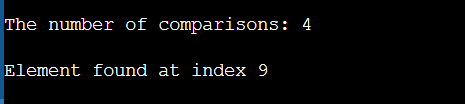
Console.WriteLine("Element found at index "

+ result);

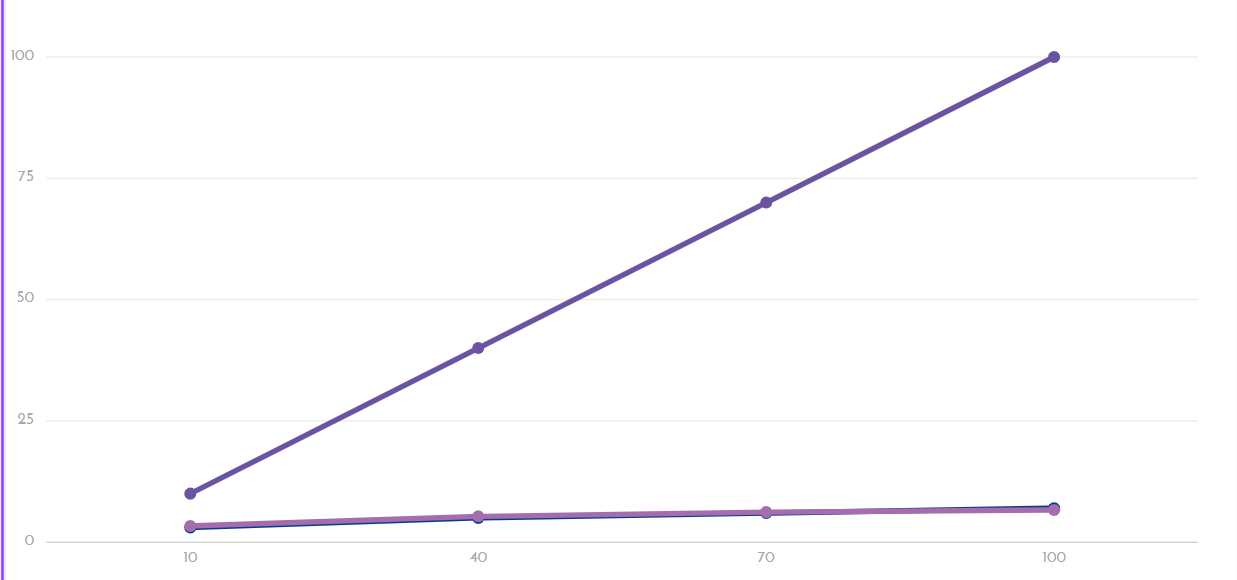
}

}

|  |  |
| --- | --- |
| Кількість елементів | Кількість порівнянь |
| 10 | 3 |
| 40 | 5 |
| 70 | 6 |
| 100 | 7 |



*Рис. 1 Результат виконання програми*



*Рис.2 Графік залежності кількості порівнянь від кількості елементів*

Верхня лінія: y = n, друга: y = log2(n), третя – лінія, побудована на основі даних з таблиці. Вона практично збіглася з y = log2(n).

Вихідний код індивідуального завдання:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

namespace SortingApp

{

public class MyArray

{

protected int[] array = new int[0];

public MyArray()

{

}

public int[] getList() { return array; }

public void addElement(int item)

{

Array.Resize(ref array, array.Length + 1);

array[array.Length - 1] = item;

}

public void setArray(int[] arr)

{

this.array = arr;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Numerics;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

namespace SortingApp

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

///

public partial class MainWindow : Window

{

public static int GetNthFibonacci\_Ite(int n)

{

int number = n - 1;

int[] Fib = new int[number + 1];

Fib[0] = 0;

Fib[1] = 1;

for (int i = 2; i <= number; i++)

{

Fib[i] = Fib[i - 2] + Fib[i - 1];

}

return Fib[number];

}

static void countSort(int[] arr)

{

int maxValue = arr.Max();

int minValue = arr.Min();

int rangeOfValues = maxValue - minValue + 1;

int[] countArray = new int[rangeOfValues];

int[] endArray = new int[arr.Length];

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

countArray[i] = 0;

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

{

countArray[arr[i] - minValue]++;

}

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

{

countArray[i] += countArray[i - 1];

}

for (int i = arr.Length - 1; i >= 0; i--)

{

endArray[countArray[arr[i] - minValue] - 1] = arr[i];

countArray[arr[i] - minValue]--;

}

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

{

arr[i] = endArray[i];

}

}

static int binarySearch(int[] arr, int x)

{

int count = 0;

int l = 0, r = arr.Length - 1;

while (l <= r)

{

int m = l + (r - l) / 2;

if (arr[m] == x)

{

return m;

}

if (arr[m] < x)

l = m + 1;

else

r = m - 1;

count++;

}

return -1;

}

public MainWindow()

{

InitializeComponent();

}

private void addArray\_Click(object sender, RoutedEventArgs e)

{

}

private void addArray\_Click1(object sender, RoutedEventArgs e)

{

}

MyArray array = new MyArray();

int indexValue = 0;

private void addElement\_Click(object sender, RoutedEventArgs e)

{

String arrField = elemField.Text;

String[] strings = arrField.Split(" ; ");

foreach (var item in strings)

{

int x = Convert.ToInt32(item);

array.addElement(x);

}

}

private void addElement1\_Click(object sender, RoutedEventArgs e)

{

}

private void amount\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void showPosition\_Clicked(object sender, TextChangedEventArgs e)

{

}

private void show\_Click(object sender, RoutedEventArgs e)

{

}

private void merge\_Click(object sender, RoutedEventArgs e)

{

}

private void show\_Click\_\_(object sender, RoutedEventArgs e)

{

}

private void add\_Element\_Click\_(object sender, RoutedEventArgs e)

{

//selecting index

indexValue = GetNthFibonacci\_Ite(System.Convert.ToInt32(indexField.Text));

int[] arr = array.getList();

int value = indexValue;

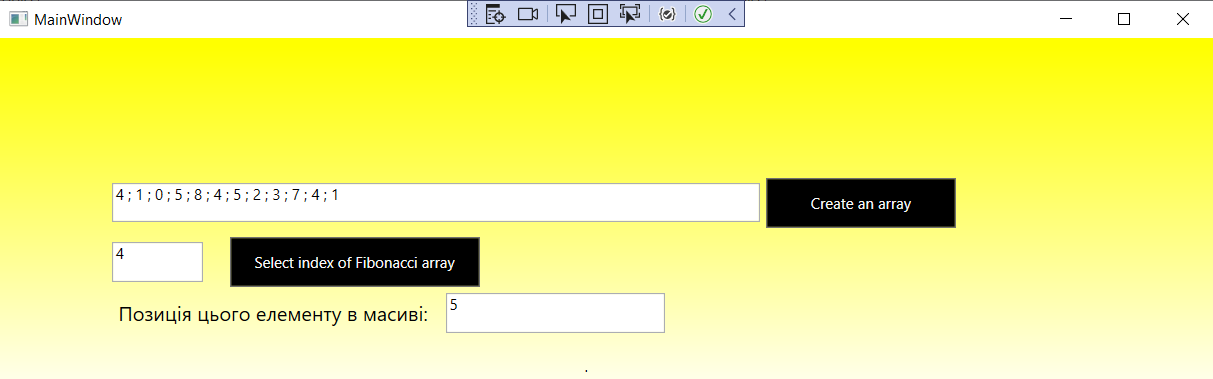
countSort(arr);

resultField1.Text = System.Convert.ToString(binarySearch(arr, value));

}

}

}



*Рис.3 Результат виконання програми*

**Висновок**

На цій лабораторній роботі я познайомилася з алгоритмом бінарного пошуку, імплементувала його у своїй програмі, виконавши індивідуальне завдання. Також я на практиці встановила складність алгоритму, що дорівнює O(log2n). Також я дослідила кількість порівнянь на основі пошуку значення при різній розмірності масиву й на основі цього побудувала порівняльні графіки.