МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут **ІКНІ**

Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи №10

На тему: «Бінарний пошук в упорядкованому масиві»

З дисципліни: «Алгоритми та структури даних»

Лектор : доцент каф.ПЗ

Коротєєва Т.О.

Виконала: ст.гр.ПЗ-23

Кохман О.В.

Прийняв: асистент каф.ПЗ

Франко А.В.

«____»_____2022 р.

∑____ _ _ _ _ _ _ _ _ .

Львів – 2022

Тема: Бінарний пошук в упорядкованому масиві.

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

Теоретичні відомості

Бінарний, або двійковий пошук – алгоритм пошуку елементу у відсортованому масиві. Це класичний алгоритм, ще відомий як метод дихотомії (ділення навпіл).

Якщо елементи масиву впорядковані, задача пошуку суттєво спрощується. Згадайте, наприклад, як Ви шукаєте слово у словнику. Стандартний метод пошуку в упорядкованому масиві – це метод поділу відрізка навпіл, причому відрізком є відрізок індексів *1..n*. Дійсно, нехай масив *A* впорядкований за зростанням і *m (k < m < l)* – деякий індекс. Нехай *Buffer = A[m]*. Тоді якщо *Buffer > b*, далі елемент необхідно шукати на відрізку *k..m-1*, а якщо *Buffer < b* – на відрізку *m+1..l*.

Для того, щоб збалансувати кількість обчислень в тому і іншому випадку, індекс *m* необхідно обирати так, щоб довжина відрізків *k..m, m..l* була (приблизно) рівною. Описану стратегію пошуку називають *бінарним пошуком*.

b – елемент, місце якого необхідно знайти. Крок бінарного пошуку полягає у порівнянні шуканого елемента з середнім елементом *Buffer = A[m]* в діапазоні пошуку *[k..l]*. Алгоритм закінчує роботу при *Buffer = b* (тоді *m* – шуканий індекс). Якщо *Buffer > b*, пошук продовжується ліворуч від *m*, а якщо *Buffer < b* – праворуч від *m*. При *l < k* пошук закінчується, і елемент не знайдено.

Покроковий опис алгоритму бінарного пошуку:

Алгоритм BS:

BS1. Дано A = $\{x_1,\ldots, x_n\}$. Цикл за індексом проходження i = 1..n. X - елемент пошуку. minNum – індекс встановлений на 1-му елементі масиву. maxNum – індекс встановлений на останньому елементі масиву. Mid – індекс який вказує на середину між low і high.
BS2. mid = minNum + maxNum / 2.BS3 Цикл поки minNum <= maxNum
BS4.Якщо A[mid] < X то minNum = mid + 1
BS5. Якщо A[mid] > X то maxNum = mid – 1
BS6. Якщо A[mid] = X вивести mid

В7. Кінець. Вихід.

Індивідуальне завдання

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

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.

Таблиця 1

Кількість елементів	**Кількість порівнянь**
10	
40	
70	
100	

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 C(n) порівнянь алгоритму бінарного пошуку.

5. З переліку завдань виконайте індивідуальне завдання запропоноване викладачем.

Код програми

Назва файлу: MyForm.h

```
#pragma once
#include <random>
#include <algorithm>

namespace lab10 {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for MyForm
```

```
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private:
		int *array;
		int sizeOfArray = 0;
		int count = 0;
	private: System::Windows::Forms::Button^ generateArrayButton;
	protected:
	private: System::Windows::Forms::NumericUpDown^ arraySizeNumericUpDown;
	private: System::Windows::Forms::NumericUpDown^
kElementOfFibNumericUpDown;
	private: System::Windows::Forms::RichTextBox^ arrayRichTextBox;
	private: System::Windows::Forms::RichTextBox^ outputSearchRichTextBox;


	private: System::Windows::Forms::Button^ searchElementButton;
	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::Label^ label2;
	private: System::Windows::Forms::Label^ elementOfFibOutputLabel;


	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			this->generateArrayButton = (gcnew
System::Windows::Forms::Button());
			this->arraySizeNumericUpDown = (gcnew
System::Windows::Forms::NumericUpDown());
			this->kElementOfFibNumericUpDown = (gcnew
System::Windows::Forms::NumericUpDown());
			this->arrayRichTextBox = (gcnew
System::Windows::Forms::RichTextBox());
```

```
                    this->outputSearchRichTextBox = (gcnew
System::Windows::Forms::RichTextBox());
                    this->searchElementButton = (gcnew
System::Windows::Forms::Button());
                    this->label1 = (gcnew System::Windows::Forms::Label());
                    this->label2 = (gcnew System::Windows::Forms::Label());
                    this->elementOfFibOutputLabel = (gcnew
System::Windows::Forms::Label());

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>arraySizeNumericUpDown))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>kElementOfFibNumericUpDown))->BeginInit();
                    this->SuspendLayout();
                    //
                    // generateArrayButton
                    //
                    this->generateArrayButton->Location =
System::Drawing::Point(71, 224);
                    this->generateArrayButton->Name = L"generateArrayButton";
                    this->generateArrayButton->Size = System::Drawing::Size(99,
23);
                    this->generateArrayButton->TabIndex = 0;
                    this->generateArrayButton->Text = L"Generate array";
                    this->generateArrayButton->UseVisualStyleBackColor = true;
                    this->generateArrayButton->Click += gcnew
System::EventHandler(this, &MyForm::generateArrayButton_Click);
                    //
                    // arraySizeNumericUpDown
                    //
                    this->arraySizeNumericUpDown->Location =
System::Drawing::Point(95, 28);
                    this->arraySizeNumericUpDown->Name =
L"arraySizeNumericUpDown";
                    this->arraySizeNumericUpDown->Size =
System::Drawing::Size(120, 20);
                    this->arraySizeNumericUpDown->TabIndex = 1;
                    //
                    // kElementOfFibNumericUpDown
                    //
                    this->kElementOfFibNumericUpDown->Location =
System::Drawing::Point(407, 28);
                    this->kElementOfFibNumericUpDown->Name =
L"kElementOfFibNumericUpDown";
                    this->kElementOfFibNumericUpDown->Size =
System::Drawing::Size(120, 20);
                    this->kElementOfFibNumericUpDown->TabIndex = 2;
                    //
                    // arrayRichTextBox
                    //
                    this->arrayRichTextBox->Location = System::Drawing::Point(27,
54);
                    this->arrayRichTextBox->Name = L"arrayRichTextBox";
                    this->arrayRichTextBox->Size = System::Drawing::Size(196,
164);
                    this->arrayRichTextBox->TabIndex = 3;
                    this->arrayRichTextBox->Text = L"";
                    //
                    // outputSearchRichTextBox
                    //
                    this->outputSearchRichTextBox->Location =
System::Drawing::Point(287, 54);
```

```
                    this->outputSearchRichTextBox->Name =
L"outputSearchRichTextBox";
                    this->outputSearchRichTextBox->Size =
System::Drawing::Size(392, 164);
                    this->outputSearchRichTextBox->TabIndex = 4;
                    this->outputSearchRichTextBox->Text = L"";
                    this->outputSearchRichTextBox->TextChanged += gcnew
System::EventHandler(this, &MyForm::outputSearchRichTextBox_TextChanged);
                    //
                    // searchElementButton
                    //
                    this->searchElementButton->Location =
System::Drawing::Point(419, 224);
                    this->searchElementButton->Name = L"searchElementButton";
                    this->searchElementButton->Size = System::Drawing::Size(102,
23);
                    this->searchElementButton->TabIndex = 5;
                    this->searchElementButton->Text = L"Search element";
                    this->searchElementButton->UseVisualStyleBackColor = true;
                    this->searchElementButton->Click += gcnew
System::EventHandler(this, &MyForm::searchElementButton_Click);
                    //
                    // label1
                    //
                    this->label1->AutoSize = true;
                    this->label1->Location = System::Drawing::Point(24, 30);
                    this->label1->Name = L"label1";
                    this->label1->Size = System::Drawing::Size(65, 13);
                    this->label1->TabIndex = 6;
                    this->label1->Text = L"Size of array";
                    //
                    // label2
                    //
                    this->label2->AutoSize = true;
                    this->label2->Location = System::Drawing::Point(284, 30);
                    this->label2->Name = L"label2";
                    this->label2->Size = System::Drawing::Size(117, 13);
                    this->label2->TabIndex = 7;
                    this->label2->Text = L"k element of Fibonacci ";
                    //
                    // elementOfFibOutputLabel
                    //
                    this->elementOfFibOutputLabel->AutoSize = true;
                    this->elementOfFibOutputLabel->Location =
System::Drawing::Point(546, 30);
                    this->elementOfFibOutputLabel->Name =
L"elementOfFibOutputLabel";
                    this->elementOfFibOutputLabel->Size =
System::Drawing::Size(0, 13);
                    this->elementOfFibOutputLabel->TabIndex = 8;
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(731, 410);
                    this->Controls->Add(this->elementOfFibOutputLabel);
                    this->Controls->Add(this->label2);
                    this->Controls->Add(this->label1);
                    this->Controls->Add(this->searchElementButton);
                    this->Controls->Add(this->outputSearchRichTextBox);
                    this->Controls->Add(this->arrayRichTextBox);
```

```
                    this->Controls->Add(this->kElementOfFibNumericUpDown);
                    this->Controls->Add(this->arraySizeNumericUpDown);
                    this->Controls->Add(this->generateArrayButton);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>arraySizeNumericUpDown))->EndInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>kElementOfFibNumericUpDown))->EndInit();
                    this->ResumeLayout(false);
                    this->PerformLayout();

             }

#pragma endregion
      private: System::Void generateArrayButton_Click(System::Object^ sender,
System::EventArgs^ e) {
             arrayRichTextBox->Text = "";

             std::random_device random_device;
             std::mt19937 generator(random_device());
             std::uniform_int_distribution<> distribution(0, 200);
             sizeOfArray = (int)arraySizeNumericUpDown->Value;

             array = new int[sizeOfArray];

             if (sizeOfArray <= 0)
                    arrayRichTextBox->Text = "";

             for (int i = 0; i < sizeOfArray; i++) {
                    array[i] = distribution(generator);
             }

             std::sort(array, array + sizeOfArray);

             for (int i = 0; i < sizeOfArray; i++) {
                    arrayRichTextBox->Text += array[i].ToString() + " ";
             }
      }

private: System::Void searchElementButton_Click(System::Object^ sender,
System::EventArgs^ e) {
      count = 0;
      outputSearchRichTextBox->Text = "";

      auto kElementOfFib = (int)kElementOfFibNumericUpDown->Value;

      auto fibNumber = fib(kElementOfFib);

      elementOfFibOutputLabel->Text = "Fibonacci element: " +
fibNumber.ToString();

      auto searchedElement = binarySearch(array, 0, sizeOfArray - 1, fibNumber);

      if (searchedElement != -1) {
             outputSearchRichTextBox->Text += "\nSearched element that's equal or
greater than " + fibNumber.ToString() + " is " +
array[searchedElement].ToString();
      }
}
```

```
int binarySearch(int arr[], int l, int r, int x)
{
	if (r >= l) {
		outputSearchRichTextBox->Text += "Step: " + count++ + ". Find middle
index of the following array:\n";

		for (int i = l; i < r + 1; i++) {
			outputSearchRichTextBox->Text += arr[i] + " ";
		}

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

		outputSearchRichTextBox->Text += "\nThe middle index is " + (mid -
l).ToString() + " and middle element is " + arr[mid].ToString() + "\n";


		if (arr[mid] >= x) {
			outputSearchRichTextBox->Text += "Step: " + count++ + ". The
middle element is greater or equal " + x.ToString() + ". Searching is finished"
"\n";
			return mid;
		}

		// It will never be executed because of individual task
		if (arr[mid] > x) {
			outputSearchRichTextBox->Text += "Step: " + count++ + ". The
middle element is greater than element" + x.ToString() + ". Searched element can
only be present in the left subaray. Search element in the following subarray:
\n";

			for (int i = l; i < mid + 1; i++) {
				outputSearchRichTextBox->Text += arr[i] + " ";
			}

			return binarySearch(arr, l, mid - 1, x);
		}

		outputSearchRichTextBox->Text += "\nStep: " + count++ + ". The
middle element is smaller than " + x.ToString() + ". Searched element can only be
present in the right subaray. Search element in the following subarray: \n";

		for (int i = mid + 1; i < r + 1; i++) {
			outputSearchRichTextBox->Text += arr[i] + " ";
		}

		outputSearchRichTextBox->Text += "\n";

		return binarySearch(arr, mid + 1, r, x);
	}

	outputSearchRichTextBox->Text += "Element was not found!";
	return -1;
}
int fib(int n)

{
	if (n <= 1)  //stopping condition

		return n;

	else   //recursive part

		return (fib(n - 1) + fib(n - 2));
```

```
}
private: System::Void outputSearchRichTextBox_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
}
};
}
```

Назва файлу: MyForm.cpp

```
#include "MyForm.h"

using namespace lab10;
int main() {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    Application::Run(gcnew MyForm());
    return 0;
}
```

Протокол роботи

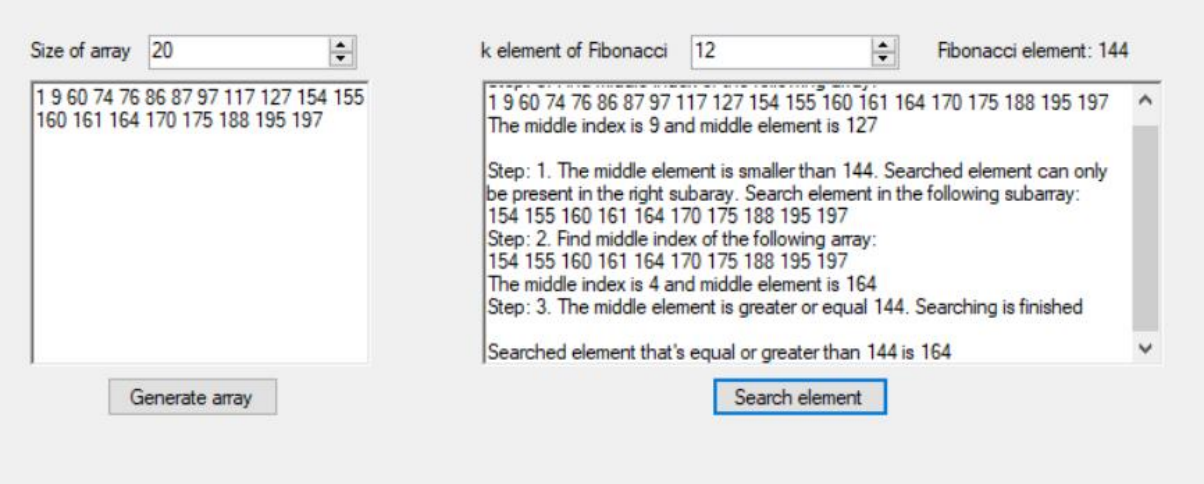

Рис. 1 Результат роботи програми.

Кількість елементів	Кількість порівнянь
10	1
40	1
70	2
100	2

Таблиця 1 Кількість порівнянь при певній кількості елементів.

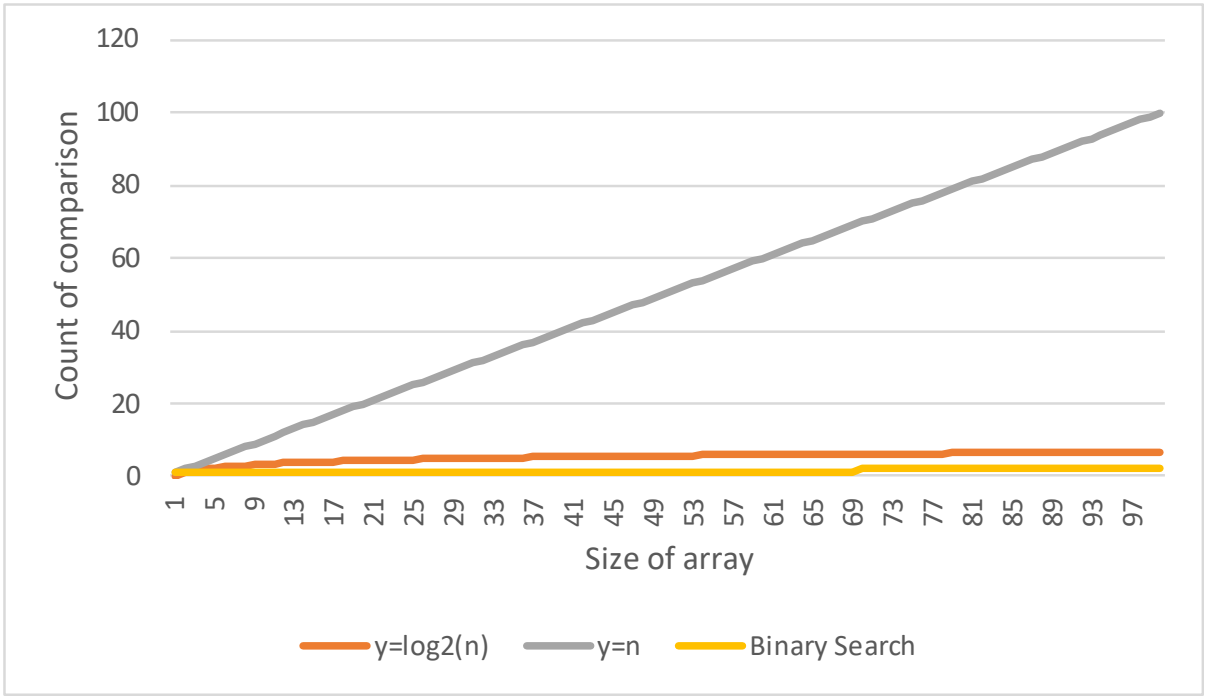

Рис. 2 Графік залежності кількості порівнянь від кількості елементів.

Висновок

На цій лабораторній роботі я дізналась про алгоритм бінарного пошуку в упорядкованому масиві, реалізувала програму за домопогою цього алгоритму та вивела результати на форму у Visual Studio 2022. Також дослідила кількості порівнянь в залежності від кількості елементів та побудувала графік їхньої залежності.