# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №3**

**«Бінарний пошук»**

**з дисципліни «Алгоритми і структури даних»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняла:

доц. Кортєєва Т.О.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

**Тема роботи**: Бінарний пошук.

**Мета роботи**: Вивчити алгоритм бінарного пошуку. Здійснити програмну реалізацію даного алгоритму. Дослідити швидкодію алгоритму.

## 1. Теоретичні відомості

Бінарний (двійковий) пошук можливий у таблицях, організованих як дерева порівнянь. Елементи в таких таблицях зберігаються в лексико­графічному /тобто алфавітному/ порядку, або в порядку зростання число­вих значень ключів.

Загальна стратегія організації дерева полягає у поділі таблиці на підтаблиці, сукупності елементів на підмножини елементів. Тоді процедура пошуку елемента в дереві порівнянь нагадує пошук імені в телефонному довіднику. Для відшукання елемента потрібно спочатку вирішити, в якій підмножині знаходиться елемент, знайти цю підмножину і після цього продовжувати пошук.

Розглянемо спочатку, як можна записати дерево у векторну пам‘ять.

Нехай ключі зберігаються увекторі *КЛЮЧ (i;j) (i, j* - відпо­відно початковий та кінцевий індекси елементів). Елементи у вектор­ній пам‘яті повинні бути впорядкованими за значенням ключів.

При пошуку якогось елемента зробимо коренем дерева вмістиме *КЛЮЧ(т),* де *m*=*[(і+j)/2]-* найбільше ціле, менше або рівне *(i+j)/2.* Тоді ліве піддерево розміщується у векторі *КЛЮЧ (i;m-1),* а пра­ве - у векторі *KЛЮЧ(m+1;j).* Цей процес повторюється, доки не буде знайдений потрібний елемент, тобто доки на зійдемось у одній вершині, для якої обидва індекси у векторі *КЛЮЧ* будуть мати однакові значення.

Тоді алгоритм *D* пошуку ключа *К* у векторі *КЛЮЧ (і;j)* можна записати такою послідовністю кроків ([*КЛЮЧ(m)]* *-* вмістиме елемента вектора з індексом *т ):*

D0. Ініціалізація індексів *і, j* (для табл.2.1 і = 1, j=8).

D1*.* Повторювати кроки D2 - D6 доти, доки *i<j*.

D2. Обчислення індекcа кореня дерева: *m=[(i+j)/2].*

D3. Якщо *[КЛЮЧ (m)]= К то REZ= т,* кінець.

D4. Інакше: якщо [ *КЛЮЧ (т)] <K*.

D5. *Тoдi j=m-1 (пошук* зліва); перехід на D2;

D6. Інакше *і=т +1* (пошук справа); перехід на D2*.*

D7*.* Кінець. (Помилка - запису в таблиці немає.)

Основна задача полягає у виключенні на кожному кроці з подальшого пошуку ак можна більшої кількості елементів. Оптимальним рішенням буде вибір середнього елемента, оскільки при цьому буде виключена половина кількості елементів. Максимальне число порівнянь дорівнює *О*(*log2 n).*

**2. Текст програми на мові програмування JAVA**

package \_asd\_lab3;

import java.util.\*;

/\*\*

\* Л/р № 3

\* Бінарний пошук

\* @author Taras

\*/

public class \_ASD\_Lab3 {

//чи показувати проміжні результати

static final boolean showIntermediateResults = !true;

//масив ключів, який зберігає розташуваня індексів вхідного масиву

//після сортування, відносно ж цього масиву до сортування

static int[] keys;

//вивід покрокової інформації

static void message(Object obj) {

if(showIntermediateResults) System.out.print(obj);

}

//бінарний пошук (масив int[] a повинен бути відсортований)

static int binarySearch(int[] a, int key) {

int low = 0;

int high = a.length - 1;

while (low <= high) {

int mid = (low + high) >>> 1; //порозрядне ділення на 2

int midVal = a[mid];

message("\nm = a[" + mid + "] = " + midVal);

if (midVal < key) {

message("\nm < " + key);

low = mid + 1;

}

else if (midVal > key) {

message("\nm > " + key);

high = mid - 1;

}

else {

message("\nm == " + key);

return mid; // знайдено

}

}

return -1; // не знайдено

}

//функція сортування

static void sort(int arr[]) {

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

// Знайдемо мінімальний елемент на

// проміжку індексів [i; a.length);

// спочатку його індекс рівний i

int minValueIndex = i;

// Перебираємо елементи, що залишилися на проміжку

for (int j = i + 1; j < arr.length; j++) {

// Якщо елемент в позиції j менший

// елемента в позиції minValueIndex, то

// необхідно обновити значення індекса

if (arr[j] < arr[minValueIndex]) {

minValueIndex = j;

}

}

// Міняємо поточний елемент з мінімальним

int temp = arr[i];

arr[i] = arr[minValueIndex];

arr[minValueIndex] = temp;

//записуємо зміни в масив ключів

temp = keys[i];

keys[i] = keys[minValueIndex];

keys[minValueIndex] = temp;

}

}

//перевірка чи алгоритм пошуку повернув правильну позицію

static boolean checkResults(int[] arr, int pos, int value) {

if(pos < 0) {

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

if(arr[i] == value) return false;

return true;

}

else

return arr[pos] == value;

}

//основна функція програми

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Array size: ");

if(!sc.hasNextInt()) {

// Якщо користувач ввів не ціле число виводимо відповідне повідомлення

System.out.println("It's not integer value! Program terminate...");

return;

}

// зчитуємо ціле число і створюємо масив

int[] a = new int[sc.nextInt()];

Calendar c = Calendar.getInstance();

Random rand = new Random(c.getTimeInMillis());

//масив ключів

keys = new int[a.length];

//b - копія масиву а, потрібно для перевірки результату

int[] b = new int[a.length];

//ініціалізація едлементів масиву випадковими числами від 0 до 99

for(int i = 0; i < a.length; i++) {

a[i] = rand.nextInt(100);

//початковий стан ключів

keys[i] = i;

//копіюємо a[i] в b[i]

b[i] = a[i];

}

System.out.println("\nArray:\n" + Arrays.toString(a));

//відсортуємо вхідний масив

sort(a);

message("\nArray after sort:\n" + Arrays.toString(a));

System.out.print("\nValue: ");

if(!sc.hasNextInt()) {

// Якщо користувач ввів не ціле число виводимо відповідне повідомлення

System.out.println("It's not integer value! Program terminate...");

return;

}

// зчитуємо ціле число і виконуємо пошук

int value = sc.nextInt();

//засічемо час початку пошуку

c = Calendar.getInstance();

long timeStart = c.getTimeInMillis();

//функція пошуку

int res = binarySearch(a, value);

//засічемо час завершення пошуку

c = Calendar.getInstance();

long timeStop = c.getTimeInMillis();

int oldPos;

if(res < 0) {

oldPos = -1;

System.out.println("\nValue not found.");

}

else {

oldPos = keys[res];

System.out.println("\nPosition: " + keys[res]);

}

System.out.println("The result is " + checkResults(b, oldPos, value) +

". Time elapsed: " + (timeStop - timeStart) + " msec.");

}

}

**3. Результат виконання програми**

Запустимо програму на виконання без виведення проміжних результатів на екран: для масиву розміром 10 елементів (рис. 3.1) та 10000 елементів (рис. 3.2).

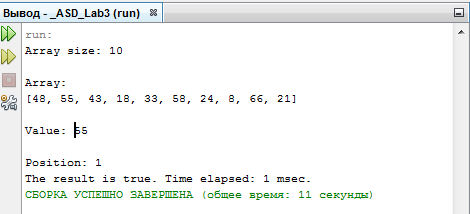


Рис. 3.1. Результат виконання програми

для масиву розміром 10 елементів

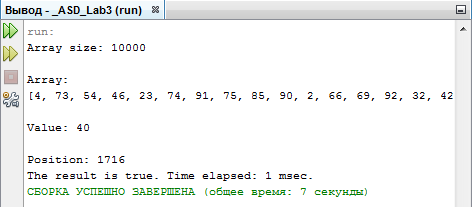


Рис. 3.2. Результат виконання програми

для масиву розміром 10000 елементів

Як бачимо на рисунках, програма виконує пошук в обох випадках правильно, про що свідчить повідомлення про перевірку результату. В першому випадку виконання алгоритму сортування займає час менше одної мілісекунд, а у другому – 1 мілісекунду.

Запустимо програму на виконання в режимі виведення проміжних результатів на екран: для масиву розміром 10 елементів (рис. 3.3) та 10000 елементів (рис. 3.4).

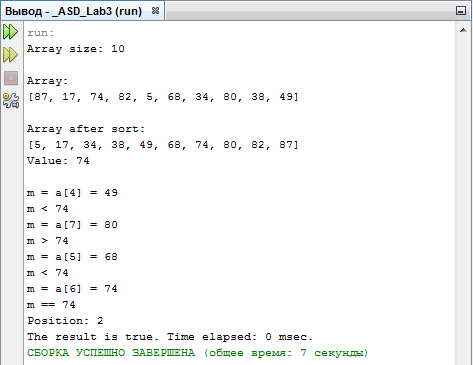


Рис. 3.3. Результат виконання програми

для масиву розміром 10 елементів

(в режимі виводу проміжних даних)

Спробуємо вивести всі проміжні дані для масиву, розміром 1000 елементів (рис. 3.4):

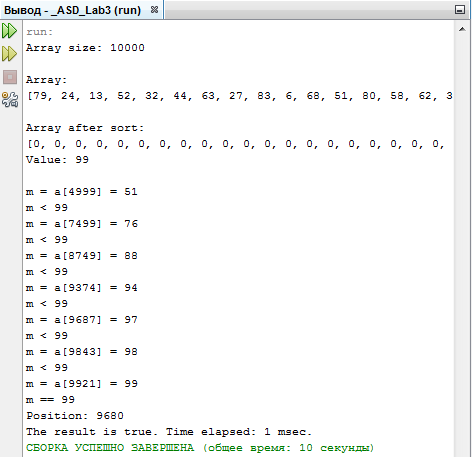


Рис. 3.4. Результат виконання програми

для масиву розміром 10000 елементів

(в режимі виводу проміжних даних)

Можемо зробити висновки, що алгоритм бінарного пошуку є досить швидким, якщо опрацьовуються відсортовані дані.

## ВИСНОВКИ

На даній лабораторній роботі я навчився програмно реалізовувати алгоритм бінарного пошуку, удосконалив свої навички у написанні підпрограм для роботи з масивами, а також оволодів основними функціями роботи з часом та генератором випадкових чисел у JAVA.