МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **ІКНІ**
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.

Виконав:
ст. гр. ПЗ – 22
Ясногородський Н.В.

Прийняв:
асист. каф. ПЗ
Франко А.В.

« ____ » _____ 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

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.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Розробити програму, яка:

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 (n) порівнянь алгоритму бінарного пошуку.
5. З переліку завдань виконайте індивідуальне завдання запропоноване викладачем.

15. Дано одновимірний масив цілих чисел B[i], де i =1,2,...,n. Всі елементи масиву, що менші за його середнє арифметичне значення, збільшити в 2 рази. Знайти позиції елементів, котрі повторюються k раз.

ВИКОНАННЯ РОБОТИ

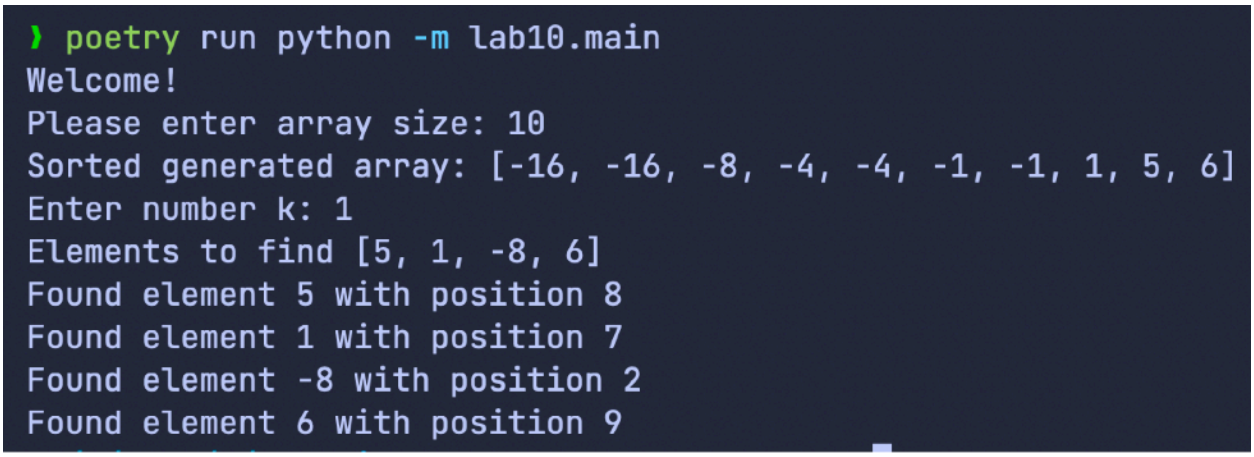
Код програми:

```
# 15.  Дано одновимірний масив цілих чисел B[i], де i =1,2,…,n.
# Всі елементи масиву, що менші за його середнє арифметичне значення,
# збільшити в 2 рази. Знайти позиції елементів, котрі повторюються k раз.

import random
from collections import defaultdict



def gen_random_int_array(n):
    return [int(random.uniform(-n, n)) for _ in range(n)]



def gen_input_data():
    n = int(input("Welcome!\nPlease enter array size: ") or 100)
    arr = gen_random_int_array(n)
    sum = 0
```

```
    for e in arr:
        sum += e
    avg = sum / n

    num_map = defaultdict(int)
    for i in range(n):
        if arr[i] < avg:
            arr[i] *= 2
        num_map[arr[i]] += 1

    arr.sort()
    print(f"Sorted generated array: {arr}")
    k = int(input("Enter number k: ") or 1)
    elements_to_find = [key for key, v in num_map.items() if v == k]

    return arr, elements_to_find


def binary_search(arr, key):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] > key:
            right = mid - 1
        elif arr[mid] < key:
            left = mid + 1
        else:
            return mid


if __name__ == "__main__":
```

```
arr, elements_to_find = gen_input_data()
print(f"Elements to find {elements_to_find}")

for el in elements_to_find:
    idx = binary_search(arr, el)
    print(f"Found element {el} with position {idx}")
```

ПРОТОКОЛ РОБОТИ



ВИСНОВКИ

Під час виконання лабораторної роботи я навчився застосовувати алгоритм бінарного пошуку при розв’язуванні задач та перевірити його ефективність на різних масивах даних. Експериментально визначив складність алгоритму.