Лабораторна робота №1

З АСД

Самойленко Олександр

ІПЗ-11/1

# Умова:

Написати програму мовою C# з можливістю вибору різних алгоритмів пошуку. Продемонструвати роботу (ефективність, час виконання) програм на різних структурах даних (масив, лінійний зв’язаний список), з різними умовами, що забезпечують зменшення часу виконання. Навести аналіз отриманих результатів. Реалізувати алгоритми:

* пошуку перебором елемента масиву, що дорівнює заданому значенню.
* пошуку з бар'єром елемента масиву, що дорівнює заданому значенню.
* бінарного пошуку елемента масиву рівного заданому значенню.
* бінарного пошуку елемента масиву, рівного заданому значенню, в якій нове значення індексу m визначалося б не як середнє значення між L і R, а згідно з правилом золотого перерізу.

# Аналіз задачі:

Заради простоти, всі розробимо алгоритми для роботи з цілими числами. Для кожного методу пошуку створимо окремий статичний клас. Для лінійного пошуку та пошуку з бар’єром також розробимо алгоритми для пошуку у зв’язному списку. Відповідно, знадобиться сам зв’язний список та алгоритм створення списку з масиву.

Спочатку вводитимемо масив, створюватимемо з нього список, і вже після того шукатимемо задані елементи. Для зручної реалізації вибору алгоритму використовуватимемо конструкцію switch.

Для отримання більш наочних результатів, проганятимемо пошукові алгоритми з тими самими даними по кілька разів.

Для того, щоб самостійно не вигадувати великі масиви даних для пошуку, в solution’ні з пошуковими алгоритмами створимо ще один консольний проект, який буде генерувати масив цілих чисел заданої довжини та в заданому проміжку значень, з унікальним або не унікальним ключем.

# Структура основних вхідних та вихідних даних:

Вхідними даними буде масив цілих чисел та елемент, що потрібно знайти.

За умовою, нам потрібно визначити час роботи алгоритмів. Тому, створимо новий клас, що зберігатиме в собі результат пошуку та час, що був витрачений на пошук. Також, цей клас міститиме метод, що виводитиме результати на консоль.

# Алгоритм розв'язання задачі:

Лінійний пошук:

Для масиву:

int i = 0, size = array.Length, position;  
bool isFound = false;  
  
while (i < size && !isFound)  
{  
 if (array[i] == key)  
 {  
 position = i;  
 isFound = true;  
 }  
 i++;  
}

Для списку:

int i = 0, position;  
bool isFound = false;  
var iterator = head;  
  
while (iterator != null && !isFound)  
{  
 if (iterator.Data == key)  
 {  
 position = i;  
 isFound = true;  
 }  
 i++;  
 iterator = iterator.Next;  
}

Пошук з бар’єром:

Для масиву:

int i = 0, size = array.Length;  
  
var newArray = new int[size + 1];  
while (i < size)  
{  
 newArray[i] = array[i];  
 i++;  
}  
newArray[i] = key;  
i = 0;  
  
while (!newArray[i].Equals(key))  
{  
 i++;  
}

Для списку:

int i = 0;  
var iterator = head;  
  
var newLast = new LinkedList(key, null);  
while (iterator.Next != null)  
{  
 iterator = iterator.Next;  
}  
iterator.Next = newLast;  
iterator = head;  
  
while (iterator.Data != key)  
{  
 i++;  
 iterator = iterator.Next;  
}

Бінарний пошук:

Для масиву:

int min = 0, max = array.Length - 1, mid;  
double lambda = (Math.Sqrt(5) + 1) / 2;  
double ratio = isGoldenRatio ? 1 + lambda : 2;  
double multiplyer = isGoldenRatio ? lambda : 1;

while (min <= max)   
{   
 mid = (int)((min + multiplyer \* max) / ratio);  
 if (key < array[mid])   
 {   
 max = mid - 1;   
 }   
 else   
 {   
 min = mid + 1;   
 }   
}

Для списку:

private static LinkedList GetElement(LinkedList start, int position)  
{  
 for (int i = 0; i < position; i++)  
 {  
 start = start.Next;  
 }  
 return start;  
}

private static int GetLength(LinkedList head)  
{  
 int i = 0;  
 while (head.Next != null)  
 {  
 i++;  
 head = head.Next;  
 }  
  
 return i;  
}

int minIndex = 0, maxIndex = GetLength(head), midIndex = 0;  
LinkedList minNode = head, maxNode = null, midNode = null;  
  
double lambda = (Math.Sqrt(5) + 1) / 2;  
double ratio = isGoldenRatio ? 1 + lambda : 2;  
double multiplayer = isGoldenRatio ? lambda : 1;  
  
while (maxNode == null || minNode.Data != maxNode.Data)  
{  
 midIndex = (int) ((minIndex + multiplayer \* maxIndex) / ratio);  
 midNode = GetElement(minNode, midIndex - minIndex);  
 if (key <= midNode.Data)  
 {  
 maxNode = midNode;  
 maxIndex = midIndex;  
 }  
 else  
 {  
 minNode = midNode.Next;  
 minIndex = midIndex + 1;  
 }  
}

# Код програми:

Увесь код програми та генератора тестів знаходиться в репозиторії GitHub:

<https://github.com/sefirus/Univ1.2/tree/master/ASD/Lab1_ASD>

# Набір тестів:

1. Ключ: 18, знаходиться на 37 позиції:

Масив, 40 елементів: -13 -20 -15 -12 -11 -9 -4 9 -9 8 15 -2 -16 -8 -13 17 -20 -1 -7 14 1 12 -3 1 6 3 -11 17 -4 -8 17 19 -16 -13 2 0 -12 18 -13 -10

1. Ключ: -33, знаходиться на 7 позиції

Масив, 100 елементів: -27 -11 0 -36 -34 10 -37 -33 -22 39 35 -17 -3 2 26 -20 18 -22 -22 -24 8 15 -10 37 -16 3 22 -35 11 0 1 8 20 28 28 37 -15 39 -9 2 -11 -38 24 9 -24 -31 -26 -9 -25 -27 -37 1 -4 -7 -35 21 -9 11 22 38 7 11 7 -29 -7 -15 -39 -9 -30 -18 29 -12 14 33 -28 19 17 -17 -32 27 19 -39 9 16 0 -31 -37 -29 -4 27 24 -28 34 5 19 -3 32 31 -29 -2

1. Ключ: -15, відсутній

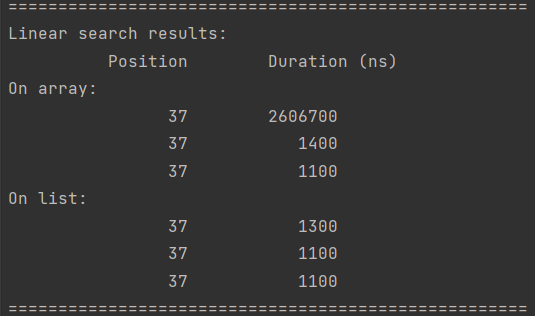
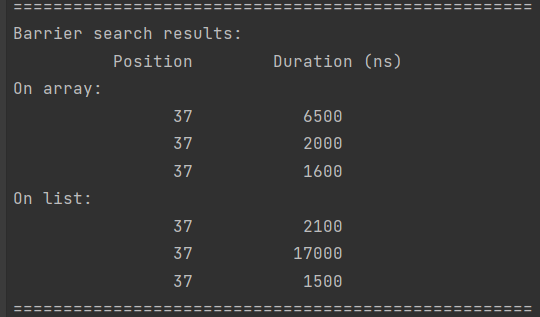
Масив, 100 елементів: -7 -13 -36 -30 22 5 31 15 22 9 -20 -4 -27 -22 39 1 -29 -38 -34 37 10 -12 19 13 -30 -12 35 11 38 -22 -37 24 -40 -14 -10 -20 -40 28 -32 -26 -17 -16 2 5 -18 12 3 -24 -36 -1 0 -28 9 -39 22 16 33 -8 -16 -24 24 -14 4 -7 -16 16 -40 24 28 -23 15 6 -37 -7 -31 20 -14 31 31 14 -3 6 -21 34 17 -12 -17 -36 -36 -19 25 -36 -14 -4 15 30 13 9 -12 -24

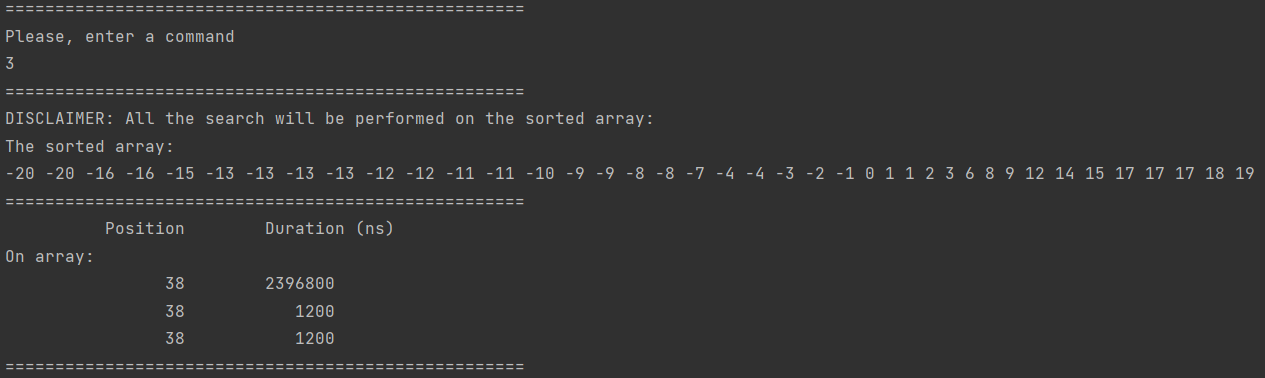
1. Ключ: 5, знаходиться на 2, 91, 123, 276, 369, 504, 509 позиціях

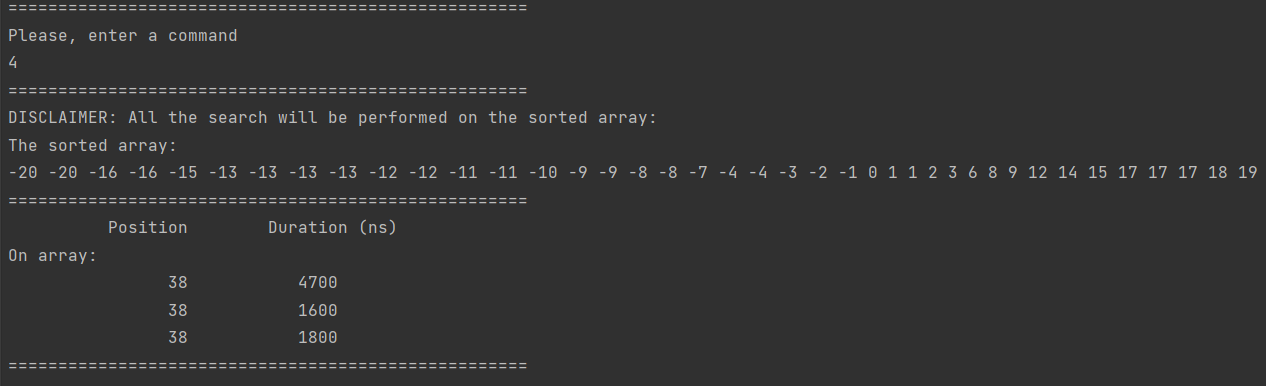
Масив, 1000 елементів: -15 -43 5 72 -22 -66 70 97 -22 79 71 60 -54 -87 -75 -31 -31 68 -81 12 33 -88 88 -68 -51 -71 35 -40 85 -75 34 -27 56 85 -93 87 -58 99 -99 -80 -68 -65 -29 76 -5 -52 -34 46 -21 -21 28 -49 -68 96 -87 -53 70 -79 38 77 -55 -37 86 48 -88 -47 -69 -74 79 30 -90 11 -2 -40 33 -18 -63 33 -1 -79 93 -96 -65 44 -6 86 31 -96 -25 58 48 5 2 -38 84 95 51 -50 91 72 83 -3 -17 -91 -77 15 -11 -31 -63 -64 17 10 -15 -85 23 60 26 11 -28 -81 -41 41 17 5 -97 -78 74 66 -70 -89 -22 -4 -44 72 -3 -63 12 8 11 -57 92 88 44 -17 -58 -96 -5 -26 -91 -35 47 -32 39 -87 52 62 -92 -41 85 63 -80 -58 90 -58 -45 94 -29 -75 -63 -84 -90 0 95 91 -67 42 -70 -50 -98 8 -55 25 61 -81 -93 36 8 45 96 -38 -94 -80 75 52 -11 29 -74 84 94 -11 9 -63 14 90 -15 99 -44 -95 -90 81 89 -23 -66 -74 16 48 -92 -80 94 13 -22 36 -86 9 33 -37 60 10 -27 -36 91 30 -44 16 -36 -16 -22 14 52 -49 -20 -76 -90 -81 93 12 68 6 -51 13 -86 -13 73 -41 85 -34 -28 3 -39 -53 -20 -17 87 -4 63 76 60 60 10 32 -43 88 9 15 -51 93 5 38 68 -39 31 -22 -80 -2 -2 -49 68 53 6 30 -32 -74 46 38 -42 -11 -94 46 97 -60 -19 -4 8 56 -100 58 34 40 63 19 -25 -45 -61 43 -52 -94 -92 6 85 -100 -88 -83 21 74 31 -74 31 -43 22 -52 -91 -44 -64 30 -96 -18 -24 66 -26 -58 15 -28 90 -30 -13 3 28 -80 -3 69 25 82 -93 96 -95 -76 52 59 -66 -92 77 -45 1 15 -7 77 -5 68 75 5 79 19 49 -69 -28 -12 -99 31 -11 -57 -32 81 59 -61 -19 -87 15 12 59 -6 -64 -36 62 99 64 0 98 57 -9 -45 -54 12 -83 62 -26 -42 78 41 -86 65 -23 43 -19 86 18 46 -18 -8 17 -77 -42 -54 -97 -43 -37 68 74 -45 45 -23 -11 69 91 53 -71 -43 89 72 82 -90 40 71 -69 -71 28 -19 -12 -18 -19 -10 -64 64 86 -31 44 0 -45 64 -97 63 -1 -21 23 -39 -8 -63 -2 -22 -78 -47 50 -5 68 -80 -5 -9 -48 -40 -49 -34 -40 84 -66 89 61 59 89 -9 -33 -53 -80 63 -68 -75 -3 -30 -18 58 -77 14 -99 23 -19 -77 5 91 10 77 49 5 78 96 -28 71 73 -56 -54 -84 -58 -48 21 -15 -2 -62 -16 -46 -47 -79 -71 -60 -27 45 -92 84 37 47 67 -17 -91 20 -78 -84 -33 -56 -56 35 25 -54 -34 99 42 24 64 43 85 -35 54 92 70 -59 93 54 -31 87 44 67 -66 -70 -77 -50 10 52 36 78 99 6 8 -79 33 -6 -53 -15 -73 -85 -8 -45 0 35 -48 39 -59 72 -27 74 -68 -60 14 90 -46 41 -66 -92 -77 -91 -50 -93 86 49 2 45 27 2 -8 -69 53 36 20 -18 83 -38 4 57 64 -12 -58 64 2 -4 -67 69 -69 -7 71 -4 10 -43 -20 -87 53 59 -73 -49 43 14 31 77 72 -76 69 -58 -8 56 -39 25 -66 -90 47 -73 43 38 -40 -76 73 -73 65 -3 82 11 25 -49 -39 -76 -63 42 20 -45 -60 -87 -51 -56 78 91 -27 59 51 -97 -26 2 2 -97 65 4 -26 -66 -78 81 30 -3 -25 -93 -55 -81 15 -77 12 -14 -37 42 22 -90 97 -100 -50 -39 -15 -8 70 -71 88 -10 20 21 -18 -67 71 -94 -69 -20 -65 96 -44 -24 80 78 -87 63 12 -53 -70 48 90 65 42 58 -49 -57 52 -28 -97 -56 -64 46 -49 -76 91 -97 56 16 41 71 -63 -27 1 47 44 97 46 69 87 45 -93 2 -10 -64 36 -25 -17 -9 93 -71 -5 47 85 -29 -22 -33 -49 -98 -89 -64 -93 24 14 -14 -38 95 92 -39 56 58 -73 34 -79 -38 -14 87 -94 -22 23 -92 -16 -10 70 -51 58 -22 -42 -6 38 55 26 79 82 -13 81 73 89 22 -43 -72 82 40 -83 -86 -65 44 -58 73 78 67 -82 -21 -75 -36 50 84 -47 -67 -30 -75 3 8 17 -31 43 -21 20 7 -66 -70 -13 -58 11 28 -64 0 -86 57 -81 -32 -23 99 -21 -63 -23 30 -35 79 -90 -35 29 -63 85 93 21 13 -82 40 -83 -70 94 -9 -50 -29 15 -21 93 92 67 -49 41 31 81 -94 31 -89 36 -53 -93 -11 32 -95 -58 -2 -89 81 97 -84 44 -36 -88 71 -26 -65 -17 -3 -74 63 41 -12 -78 82 -41 -67 -14 -45 -31 -25 62 93 -21 -4 27 -41 42 -27 47 -46 -29 34 -99 -53 46 13 6 -39 -11 87 -65 -25 -27 -80 -72 37 9 -93 43 -16 81 -1 -36 97 -75 -22 -79 2 7 58 81 -64 60 93 -23 89 21 -52 -15 35 -63 -48 56 -13 -89 -57 -29 -89 18 94 31 65

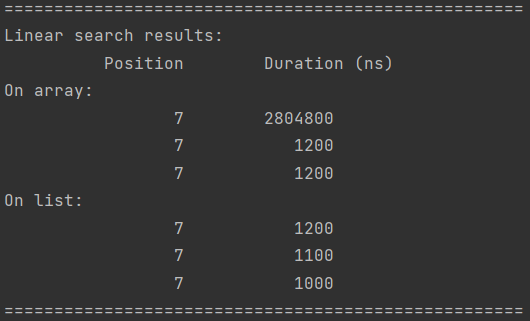
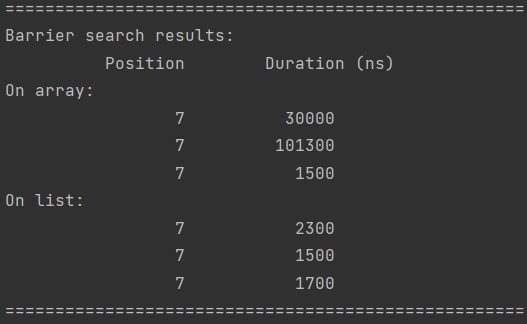
# Результати тестування:

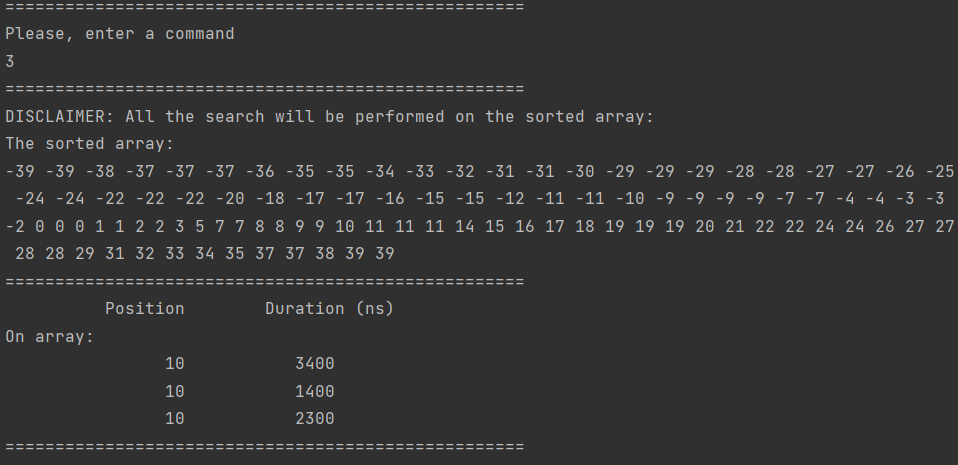
Тест 1:

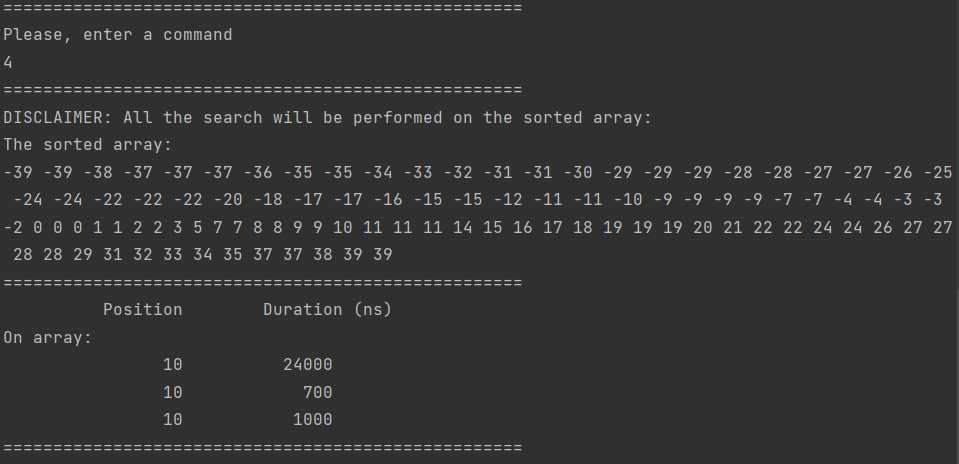


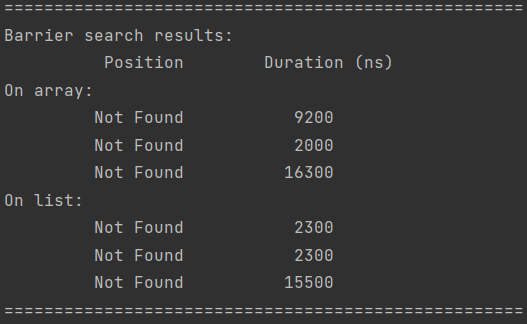
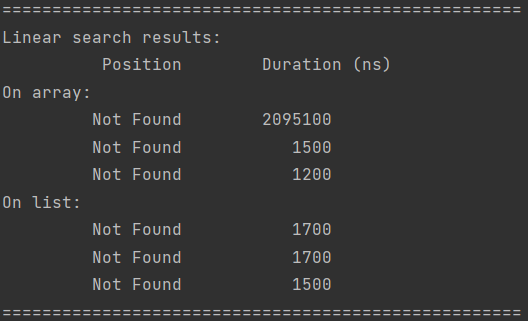


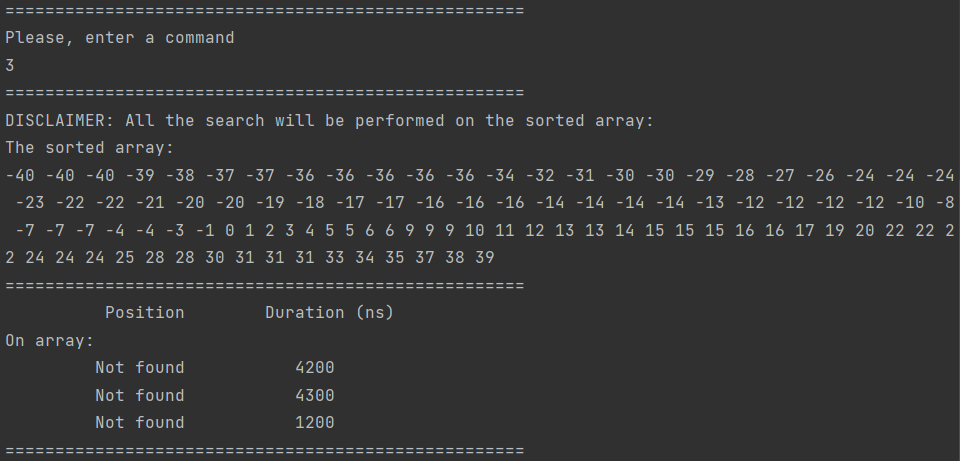


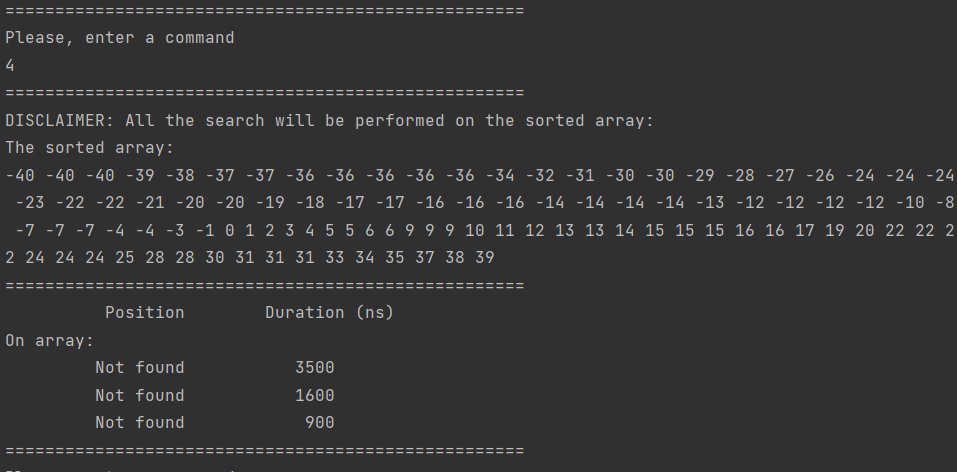
Тест 2:

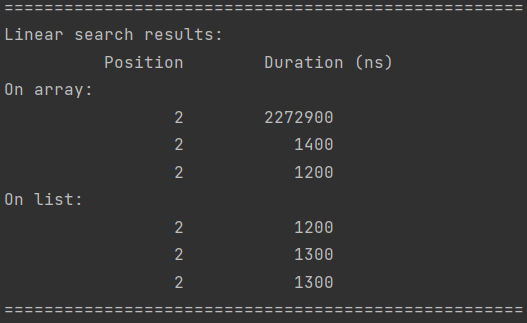
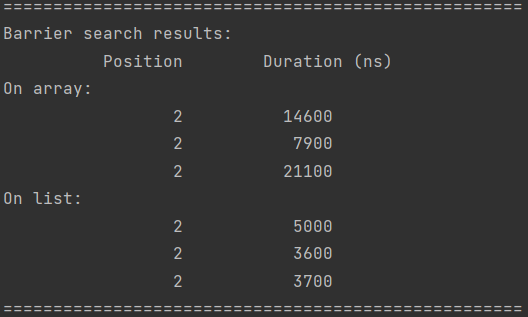


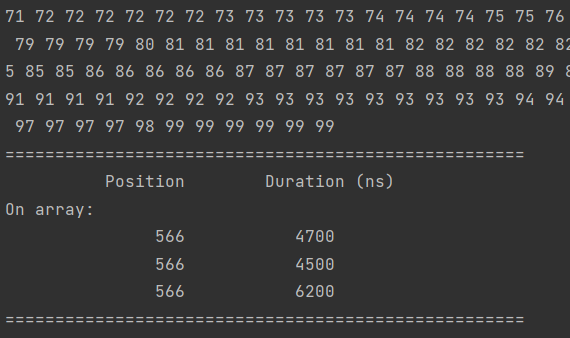


Тест 3:

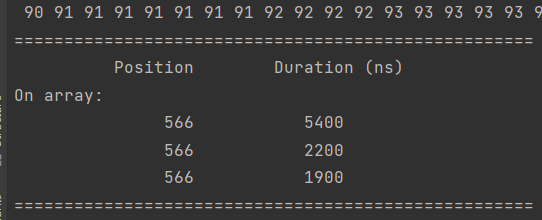




Тест 4:

Звичайний бінарний: 

Бінарний з золотим перерізом:



Як бачимо, помилок в остаточній версії немає.

В процесі розробки були численні помилки з бінарним пошуком з золотим перерізом, через неправильну реалізацію формули ділення відрізка у заданому відношенні. Основна проблема була в тому, що у формулі

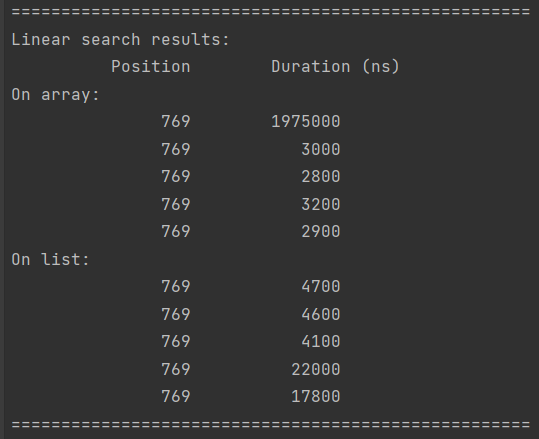
, де – відношення ділення

був забутий множник у знаменнику, та часто ставалась помилка виходу за межі масиву

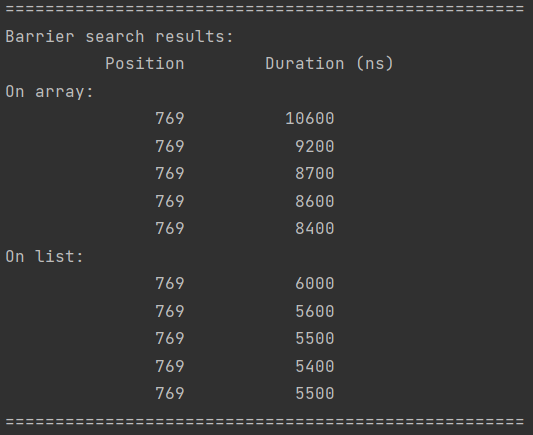
# Тестування:

Проведемо тестування на одному й тому самому випадковому масиві розміром у 1000 елементів:

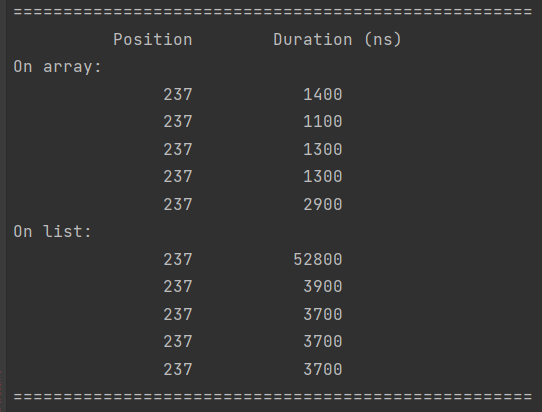
Лінійний пошук:



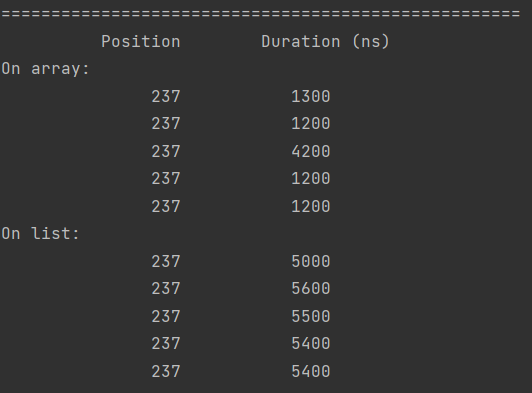
Пошук з бар’єром:



Звичайний бінарний пошук:

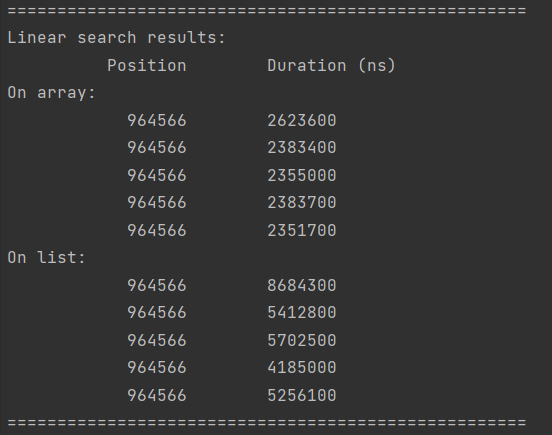


Бінарний пошук з правилом золотого перерізу:

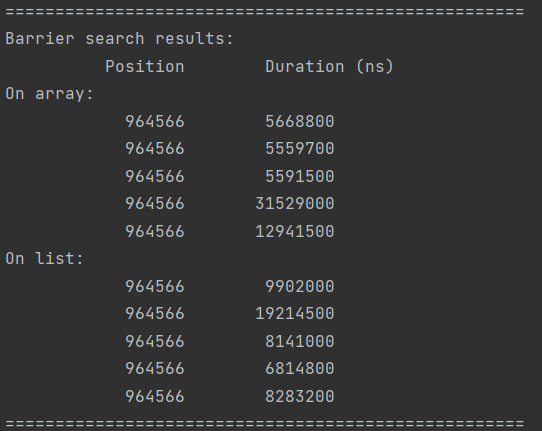


Проведемо аналогічне тестування на випадковому масиві розміром у 1000000 елементів:

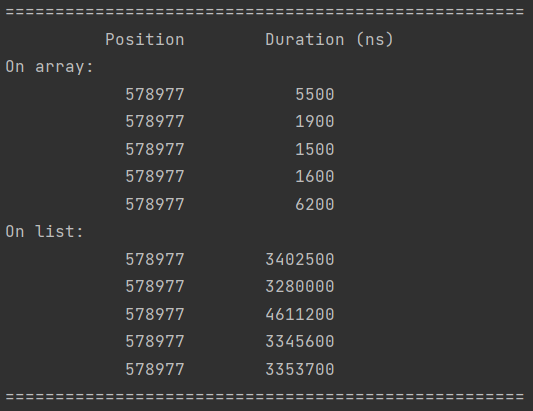
Лінійний пошук:



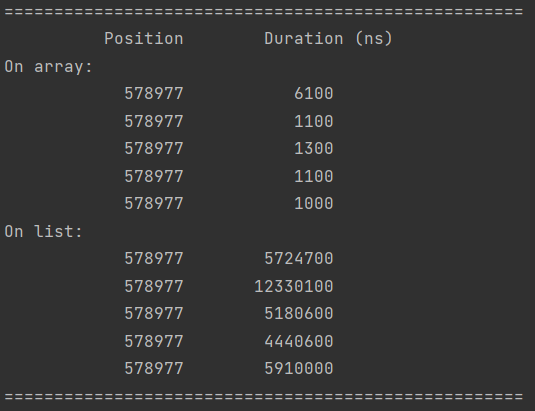
Пошук з бар’єром:



Звичайний бінарний пошук:



Бінарний пошук з правилом золотого перерізу:



# Висновок:

Виконано завдання першої лабораторної роботи. Реалізовано різні алгоритми пошуку на різних структурах даних. Найефективнішим алгоритмом для масиву виявився бінарний пошук з правилом золотого перерізу, а для зв’язного списку - лінійний