**ALGORITMA PENCARIAN (SEARCHING ALGORITHM)**

Algoritma pencarian dirancang untuk menguji keberadaan sebuah elemen atau menerima elemen data dari sebarang struktur data dimana data tersimpan. Berdasarkan jenis operasi pencarian, dikelompokkan dalam 2 kategori :

1. Sequential Search.

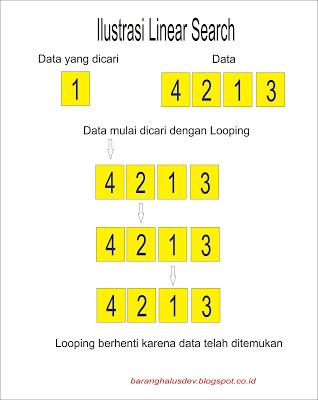
Dalam hal ini, daftar(list) atau array dilalui secara berurutan dan setiap elemen diperiksa. Sebagai contoh: linear search

1. Interval Search

Algoritma ini secara khusus dirancang untuk pencarian dalam struktur data yang diurutkan. Jenis algoritma pencarian ini jauh lebih efisien daripada Linear Search karena mereka berulang kali menargetkan pusat struktur pencarian dan membagi ruang pencarian menjadi dua. Sebagai contoh: Binary Search

**Linear Search**

Dalam jenis pencarian ini, kita hanya menelusuri daftar secara lengkap dan mencocokkan setiap elemen dari daftar dengan item yang lokasinya akan ditemukan. Jika kecocokan ditemukan maka lokasi item dikembalikan jika algoritma mengembalikan NULL.



Masalah: Diberikan array arr [] dengan n elemen, tulis fungsi untuk mencari elemen yang diberikan x dalam arr [].

Input :

arr[] = {10, 20, 80, 30, 60, 50, 110, 100, 130, 170}

x = 110;

Output : 6

Element x is present at index 6

Input : arr[] = {10, 20, 80, 30, 60, 50, 110, 100, 130, 170}

x = 175;

Output : -1

Element x is not present in arr[].

Cara Kerja :

* Dimulai dari elemen ujung kiri dalam arr[] dan setiap elemen satu per satu dibandingkan dengan x
* Jika x sesuai dengan elemen tersebut, mengembalikan indeks
* Jika x tidak sesuai dengan elemen manapun, mengembalikan nilai -1

**Algorithm:**

* Langkah 1: Melintasi array
* Langkah 2: Mencocokkan elemen kunci terhadap elemen array
* Langkah 3: Jika elemen kunci ditemukan, kembali ke posisi indeks dari elemen array.
* Langkah 4: I Jika elemen kunci tidak ditemukan , kembali ke -1

Contoh coding :

* Jika elemen array sudah ditentukan dalam coding

|  |
| --- |
| Public class LinearSearch {  public static int search(int arr[], int x) {       int n = arr.length;       for(int i = 0; i < n; i++) {           if(arr[i] == x)              return i;       }       return -1;  }  public static void main(String args[]) {       int arr[] = { 2, 3, 4, 10, 40 };       int x = 10;         int result = search(arr, x);       if(result == -1)           System.out.print("Element is not present in array");       else           System.out.print("Element is present at index " + result);  }  } |

**Output:**

Element is present at index 3

* Jika panjang array ( banyaknya elemen) dan elemen array di masukkan sebagai input ( belum ditentukan dalam coding)

import java.util.Scanner;

class LinearSearchExample

{

public static void main(String args[])

{

int counter, num, item, array[];

//Input banyaknya elemen

Scanner input = new Scanner(System.in);

System.out.println("Enter number of elements:");

num = input.nextInt();

//Input bilangan2 pada array

array = new int[num];

System.out.println("Enter " + num + " integers");

//Proses pencarian

for (counter = 0; counter < num; counter++)

array[counter] = input.nextInt();

System.out.println("Enter the search value:");

item = input.nextInt();

for (counter = 0; counter < num; counter++)

{

if (array[counter] == item)

{

System.out.println(item+" is present at location "+(counter+1));

/\*Item is found so to stop the search and to come out of the

\* loop use break statement.\*/

break;

}

}

if (counter == num)

System.out.println(item + " doesn't exist in array.");

}

}

**Output 1:**

Enter number of elements:

6

Enter 6 integers

22

33

45

1

3

99

Enter the search value:

45

45 is present at location 3

**Output 2:**

Enter number of elements:

4

Enter 4 integers

11

22

4

5

Enter the search value:

99

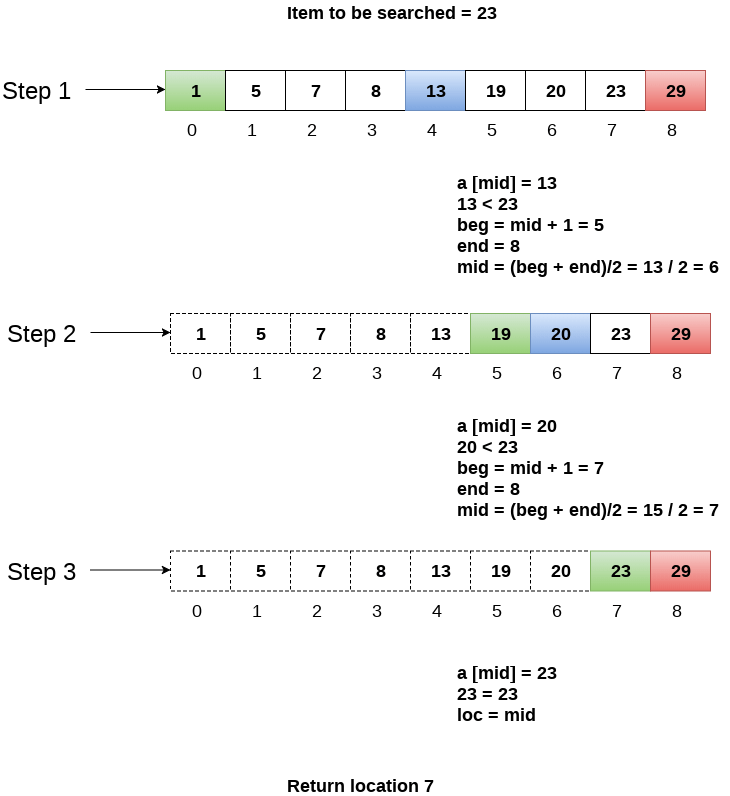
99 doesn't exist in array.

**Binary Search**

Cari array yang diurutkan dengan berulang kali membagi interval pencarian menjadi dua. Mulailah dengan interval yang mencakup seluruh array. Jika nilai kunci pencarian kurang dari item di tengah interval, persempit interval ke bagian bawah. Kalau tidak, persempit ke bagian atas. Periksa berulang kali hingga nilainya ditemukan atau intervalnya kosong.

Cara kerja :

* Bandingkan x dengan elemen tengah
* Jika x cocok dengan elemen tengah, akan mengembalikan indeks tengah
* Jika x > elemen tengah, maka x hanya bisa terletak di setengah sub array kanan setelah elemen tengah. Berulang lagi untuk setengah sub array kanan
* Juga x < elemen tengah, maka x hanya bisa terletak di setengah sub array kiri setelah elemen tengah. Berulang lagi untuk setengah sub array kiri





Contoh coding :

* Jika elemen array sudah ditentukan dalam coding

class BinarySearch {

    int binarySearch(int arr[], int l, int r, int x)

    {

        if (r >= l) {

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

              if (arr[mid] == x)

                return mid;

    if (arr[mid] > x)

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

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

        }

        return -1;

    }

    public static void main(String args[])

    {

        BinarySearch ob = new BinarySearch();

        int arr[] = { 2, 3, 4, 10, 40 };

        int n = arr.length;

        int x = 10;

        int result = ob.binarySearch(arr, 0, n - 1, x);

        if (result == -1)

            System.out.println("Element not present");

        else

            System.out.println("Element found at index " + result);

    }

}

OUTPUT

Element is present at index 3

* Jika panjang array ( banyaknya elemen) dan elemen array di masukkan sebagai input ( belum ditentukan dalam coding)

import java.util.Scanner;

class BinarySearchExample

{

public static void main(String args[])

{

int counter, num, item, array[], first, last, middle;

//To capture user input

Scanner input = new Scanner(System.in);

System.out.println("Enter number of elements:");

num = input.nextInt();

//Creating array to store the all the numbers

array = new int[num];

System.out.println("Enter " + num + " integers");

//Loop to store each numbers in array

for (counter = 0; counter < num; counter++)

array[counter] = input.nextInt();

System.out.println("Enter the search value:");

item = input.nextInt();

first = 0;

last = num - 1;

middle = (first + last)/2;

while( first <= last )

{

if ( array[middle] < item )

first = middle + 1;

else if ( array[middle] == item )

{

System.out.println(item + " found at location " + (middle + 1) + ".");

break;

}

else

{

last = middle - 1;

}

middle = (first + last)/2;

}

if ( first > last )

System.out.println(item + " is not found.\n");

}

}

**Output 1:**

Enter number of elements:

7

Enter 7 integers

4

5

66

77

8

99

0

Enter the search value:

77

77 found at location 4.

**Output 2:**

Enter number of elements:

5

Enter 5 integers

12

3

77

890

23

Enter the search value:

99

99 is not found.

**TUGAS :**

Buatlah coding untuk menampilkan median dari dua array berupa barisan bilangan yang tidak terurut

Contoh :

Array1[] = {13, 3, 15, 9, 10}

Array1[] = {20, 4, 16, 6, 12}

Langkah 1 : urutkan masing2 array

Array1[] = {3, 9, 10, 13, 15}

Array1[] = {4, 6, 12, 16, 20}

Langkah 2 : gabunglah 2 array

{3,4,6,9, 10,12,13,15,16,20}

Langkah 3 : dapatkan median nya

Median =(10+12)/2 = 11