LAPORAN

Struktur Data Linear
Praktikum 3 : Searching

NAMA : Johanes Yogtan WR

NIM : 215314105

Program Studi INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA

A. Sequential Search

- Input (screenshot)

```
package Modul_Searching;
import java.util.Scanner;
public class Tugas2Sequential {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int array[] = {5, 8, 26, 15, 11, 31};
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i]+" ");
        }
        System.out.print("\nMasukkan Data Pencarian : ");
        int cari = sc.nextInt();
        searchSequential(array, cari);
    }
    static int searchSequential(int[] array, int cari) {
        for (int i = 0; i < array.length; i++) {
            if (array[i] == cari) {
                System.out.println("Data Ditemukan di Index "+i);
                return i;
            }
        }
        System.out.println("Data tidak ditemukan");
        return -1;
    }
}
```

- Output (screenshot)

```
run:
5 8 26 15 11 31
Masukkan Data Pencarian : 26
Data Ditemukan di Index 2
```

- Ilustrasi

Int[] array, mencari angka 26

5	8	26	15	11	31
0	1	2	3	4	5

Langkah 1 :

5	8	26	15	11	31

Langkah 2 :

	8	26	15	11	31

Langkah 3 :

		26	15	11	31

- Penjelasan

  Sequential search melakukan pencarian dengan membandingkan data yang dicari satu persatu dari awal hingga sampai hingga data ketemu atau tidak ketemu. Seperti program diatas ini, pertama-tama kita membuat data arraynya terlebih dahulu (Disini saya membuat data acak dengan jumlah 6) , lalu ketik data yang ingin kita cari (Disini saya mencari data 26). Setelah itu kita membuat metode dengan isian perulangan yang dimana berfungsi untuk menjalankan data array, selama perulangan tersebut berjalan kita melakukan pengecakan apakah data dalam array tersebut sama dengan data yang kita cari, dalam perulangan pengecekan ini dilakukan secara satu persatu hingga data yang kita cari ketemu. Ternyata setelah dicek data yang kita cari sama dengan data di indek 2 array, maka data tersebut akan di return dan dicetak;

B. Binary Search

- Input (screenshot)

```
package Modul_Searching;
import java.util.Arrays;
import java.util.Scanner;
public class Tugas2bBinary {
    public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);

        int array[] = {5, 8, 26, 15, 11, 31};
        Arrays.sort(array);
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i]+" ");
        }

        System.out.print("\nMasukkan Data Pencarian : ");
        int cari = sc.nextInt();

        binarySearch(array, 0, array.length, cari);
    }
    public static int binarySearch(int array[], int indekAwal, int indekAkhir, int cari) {
        int indekTengah = 0;
        while (indekAwal <= indekAkhir) {
            indekTengah = (indekAwal + indekAkhir) / 2;
            if (array[indekTengah] == cari) {
                System.out.println("Data Ditemukan di Indek " + indekTengah);
                return indekTengah;
            } else if (array[indekTengah] < cari) {
                indekAwal = indekTengah + 1;
            } else {
                indekAkhir = indekTengah - 1;
        }
            }
        System.out.println("Data Tidak Ditemukan");
        return -1;
    }
}
```

- Output (screenshot)

```
run:
5 8 11 15 26 31
Masukkan Data Pencarian : 11
Data Ditemukan di Indek 2
```

- Ilustrasi

  Int[] array, mencari angka 11

5	8	26	15	11	31
0	1	2	3	4	5

  Langkah 1 :

5	8	**11**	15	26	31

  Langkah 2 :

5	8	**11**	15	26	31

  Langkah 3 :

5	8	**11**	15	26	31

  Langkah 4 :

5	8	11	15	26	31

- Penjelasan

  Pencarian secara biner mencari simbol dari tengah daftar sampai data terakhir, dan membandingkannya dengan simbol yang sedang dicari dalam data yang telah diurutkan. Seperti program diatas ini, pertama-tama kita membuat data arraynya terlebih dahulu (Disini saya membuat data acak dengan jumlah 6) , urutkan datanya menggunakan model sort java dan tampilkan data yang telah diurutkan, lalu ketik data yang ingin kita cari (Disini saya mencari data 11). Setelah itu kita membuat metode dengan isian perulangan yang dimana berfungsi untuk menjalankan data array(indek awal kurang dari indek akhir), selama perulangan tersebut berjalan kita melakukan pengecakan apakah data dalam array tersebut sama dengan data yang kira cari, dengan cara membagi nilai awal dan akhirnya yang dimana kita mendapatkan nilai tengahnya setelah itu dicek apakah indek nilai tengah sama dengan data yang dicari, jiak tidak kita cek lagi apakah nilaitengah tersebut datanya lebih kecil dari yang kita cari, apabila nilai tengah tersebut datanya lebih kecil maka kita menjumlahkan nilai tengah dengan satu yang dimasukkan ke indek awal dalam perulangan nantinya, jika tidak nilai tengah akan dikurangi satu dan dimasukkin di indek akhirnya dalam perulangan nantinya.

C. Interpolation

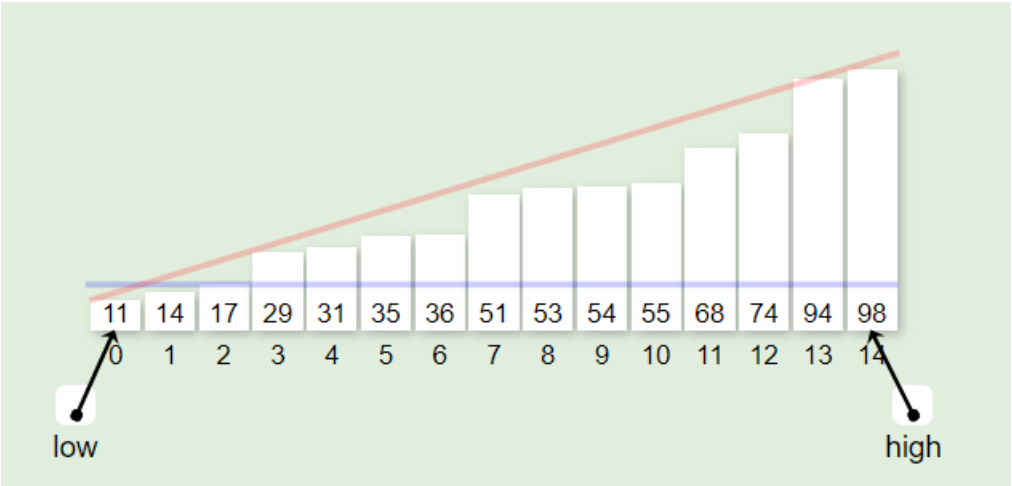
- Input (screenshot)

```
package Modul_Searching;
import java.util.Arrays;
import java.util.Scanner;
public class Tugas3 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        int array[] = {11, 14, 17, 29, 31, 35, 36, 51, 53, 54, 55, 68, 74, 94, 98};
        Arrays.sort(array);
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i]+" ");
        }

        System.out.print("\nMasukkan Data Pencarian : ");
        int cari = scan.nextInt();

        interpolationSearch(array, array.length, cari);
    }
    public static int interpolationSearch(int array[], int arrayLength, int cari) {
        int indekAkhir = 0, indekTengah = (arrayLength - 1);
        while (indekAkhir <= indekTengah && cari >= array[indekAkhir] && cari <= array[indekTengah]) {
            if (indekAkhir == indekTengah) {
                if (array[indekTengah] == cari) {
                    return 1;
                } else {
                    return -1;
                }
            }
            int position = indekAkhir + (((indekTengah - indekAkhir) / (array[indekTengah]
                    - array[indekAkhir])) * (cari - array[indekAkhir]));
            if (array[position] == cari) {
                System.out.println("Data Ditemukan di Indek "+position);
                return 1;
            } else if (array[position] < cari) {
                indekAkhir = position + 1;
            } else if (array[position] > cari) {
                indekTengah = position - 1;
            }
        }
        System.out.println("Data Tidak Ditemukan");
        return -1;
    }
}
```

- Output (screenshot)

```
run:
11 14 17 29 31 35 36 51 53 54 55 68 74 94 98
Masukkan Data Pencarian : 17
Data Ditemukan di Indek 2
```

- Ilustrasi dan penjelasan :



Pencarian Interpolasi adalah peningkatan dari Pencarian Biner misalnya, di mana nilai-nilai dalam array yang diurutkan didistribusikan secara seragam. Interpolasi membangun titik-titik data baru dalam kisaran kumpulan titik-titik data yang diketahui secara diskrit. Pencarian Biner selalu menuju ke elemen tengah untuk diperiksa. Di sisi lain, pencarian interpolasi dapat pergi ke lokasi yang berbeda sesuai dengan nilai kunci yang dicari. Misalnya, jika nilai kunci lebih dekat ke elemen terakhir, pencarian interpolasi kemungkinan akan memulai pencarian ke sisi akhir.

-