



Praktikum Day #3


Sorting dan Searching

Lab Assistant :
Fikri & Jo



Sekedar Mengingatkan

Function ?



```
public static void fungsi_gue(){  
    System.out.println("Ini adalah fungsi");  
}
```



Algoritma Searching

- Linear Search
- Binary Search



OPO SE SEARCHING IKU?

Proses mendapatkan data informasi berdasarkan kunci tertentu (Keinginan pengguna) dari sejumlah informasi yang telah disimpan

SEDANG MENCARI




KEADILAN DI INDONESIA



Jenis Searching

- **Single Match** : Pencarian yang menghasilkan satu data saja. Contoh : mencari mahasiswa dengan NRP yang spesifik
- **Multiple Match** : menghasilkan lebih dari satu data. Contoh : mencari mahasiswa dengan nama Fajar di kampus.





Linear Search

**Suatu teknik pencarian data yang akan
menelusuri tiap elemen satu per-satu dari
awal sampai akhir**



Case

Best Case : Saat data yang dicari terletak di indeks array awal sehingga waktu yang dibutuhkan untuk pencarian data sedikit

Worst Case : Saat data yang dicari terletak di indeks array terakhir



Contoh

Carilah data bernilai 6 terdapat pada urutan beberapa!

0	1	2	3	4	5	6	7	Index
8	10	6	-2	11	7	1	100	Value

Iterasi:

6 = 8 (TIDAK)

6 = 10 (TIDAK)

6 = 6 (YA)

Urutan = index + 1

Urutan = 2 + 1

Urutan = 3



Algoritma

```
procedure PencarianBeruntun(input  $a_1, a_2, \dots, a_n$  : integer,  $x$   
                           output  $idx$  : integer)
```

Deklarasi

```
   $k$  : integer  
   $ketemu$  : boolean    { bernilai true jika  $x$  ditemukan atau  $f$   
                        tidak ditemukan }
```

Algoritma:

```
   $k \leftarrow 1$   
   $ketemu \leftarrow \text{false}$   
  while ( $k \leq n$ ) and (not  $ketemu$ ) do  
    if  $a_k = x$  then  
       $ketemu \leftarrow \text{true}$   
    else  
       $k \leftarrow k + 1$   
    endif  
  endwhile  
  {  $k > n$  or  $ketemu$  }  
  
  if  $ketemu$  then    {  $x$  ditemukan }  
     $idx \leftarrow k$   
  else  
     $idx \leftarrow 0$       {  $x$  tidak ditemukan }  
  endif
```



Kompleksitas

Kasus terbaik terjadi bila $a_1 = x$

$$T(n) = 1$$

Kasus terburuk terjadi bila $a_n = x$ atau x tidak ditemukan

$$T(n) = n$$

Kasus rata rata

$$T(n) = (1 + 2 + 3 + 4 + \dots + n) / n = (n + 1) / 2$$

```
package Searching;

public class LinierSearchingV1 {

    public static void main(String[] args) {
        int [] data = {5, 29, 12, 15, 37, 23, 27, 38, 22, 54, 14, 78, 70};
        int key=22;
        int indeks;
        indeks= pencarianLinier(data, key);

        if (indeks !=0)
            System.out.println("Nomor "+key+" Berada Pada Urutan Ke - "+(indeks));
        else
            System.out.println("Nomor "+key+" Tidak ditemukan ");
    }

    public static int pencarianLinier(int[] data, int key){
        int k = 0;
        int n = data.length;
        boolean ketemu = false;
        while (k < n && ketemu==false){
            if(key==(data[k])){
                ketemu = true;    }
            else {
                k=k+1;    }
        }
        if (ketemu)
            return k+1;
        else
            return 0;
    }
}
```

Contoh Program 1

```
package Searching;
public class LinearSearching {
    static String [] Data = {"A", "B", "C", "D", "E", "F", "C"};

    public static void main(String[] args) {
        String karakter="K";
        searching(karakter);
    }

    public static void searching(String X){
        int k = 0;
        int n = Data.length;
        boolean ketemu = false;
        while (k < n && ketemu==false){
            if(X.equals(Data[k])){
                ketemu = true;}
            else {
                k=k+1;
            }
        }

        if(ketemu){
            System.out.println("Data "+ X +" berada pada urutan ke-"+(k+1));
        } else {
            System.out.println("Data "+ X +" Tidak Ditemukan");
        }
    }
}
```

Contoh Program 2



Binary Search

**Teknik pencarian data yang dimulai dari
pertengahan data yang telah terurut
dengan memperkecil lingkup pencarian
pada array.**

Contoh Data:

ILUSTRASI

Misalnya data yang dicari **17**

◦ 0	1	2	3	4	5	6	7	8
◦ 3	9	11	12	15	17	23	31	35
◦ A				B				C

◦ Karena $17 > 15$ (data tengah), maka: awal = tengah + 1

◦ 0	1	2	3	4	5	6	7	8
◦ 3	9	11	12	15	17	23	31	35
◦					A	B		C

◦ Karena $17 < 23$ (data tengah), maka: akhir = tengah – 1


◦ 0	1	2	3	4	5	6	7	8
◦ 3	9	11	12	15	17	23	31	35
◦					A=B=C			

◦ Karena $17 = 17$ (data tengah), maka KETEMU!



Cara Kerja Algoritma

1. Data Diambil dari posisi 1 sampai akhir
2. Cari posisi data tengah dengan $(\text{posisi awal} + \text{akhir})/2$
3. Jika data yang dicari lebih besar, maka nilai posisi awal = posisi tengah + 1
4. Jika data yang dicari lebih kecil, maka nilai posisi akhir = posisi tengah - 1
5. Jika data sama, berarti ketemu



```
procedure PencarianBiner(input  $a_1, a_2, \dots, a_n$  : integer,  $x$  : integer,  
                        output  $idx$  : integer)
```

Deklarasi

```
   $i, j, mid$  : integer  
  ketemu : boolean
```

Algoritma

```
   $i \leftarrow 1$   
   $j \leftarrow n$   
   $ketemu \leftarrow \text{false}$   
  while (not ketemu) and (  $i \leq j$  ) do  
     $mid \leftarrow (i+j) \text{ div } 2$   
    if  $a_{mid} = x$  then  
       $ketemu \leftarrow \text{true}$   
    else  
      if  $a_{mid} < x$  then      ( cari di belahan kanan )  
         $i \leftarrow mid + 1$   
      else                    ( cari di belahan kiri )  
         $j \leftarrow mid - 1$ ;  
      endif  
    endif  
  endwhile  
  (ketemu or  $i > j$  )  
  
  if ketemu then  
     $idx \leftarrow mid$   
  else  
     $idx \leftarrow 0$   
  endif
```

ALGORITMA



KOMPLEKSITAS

1. *Kasus terbaik*

$$T_{\min}(n) = 1$$

2. *Kasus terburuk:*

$$T_{\max}(n) = {}^2\log n + 1$$

$n=1$ → pencarian $T=1$

$n=2=2^1$ → pencarian $T=2$

$n=4=2^2$ → pencarian $T=3$

$n=8=2^3$ → pencarian $T=4 = 3 + 1$

:

$n=2^k$ → pencarian $T = k + 1 = {}^2\log n + 1$

```
package Searching;
public class BinarySearch {

    public static void main(String[] args) {
        int [] data = {5, 9, 12, 15, 17, 23, 27, 38, 42, 54, 64, 78, 90};
        int key=55;
        int indeks;
        indeks= pencarianBinary(data, key);

        if (indeks !=0)
            System.out.println("Nomor "+key+" Berada Pada Urutan Ke - "+(indeks));
        else
            System.out.println("Nomor "+key+" Tidak ditemukan ");
    }

    public static int pencarianBinary(int[] data, int key) {
        int bawah = 0;
        int atas = data.length - 1;

        while (atas >= bawah) {
            int tengah = (bawah + atas) / 2;
            if (key == data[tengah]){
                return (tengah+1);
            }
            else if (key < data[tengah]){
                atas = tengah - 1; }
            else{
                bawah = tengah + 1; }
        }
        return 0;
    }
}
```

Contoh Program 1

```
package Searching;
public class BinarySearch {
    static int [] data = {5, 9, 12, 15, 17, 23, 27, 38, 42, 54, 64, 78, 90}

    public static void main(String[] args) {
        int key=38;
        System.out.println(pencarianBinary(key));
    }

    public static String pencarianBinary(int key) {
        int bawah = 0;
        int atas = data.length - 1;

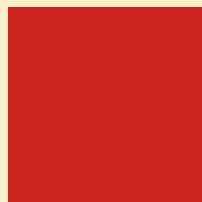
        while (atas >= bawah) {
            int tengah = (bawah + atas) / 2;
            if (key < data[tengah]){
                atas = tengah - 1;
            } else if (key == data[tengah]){
                return "Nomor "+key+" Berada Pada Urutan Ke - "+(tengah+1);
            } else{
                bawah = tengah + 1;
            }
        }
        return "Data Tidak Ditemukan";
    }
}
```

Contoh Program 2



SELESAI





Sorting dalam Algoritma?

“Algoritma yang berfungsi untuk meletakkan elemen-elemen dengan tipe data **numerik** maupun **karakter** ke dalam suatu **urutan tertentu**.”



Jenis-jenis Algoritma Sorting

Exchange Sort

- **Bubble Sort**
- Quick Sort

Selection Sort

- **Selection Sort**
- Heap Sort

Insertion Sort

- **Insertion Sort**
- Shell Sort

Merge Sort

- Merge Sort



Bubble Sort

“Exchange sort made simple”



How does it **works** /?

1. **Membandingkan** elemen sekarang dengan elemen berikutnya, anggaplah elemen sekarang sebagai elemen ke-**0** dan elemen berikutnya sebagai el. ke-**1**
2. Jika el. **0** $>$ el. **1**, maka kedua elemen tersebut **ditukar**.
3. Satu iterasi selesai, lanjutkan untuk semua elemen.
4. Proses sorting **berhenti apabila** seluruh elemen sudah diperiksa dan tidak ada pertukaran lagi yang terjadi, serta tercapai urutan yang telah diinginkan.



Let's Code !

Deklarasikan suatu array dengan elemen acak
 $\{7, 3, 6, 2, 1\}$

Untuk i dari 0 sampai $i < 4$ lakukan:

 Untuk j dari 0 sampai $j < 4-i$ lakukan:

 Jika $a[j] > a[j + 1]$ maka:

 Tukar $a[j]$ dengan $a[j + 1]$

Cetak array setelah sorting



Insertion Sort

"Key adalah kunci"



How does it **works** /?

1. Dimulai dari elemen ke-**1** dalam array, **anggap** elemen ke-0 sudah urut.
2. Elemen saat ini (elemen ke-**1**) adalah **key**, key ini berperan dalam perbandingan yang terjadi di dalam array nya.
3. **Bandingkan** key dengan elemen sebelumnya (elemen ke-0).
4. Jika elemen sebelumnya (elemen ke-0) $>$ key (elemen ke-1), **tukar** posisi antara elemen sebelumnya dengan key.
5. Iterasi pertama done!, lanjut ke iterasi selanjutnya.



Let's Code !

Deklarasikan suatu array dengan elemen acak
{5, 3, 4, 1, 2};

Untuk i dari 1 sampai $i < 5$ lakukan:

Simpan nilai $arr[i]$ ke dalam variabel key

Set $j = i - 1$

Selama $j \geq 0$ dan $arr[j] > key$ lakukan:

Geser elemen $arr[j]$ ke posisi $arr[j + 1]$

Kurangi j dengan 1

Tempatkan key pada posisi $arr[j + 1]$

Cetak array setelah sorting



Selection Sort

“pilih, bandingkan semua, baru tukar”



How does it **works** /?

1. Dimulai dari elemen ke-**0** dalam array, **anggap** elemen ke-**0** adalah elemen paling kecil di array. Assign ke variabel misalnya adalah "min"
2. Bandingkan nilai array "min" terhadap semua elemen di dalam array.
3. Jika ada elemen dalam array $<$ "min", maka tukar. Jika tidak maka lanjutkan saja ke iterasi berikutnya.
4. Iterasi pertama selesai, lanjut.



Let's Code !

Deklarasikan suatu array dengan elemen acak
misal :

{64, 25, 12, 22, 11};

Untuk i dari 0 sampai panjang i < 4 lakukan:

Set min = i

Untuk j dari i + 1 sampai j < 5 lakukan:

Jika $\text{arr}[j] < \text{arr}[\text{min}]$ maka:

Set min = j

Tukar $\text{arr}[i]$ dengan $\text{arr}[\text{min}]$

Cetak array setelah sorting



Latihan

Mas Jo ingin mensortir alphabet acak sebagai berikut :

'd','c','a',b,'f','e'

Bantu mas jo untuk menyusun alphabet menggunakan bahasa pemrograman java dengan menggunakan **Bubble Sort** !.