

# **TUGAS MINGGU 6**

## **Buble Sort, Selection Sort, Sequential search, Binary Search**



**GOLONGAN A**

**Disusun oleh :**

**Nama : Noga Muktiwati**

**NIM : E41200415**

**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNOLOGI INFORMASI POLITEKNIK  
NEGERI JEMBER 2020**

## Tugas Minggu 6

### A. Buble Sort

- ❖ **Ascending : Pengurutan dari (kecil – besar)**
- ❖ **Descending : Pengurutan dari (besar – kecil)**

**a. Pengertian**

Bubble sort ialah algoritma sorting paling sederhana yaitu pengurutan data dengan cara membandingkan elemen yang sekarang dengan elemen yang berikutnya, jika elemen sekarang > elemen berikutnya maka tukar.

**b. Langkah**

Ada dua langkah/ tindakan yang terus dilakukan secara berulang hingga data berurutan. Dua tindakan tersebut adalah :

- 1) Membandingkan 2 item
- 2) Menukar (swap) dua item, atau menyalin (copy) satu item

**c. Algoritma sorting**

- a) Bandingkan 2 item
- b) Jika item pertama ( sebelah kiri) > item ke dua (sebelah kanan) lakukan swap
- c) Pindah posisi kanan. Lakukan langkah dan b hingga posisi sampai di batas akhir
- d) Ketika satu item sudah beberapa sesuai urutan ulangi langkah a-c hingga semua item sesuai urutan.

**d. PseudoCode – Buble Sort**

```
public void bubble_sort ( int arr[] ){
    int N = arr.length;
    int temp;
    for( int i = N - 1; i >= 0; i--) {
        for( int j = 0; j < i; j++) {
            if( arr[j] > arr [j+1]) {
                // SWAP
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

e. **Implementasi Kode Buble Sort**

❖ **Acending**

- ✓ Implementasikan baris kode ke dalam sebuah program java, sehingga dapat digunakan untuk mengurutkan elemen array !

```

6 package bubblesort;
7 /**
8  *
9  * @author NOGA MUKTIWATI
10 */
11 public class BubleSort {
12     static void bubbleSort(int[] noga) {
13         int n = noga.length;
14         int temp = 0;
15         for (int i = 0; i < n; i++) {
16             for (int j = 1; j < (n - i); j++) {
17                 if (noga[j - 1] > noga[j]) {
18                     temp = noga[j - 1];
19                     noga[j - 1] = noga[j];
20                     noga[j] = temp;
21                 }
22             }
23         }
24     }
25     public static void main(String[] args) {
26         int noga[] = {1, 3, 2, 5, 4};
27         System.out.println("Array before bubble sort : ");
28         for (int i = 0; i < noga.length; i++) {
29             System.out.print(noga[i] + " ");
30         }
31         System.out.println();
32         bubbleSort(noga);
33         System.out.println("Array after bubble sort : ");
34         for (int i = 0; i < noga.length; i++) {
35             System.out.print(noga[i] + " ");
36         }
37     }
38 }

```

- ✓ Insert 5 item pada array program.

```

26 | int noga[] = {1, 3, 2, 5, 4};

```

- ✓ Tampilkan isi array sebelum dilakukan sorting. Kemudian urutkan dan tampilkan isi array setelah dilakukan sorting. Jalankan program yang telah Anda buat dan tulis output program tersebut!

```

Output - BubleSort (run) x
run:
Array before bubble sort :
1 3 2 5 4
Array after bubble sort :
1 2 3 4 5 BUILD SUCCESSFUL (total time: 0 seconds)

```

- ✓ Untuk sorting array yang memiliki 5 item, berapa jumlah perbandingan item yang dilakukan hingga semua item sesuai urutan?

**Ada 10 kali perbandingan hingga semua item terurut.**

```

run:
Array before bubble sort :
1 3 2 5 4
A :1 < B :3
A :3 < B :2
A :3 < B :5
A :5 < B :4
A :1 < B :2
A :2 < B :3
A :3 < B :4
A :1 < B :2
A :2 < B :3
A :1 < B :2
Array after bubble sort :
1 2 3 4 5 BUILD SUCCESSFUL (total time: 0 seconds)

```

- ✓ Tambahkan kode program pada listing di atas untuk menampilkan isi array sebelum baris kode pertukaran/swap.

```

static void bubbleSort(int[] noga) {
    int n = noga.length;
    int temp = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 1; j < (n - i); j++) {
            System.out.println ("A :"+ noga[j - 1]+ " < "+"B :"+ noga[j]);
            if (noga[j - 1] > noga[j]) {
                //swap the element
                temp = noga[j - 1];
                noga[j - 1] = noga[j];
                noga[j] = temp;
            }
        }
    }
}

```

- ✓ Tuliskan isi array pada 4 perulangan pertama. Jelaskan!

```
run:
Array before bubble sort :
1 3 2 5 4
A :1 < B :3
A :3 < B :2 → swap
A :3 < B :5
A :5 < B :4 → swap
A :1 < B :2
A :2 < B :3
A :3 < B :4
A :1 < B :2
A :2 < B :3
A :1 < B :2
Array after bubble sort :
1 2 3 4 5 BUILD SUCCESSFUL (total time: 0 seconds)
```

✓ Silahkan modifikasi listing kode algoritma bubble sort di atas sehingga bisa mengurutkan secara descending

❖ **Descending**

❖ **Source code:**

```
BubbleSort.java x Descending.java x
package bubblesort;

/**
 *
 * @author NOGA MUKTIWATI
 */
public class Descending {
    static void bubbleSortdesc(int[] noga) {
        int n = noga.length;
        int temp = 0;
        for (int i = 0; i < n; i++) {
            for (int j = 1; j < (n - i); j++) {
                if (noga[j - 1] < noga[j]) {
                    temp = noga[j - 1];
                    noga[j - 1] = noga[j];
                    noga[j] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        int noga[] = {3, 95, 35, 2, 12, 320, 97};
        System.out.println("Array before bubble sort : ");
        for (int i = 0; i < noga.length; i++) {
            System.out.print(noga[i] + " ");
        }
        System.out.println();
        bubbleSortdesc(noga);
        System.out.println("Array after bubble sort descending : ");
        for (int i = 0; i < noga.length; i++) {
            System.out.print(noga[i] + " ");
        }
    }
}
```

❖ **Output :**

```
Output - Buble Sort (run) x
run:
Array before bubble sort :
3 95 35 2 12 320 97
Array after bubble sort descending :
320 97 95 35 12 3 2 BUILD SUCCESSFUL (total time: 0 seconds)
```

f. **Praktik BKPM Minggu 6**

✓ **Buble Sort Source Code:**

```
BubleSort.java x
/*
 * @author NOGA MUKTIWATI
 */
public class BubleSort {
    static void BubleSort(int [] arr)
    {
        int n = arr.length;
        int temp = 0;
        for (int i = 0; i < n; i++)
        {
            for (int j = 1; j < (n-i); j++)
            {
                if (arr [j-1] > arr [j]) {
                    //swap element
                    temp = arr [j-1];
                    arr[j-1] = arr [j];
                    arr[j] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        int arr [] = {3, 60, 2, 45, 320,5};
        System.out.println ("Array Before Buble Sort");
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr [i] + "");
        }

        System.out.println ();
        BubleSort (arr);

        System.out.println ("Array After Buble Sort");
        for (int i = 0; i < arr.length; i++){
            System.out.print (arr[i] + "");
        }
    }
}
```

✓ **Output**

```
Output - Buble Sort (run) x
run:
Array Before Buble Sort
3602453205
Array After Buble Sort
2354560320BUILD SUCCESSFUL (total time: 0 seconds)
```

✓ **SequentialSearch**

Algoritma pencarian adalah algoritma yang menerima sebuah argumen kunci dan dengan langkah-langkah tertentu akan mencari rekaman dengan kunci tersebut.

Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu:

**Data yang dicari ditemukan (successful) tidak ditemukan (unsuccessful).** Sekuensial search disebut dengan pencarian linier yang merupakan metode pencarian paling sederhana.

Prinsip: data yang ada dibandingkan satu per satu secara berurutan dengan yang dicari sampai data tersebut ditemukan atau tidak ditemukan.

Algoritma:

- 1)  $i \Downarrow 0$
- 2) ketemu  $\Downarrow$  false
- 3) Selama (tidak ketemu) dan  $(i \leq N)$  kerjakan baris 4
- 4) Jika  $(Data[i] = x)$  maka ketemu  $\Downarrow$  true, jika tidak  $i \Downarrow i+1$
- 5) Jika (ketemu) maka  $i$  adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.

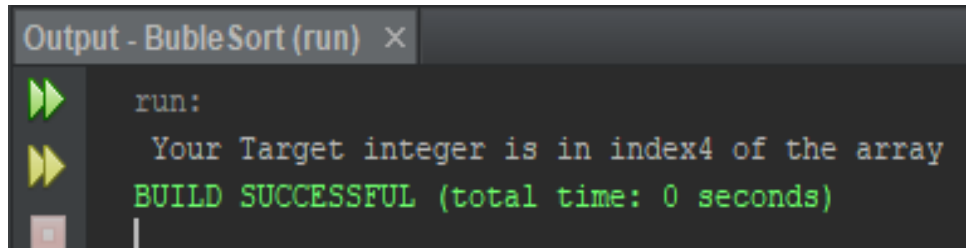
Pencarian sekuensial search pada dasarnya hanya melakukan pengulangan dari 1 sampai dengan jumlah data, dimana setiap pengulangan, dibandingkan data ke- $i$  dengan yang dicari sampai data ditemukan atau tidak ditemukan. Sehingga pada kasus pencarian yang paling buruk, untuk sebanyak  $N$  elemen data harus dilakukan pencarian sebanyak  $N$  kali itu.

### ✓ Source Code sequential search

```
6 package bubblesort;
7 /**
8  *
9  * @author NOGA MUKTIWATI
10 */
11 public class SequentialSearch {
12     public static void main(String[] args) {
13         int[] exampleVariableOne = {2, 9, 6, 7, 4, 5, 3, 0, 1};
14         int target = 4;
15         sequentialSearch(exampleVariableOne, target);
16     }
17
18     public static void sequentialSearch(int[] parameterOne, int parameterTwo) {
19         int index = -1;
20         // searches each index of the array until it reaches the last index
21         for (int i = 0; i < parameterOne.length; i++) {
22             if (parameterOne[i] == parameterTwo) {
23                 index = i;
24                 break;
25             }
26         }
27         if (index == -1) {
28             System.out.println(" Your Target integer does not exist in the array");
29         } else {
30             System.out.println(" Your Target integer is in index" + index + " of the array");
31         }
32     }
33 }
34 }
```

### ✓ Output





```
Output - BubleSort (run) x
run:
Your Target integer is in index4 of the array
BUILD SUCCESSFUL (total time: 0 seconds)
```

- g. Carilah referensi tentang teknik selection sort dan buatlah resume meliputi: pengertian dan algoritma!

## Selection Sort

**Selection sort** merupakan sebuah teknik pengurutan dengan cara mencari nilai tertinggi / terendah di dalam array kemudian menempatkan nilai tersebut di tempat semestinya.

- ❖ (**Ascending**) = kecil ke besar
- ❖ (**Descending**).= besar ke kecil

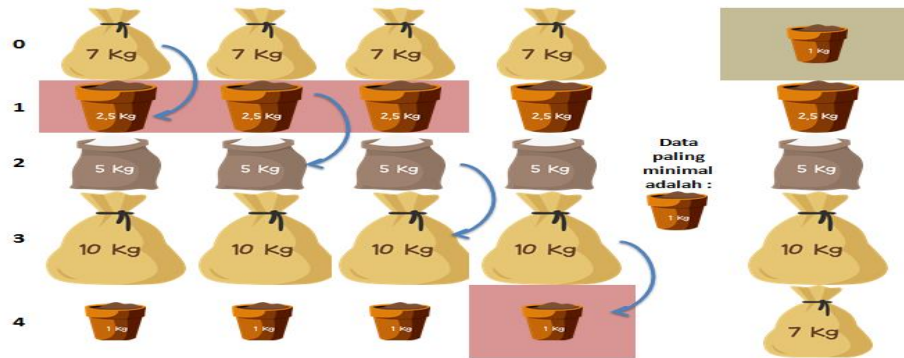
Berikut tahapan algoritma selection sort (ascending order):

1. Algoritma ini dimulai dari indeks awal sampai dengan indeks akhir data
2. Cari data dengan nilai paling minimal (dari indeks awal sampai dengan indeks akhir) melalui proses perbandingan.
3. Letakkan data minimal ini di indeks awal
4. Ulangi lagi proses pencarian data paling minimal (dari indeks awal+1 sampai dengan indeks akhir, karena indeks awal sudah terisi data yang tepat).
5. Letakkan data ini pada indeks awal+1
6. Ulangi lagi proses pencarian data paling minimal (dari indeks awal+2 sampai dengan akhir) dan letakkan data ini pada indeks awal+2, dst.

Misalkan terdapat suatu data, yaitu **[7, 2.5, 5, 10, 1]**. Maka proses pengurutan secara ascending (dari nilai kecil sampai nilai besar) dengan menggunakan algoritma Selection Sort adalah sebagai berikut :

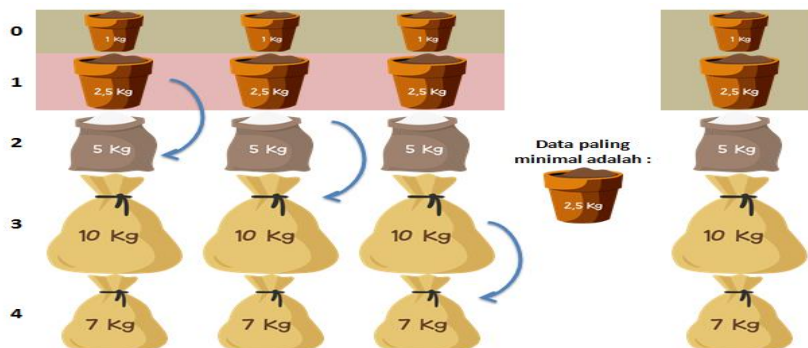
**Iterasi Pertama.** Pada iterasi pertama ini, data yang memiliki nilai paling kecil, akan menempati posisi dengan indeks paling awal. Untuk mencari data dengan nilai paling kecil ini, dilakukan perbandingan sebagai berikut:

1. antara indeks ke-0 dengan ke-1, sehingga nilai terkecil saat ini adalah 2.5
2. antara indeks ke-1 dengan ke-2, nilai terkecil tetaplah 2.5
3. antara indeks ke-2 dengan ke-3, nilai terkecil tetaplah 2.5
4. antara indeks ke-3 dengan ke-4, didapatkan nilai terkecil adalah 1. Pada saat pencarian nilai terkecil ini tidak dilakukan proses pertukaran posisi (berbeda dengan algoritma bubble sort). Setelah didapatkan nilai terkecil, maka nilai 1 ini akan ditukar dengan data pada posisi dengan indeks ke-0, sehingga setelah iterasi pertama berakhir, data 1, akan menempati indeks ke-0. Ilustrasi di lihat pada gambar di bawah ini



**Iterasi Kedua.** Pada iterasi kedua ini, data yang memiliki nilai paling kecil kedua, akan menempati posisi dengan indeks setelah data paling kecil, yaitu indeks ke-1. Untuk mencari data dengan nilai paling kecil kedua ini, dilakukan perbandingan, hanya saja data pada indeks ke-0 tidak perlu dibandingkan lagi, karena data pada indeks ke-0 sudah berisi data yang tepat. Perbandingan yang dilakukan untuk mencari nilai minimal kedua adalah sebagai berikut:

1. antara indeks ke-1 dengan ke-2, nilai terkecil tetaplah 2.5
2. antara indeks ke-2 dengan ke-3, nilai terkecil tetaplah 2.5
3. antara indeks ke-3 dengan ke-4, didapatkan nilai terkecil adalah 2.5 Setelah didapatkan nilai minimal, yaitu 2.5, maka nilai ini akan ditukar dengan data pada posisi dengan indeks ke-1, sehingga setelah iterasi kedua berakhir, data 2.5, akan menempati indeks ke-1. Ilustrasi iterasi kedua ini dapat dilihat pada Gambar 2.



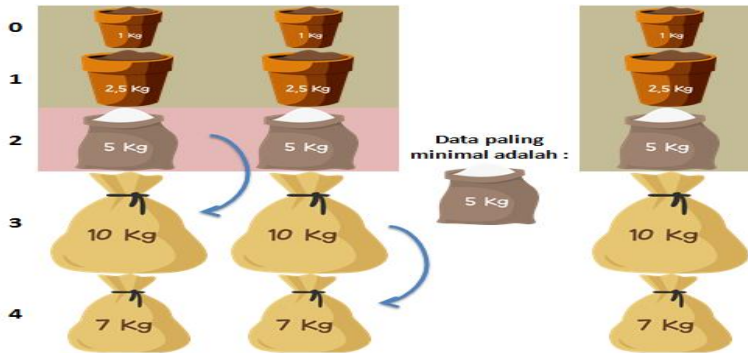
Gambar 2. Iterasi Kedua Algoritma Selection Sort

**Iterasi Ketiga.** Iterasi ketiga bertujuan untuk menempatkan data minimal yang ketiga pada posisi yang tepat, yaitu indeks ke-2. Untuk mencari data minimal ketiga ini, dilakukan perbandingan, hanya saja data pada indeks ke-0 dan ke-1 tidak perlu dibandingkan lagi, karena data pada indeks ke-0 dan ke-1, sudah berisi data yang tepat.

Perbandingan yang dilakukan untuk mencari nilai minimal kedua adalah sebagai berikut:

1. antara indeks ke-2 dengan ke-3, nilai terkecil adalah 5
2. antara indeks ke-3 dengan ke-4, didapatkan nilai terkecil adalah 5 Setelah didapatkan nilai minimal, yaitu 5, maka nilai ini akan ditukar dengan data pada posisi dengan indeks

ke-2, sehingga setelah iterasi kedua berakhir, data 5, akan menempati indeks ke-2. Ilustrasi iterasi kedua ini dapat dilihat pada Gambar 3.



Gambar 3. Iterasi Ketiga Algoritma Selection Sort

Proses ini dilakukan secara terus menerus sampai proses pengecekan dilakukan pada data terakhir, sehingga akhir dari algoritma selection sort ini, data sudah dalam keadaan terurut, seperti yang terlihat pada Gambar 4.



Gambar 3. Iterasi Terakhir Algoritma Selection Sort

- h. Carilah referensi tentang teknik binary search dan buatlah resume meliputi: pengertian dan algoritma

### Binary Search

**Binary search adalah** algoritma pencarian untuk data yang terurut. Pencarian dilakukan dengan cara menebak apakah data yang dicari berada ditengah-tengah data, kemudian membandingkan data yang dicari dengan data yang ada ditengah. Bila data yang ditengah sama dengan data yang dicari, berarti data ditemukan.

### Pencarian Biner (Binary Search) dilakukan untuk :

1. Memperkecil jumlah operasi pembandingan antara data yang dicari dg data di dalam tabel, khususnya untuk jumlah data ukuran besar.
2. Prinsip dasarnya : melakukan proses pembagian ruang pencarian secara berulang sampai data ditemukan/sampai ruang pencarian tak dapat dibagi lagi (berarti ada kemungkinan data tidak ditemukan)
3. Syarat utama untuk pencarian biner adalah data di dalam tabel harus sudah terurut misalkan terurut menaik

Algoritmanya :

```
Kamus
Const N : integer = 8 { misalkan jumlah elemen array maksimum = 8 }
Type A = array [ 1 ..... N ] of integer
Cari, BatasAtas, BatasBawah, Tengah : Integer
Ketemu : boolean
ALGORITMA
Input (cari) { meminta nilai data yang akan dicari}
BatasAtas ← 1 { indeks array dimulai dari 1 }
BatasBawah ← N
Ketemu ← False
While (BatasAtas < BatasBawah) and (not ketemu) do
    Tengah ← (BatasAtas + BatasBawah) div 2
    If A [Tengah] = cari then
        Ketemu ← true
    Else
        If ( A [Tengah] < cari ) then { cari di bagian kanan }
            BatasAtas ← Tengah + 1
        Else
            BatasBawah ← Tengah - 1 { cari di bagian kiri }
        Endif
    Endif
EndWhile
If (ketemu) then
    Output ( 'Data berada di index nomor', Tengah )
Else
    Output ( 'Data tidak ditemukan' )
Endif
```

Contoh Nilai-Nilai data yang sudah terurut :

A	2	5	8	12	15	25	37	57
	1	2	3	4	5	6	7	8

**Kasus 1 : cari = 12**

**Loop pertama :**  $Tengah = (BatasAtas + BatasBawah) \div 2 = (1 + 8) \div 2 = 4$

$A[Tengah] = A[4] = 12$ , berarti loop pertama data langsung ditemukan

**Kasus 2 : cari = 15**

**Loop pertama :**  $Tengah = (BatasAtas + BatasBawah) \div 2 = (1 + 8) \div 2 = 4$

$A[Tengah] = A[4] = 12 < cari = 15$ , berarti  $BatasAtas = Tengah + 1 = 4 + 1 = 5$

**Loop kedua :**  $Tengah = (BatasAtas + BatasBawah) \div 2 = (5 + 8) \div 2 = 6$

$A[Tengah] = A[6] = 25 > cari = 15$ , berarti  $BatasBawah = Tengah - 1 = 6 - 1 = 5$

**Loop ketiga :**  $Tengah = (BatasAtas + BatasBawah) \div 2 = (5 + 5) \div 2 = 5$

$A[Tengah] = A[5] = 15$ , berarti setelah loop ketiga, data ditemukan

**Kasus 3 : cari = 10**

**Loop pertama :**  $Tengah = (BatasAtas + BatasBawah) \div 2 = (1 + 8) \div 2 = 4$

$A[Tengah] = A[4] = 12 > cari = 10$ , berarti  $BatasBawah = Tengah - 1 = 4 - 1 = 3$

**Loop kedua :**  $Tengah = (BatasAtas + BatasBawah) \div 2 = (1 + 3) \div 2 = 2$

$A[Tengah] = A[2] = 5 < cari = 10$ , berarti  $BatasAtas = Tengah + 1 = 2 + 1 = 3$

**Loop ketiga :**  $Tengah = (BatasAtas + BatasBawah) \div 2 = (3 + 3) \div 2 = 3$

$A[Tengah] = A[3] = 8$ , berarti setelah loop ketiga, data tidak ditemukan

Untuk jumlah data sebanyak  $n$ , maka proses perbandingan maksimal sebanyak  $(\log n)$  kali. Untuk contoh di atas, jumlah data 8, maka proses perbandingan maksimal sebanyak 3 kali.

**TERIMAKASIH ☺**