PRAKTIKUM 9-10
ALGORITMA PENGURUTAN (INSERTION DAN SELECTION)

A. TUJUAN PEMBELAJARAN

1. Memahami mengenai algoritma pengurutan *insertion sort* dan *selection sort.*
2. Mampu mengimplementasikan algoritma pengurutan *insertion sort* dan *selection sort* secara *ascending* dan *descending.*

B. DASAR TEORI

1. Algoritma *Insertion Sort*

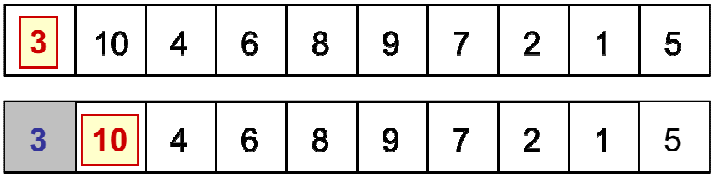
Salah satu algoritma sorting yang paling sederhana adalah *insertion sort*. Algoritma *insertion sort* pada dasarnya memilah data yang akan urutkan menjadi 2 bagian, yang belum diurutkan dan yang sudah diurutkan. Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkan sesuai posisinya pada bagian lain dari array yang telah diurutkan. Langkah ini dilakukan secara berulang hingga tak ada lagi elemen tersisa pada bagian array yang belum diurutka.

Metode insection sort merupakan metode yang mengurutkan bilangan-bilangan yang telah terbaca, dan berikutnya secara berulang akan menyisipkan bilangan-bilangan dalam array yang belum terbaca ke sisi kiri array yang telah terurut. Kita mengambil pada bilangan yang paling kiri. Bilangan tersebut dikatakan urut terhadap dirinya sendiri karena bilangan yang di bandingkan baru 1.

3	10	4	6	8	9	7	2	1	5

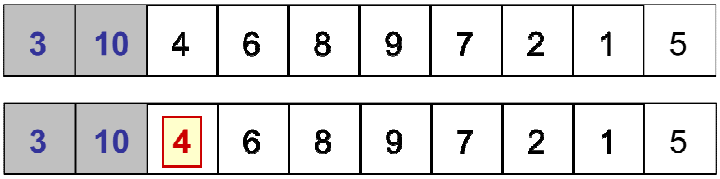
Gambar 1. Langkah-langkah pengurutan metode Insertion Sort (1)

Cek bilangan ke 2 (10) apakah lebih kecil dari bilangan yang ke 1(3).Apabila lebih kecil maka ditukar. Tapi kali ini bilangan ke 1 lebih kecil dari bilangan ke 2 maka tidak ditukar.



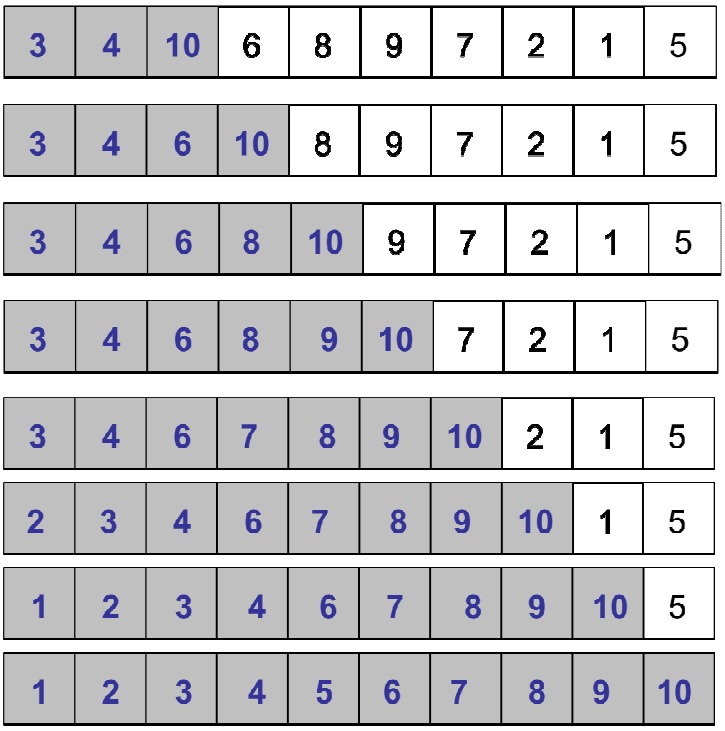
Gambar 2. Langkah-langkah pengurutan metode Insertion Sort (2)

Pada kotak warna abu2 sudah dalam keadaan terurut. Kemudian membandingkan lagi pada bilangan selanjutnya yaitu bilangan ke 3 (4). Bandingkan dengan bilangan yang ada di sebelah kirinya. Pada kasus ini bilangan ke 2 bergeser dan digantikan bilangan ke 3.



Gambar 3. Langkah-langkah pengurutan metode Insertion Sort (3)

Lakukan langkah seperti di atas pada bilangan selanjutnya. 4 bilangan pertama sudah dalam keadaan terurut relatif. Ulangi proses tersebut sampai bilangan terakhir disisipkan.

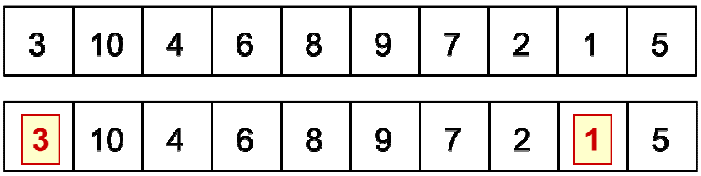


Gambar 4. Langkah-langkah pengurutan metode Insertion Sort (4)

2. Algoritma *Selection Sort*

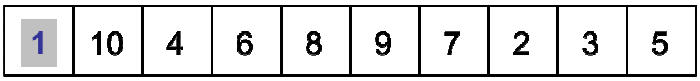
Ide utama dari algoritma *selection sort* adalah memilih elemen dengan nilai paling rendah dan menukar elemen yang terpilih dengan elemen ke-*i*. Nilai dari *i* dimulai dari 1 ke *n*, dimana *n* adalah jumlah total elemen dikurangi 1.

Cek seluruh array dan cari array yang mempunyai nilai terkecil → index 8 (1). Setelah ketemu tukar dengan array yang berada di pojok kiri (3).



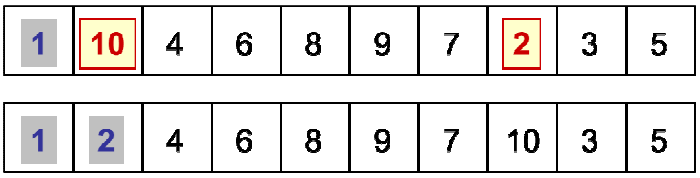
Gambar 5. Langkah-langkah pengurutan metode Selection Sort (1)

Setelah di tukar bagian yang berwarna abu-abu merupakan index yang telah terurutkan.



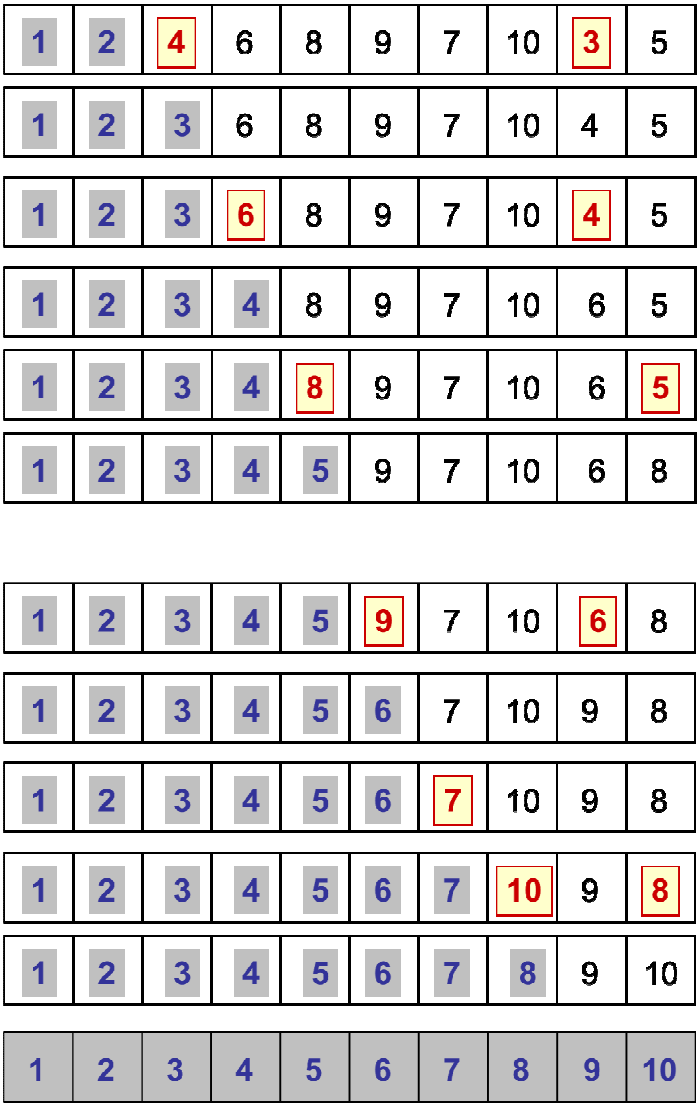
Gambar 6. Langkah-langkah pengurutan metode Selection Sort (2)

Kemudian cari bilangan terkecil selanjutnya (selain di kotak abu-abu) yaitu bilangan 2 dan tukar dengan sebelah array yang telah terurutkan.



Gambar 7. Langkah-langkah pengurutan metode Selection Sort (3)

Dua array sudah terurutkan. Kemudian ulangi langkah di atas dan lakukan langkah yang sama yaitu pilih terkecil dan tukar dengan sebelah array yang sudah terurutkan.



Gambar 7. Langkah-langkah pengurutan metode Selection Sort (4)

C. TUGAS PENDAHULUAN

Jawablah pertanyaan berikut ini :

1. Tuliskan algoritma pengurutan *insertion sort* secara *ascending.*
2. Tuliskan algoritma pengurutan *selection sort* secara *ascending.*

D. PERCOBAAN

Percobaan 1 : *Insertion sort* secara *ascending*

```
public class Insertion {
```

```
   public static <T extends Comparable<? super T>> void
                 insertionSort(T[] arr){
      for (int i=arr.length-1 ; i>=0 ; i--) {
          for (int j = i+1, k = i; j <arr.length ; j++){
              if (arr[j].compareTo(arr[k]) > 0) {
                  break;
              } else {
                  T temp = arr[k];
                  arr[k] = arr[j];
                  arr[j] = temp;
                  k = j;
              }
          }
      }
    }

    public static <T> void tampil(T data[]) {
        for (T objek : data) {
            System.out.print(objek + " ");
        }
        System.out.println("");
    }

    public static void main(String[] args) {
        Integer data[] = new Integer[10];
        for(int a=0 ; a<data.length ; a++)
        {
            data[a]= (int)(Math.random()*20+1);
        }
        long awal = System.currentTimeMillis();
        insertionSort(data);
        long sisaWaktu = System.currentTimeMillis() - awal;
        tampil(data);
        System.out.println("Waktu " + sisaWaktu);
    }
}
```

Percobaan 2 : *Selection sort* secara *ascending*

```
public class Selection {
   public static <T extends Comparable<? super T>> void
               selectionSort(T[] arr) {
      T temp;
      for(int i=arr.length-1 ; i>=0 ; i--){
         int max = i;
         for(int j=i-1 ; j>=0 ; j--){
             if(arr[j].compareTo(arr[max]) > 0)
             max = j;
         }
         temp = arr[i];
         arr[i] = arr[max];
         arr[max] = temp;
      }
```

```
    }

    public static <T> void tampil(T data[]) {
        for (T objek : data) {
            System.out.print(objek + " ");
        }
        System.out.println("");
    }

    public static void main(String[] args) {

        Integer data[] = new Integer[10];
        for(int a=0 ; a<data.length ; a++){
            data[a]= (int)(Math.random()*13+1);
        }
        long awal = System.currentTimeMillis();
        selectionSort(data);
        long sisaWaktu = System.currentTimeMillis() - awal;
        tampil(data);
        System.out.println("Waktu " + sisaWaktu);
    }
}
```

E. LATIHAN

1. Dari percobaan 1 tambahkan method untuk melakukan pengurutan *insertion sort* secara *descending*.
2. Dari percobaan 2 tambahkan method untuk melakukan pengurutan *selection sort* secara *descending*.
3. Buatlah program untuk melakukan pengurutan *insertion sort* terurut dari kanan ke kiri secara *ascending* dan *descending*.
4. Buatlah program untuk melakukan pengurutan *selection sort* terurut dari kanan ke kiri secara *ascending* dan *descending*.
5. Buatlah class `Mahasiswa` dengan variable `nrp` dan `nama` yang memiliki tipe `String`! Class `Mahasiswa` mengimplementasikan interface `Comparable`, selanjutnya implementasikan fungsi abstract `compareTo()`, untuk membandingkan dua objek mahasiswa berdasarkan `nrp`.

```
public class Mahasiswa implements Comparable <Mahasiswa> {
    private String nrp ;
    private String nama ;
    @Override
```

```
    public int compareTo(Mahasiswa o) {…}

    @Override
    public String toString() {…}
}
```

Lakukan pengujian fungsi `insertionSort()` lagi, sebelumnya tambahkan pada fungsi main() seperti di bawah ini !

```
public class Latihan {
   public static <T extends Comparable<? super T>> void
                 insertionSort(T[] arr){
       ……
   }
   public static <T> void tampil(T data[]) {
      …….
   }
   public static void main(String[] args) {
        Mahasiswa arr8[] = {new Mahasiswa("02", "Budi"),
                            new Mahasiswa("01", "Andi"),
                            new Mahasiswa("04", "Udin"),
                            new Mahasiswa("03", "Candra")};
        long awal = System.currentTimeMillis();
        insertionSort (arr8);
        tampil(arr8);
        long sisaWaktu = System.currentTimeMillis() - awal;
        System.out.println("Waktu " + sisaWaktu);
    }
}
```

6. Pada soal 5 lakukan untuk algoritma *selection sort*.

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.