# SORTING <PENGURUTAN>

DWI RATNA S

# SORTING

Pengurutan data dalam struktur data sangat penting terutama untuk data yang beripe data numerik ataupun karakter. Pengurutan dapat dilakukan secara ascending (urut naik) dan descending (urut turun)

Pengurutan (Sorting) adalah proses pengurutan data yang sebelumnya disusun secara acak sehingga tersusun secara teratur menurut aturan tertentu

Algoritma pengurutan adalah algoritma yang meletakkan elemen-elemen suatu kumpulan data dalam urutan tertentuPengurutan

# KLASIFIKASI ALGORITMA PENGURUTAN

BERDASARKAN KOMPLEKSITAS

TEKNIK YANG DILAKUKAN

STABILITAS

MEMORI YANG DIGUNAKAN

REKURSIF/TIDAK

ATAUPUN PROSES YANG TERJADI.

# PENGURUTAN (SORTING)

Pengurutan (*Sorting*) adalah operasi yang sangat banyak dilakukan dalam ‘Bussiness Data Processing’.

Jenis-jenis Algoritma *Sorting* (Pengurutan) :

1. EXCHANGE SORT
2. SELECTION SORT
3. INSERTION SORT
4. MERGE SORT
5. HEAP SORT.

1. Exchange Sort

   Prinsip dari exchange sort adalah melakukan pembandingan antar data, dan melakukan pertukaran apabila urutan yang didapat belum sesuai. Contohnya adalah :

   Bubble *sort, Cocktail sort, Comb sort, Gnome sort, Quicksort.*

2. *Selection Sort*

   Prinsip utama algoritma dalam klasifikasi ini, adalah mencari elemen yang tepat untuk diletakkan di posisi yang telah diketahui, dan meletakkannya di posisi tersebut setelah data tersebut ditemukan. Algoritma yang dapat diklasifikasikan ke dalam kategori ini adalah :

   *Selection sort, Heapsort, Smoothsort, Strand sort.*

3. Insertion Sort

   Algoritma pengurutan yang diklasifikasikan ke dalam kategori ini mencari tempat yang tepat untuk suatu elemen data yang telah diketahui ke dalam subkumpulan data yang telah terurut, kemudian melakukan penyisipan (insertion) data di tempat yang tepat tersebut. Contohnya adalah :

   *Insertion sort, Shell sort, Tree sort, Library sort, Patience sorting.*

1. Merge Sort

Dalam algoritma ini kumpulan data dibagi menjadi subkumpulan subkumpulan yang kemudian subkumpulan tersebut diurutkan secara terpisah, dan kemudian digabungkan kembali dengan metode merging. Dalam kenyataannya algoritma ini melakukan metode pengurutan merge sort juga untuk mengurutkan subkumpulan data tersebut, atau dengan kata lain, pengurutan dilakukan secara rekursif. Contohnya adalah :

Merge sort.

.

*2. Non-Comparison Sort*

Sesuai namanya dalam proses pengurutan data yang dilakukan algoritma ini tidak terdapat pembandingan antardata, data diurutkan sesuai dengan pigeon hole principle. Dalam kenyataanya seringkali algoritma non-comparison sort yang digunakan tidak murni tanpa pembandingan, yang dilakukan dengan menggunakan algoritma- algoritma pengurutan cepat lainnya untuk mengurutkan subkumpulan-subkumpulan datanya. Contohnya adalah :

Radix sort, Bucket sort, Counting sort, Pigeonhole sort, Tally sort.

# TIGA ALGORITMA PENGURUTAN BERBASIS-PEMBANDINGAN

*Bubble Sort*

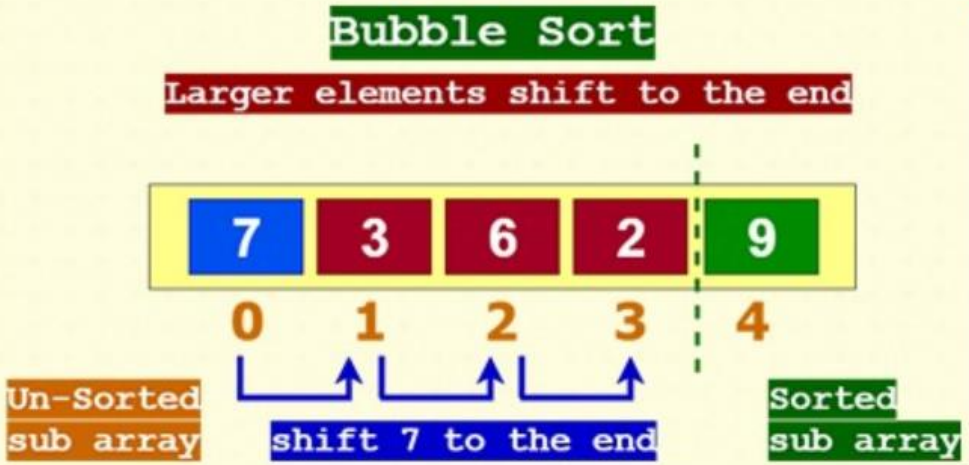
*Selection Sort*

*Insertion Sort*

# BUBBLE SORT

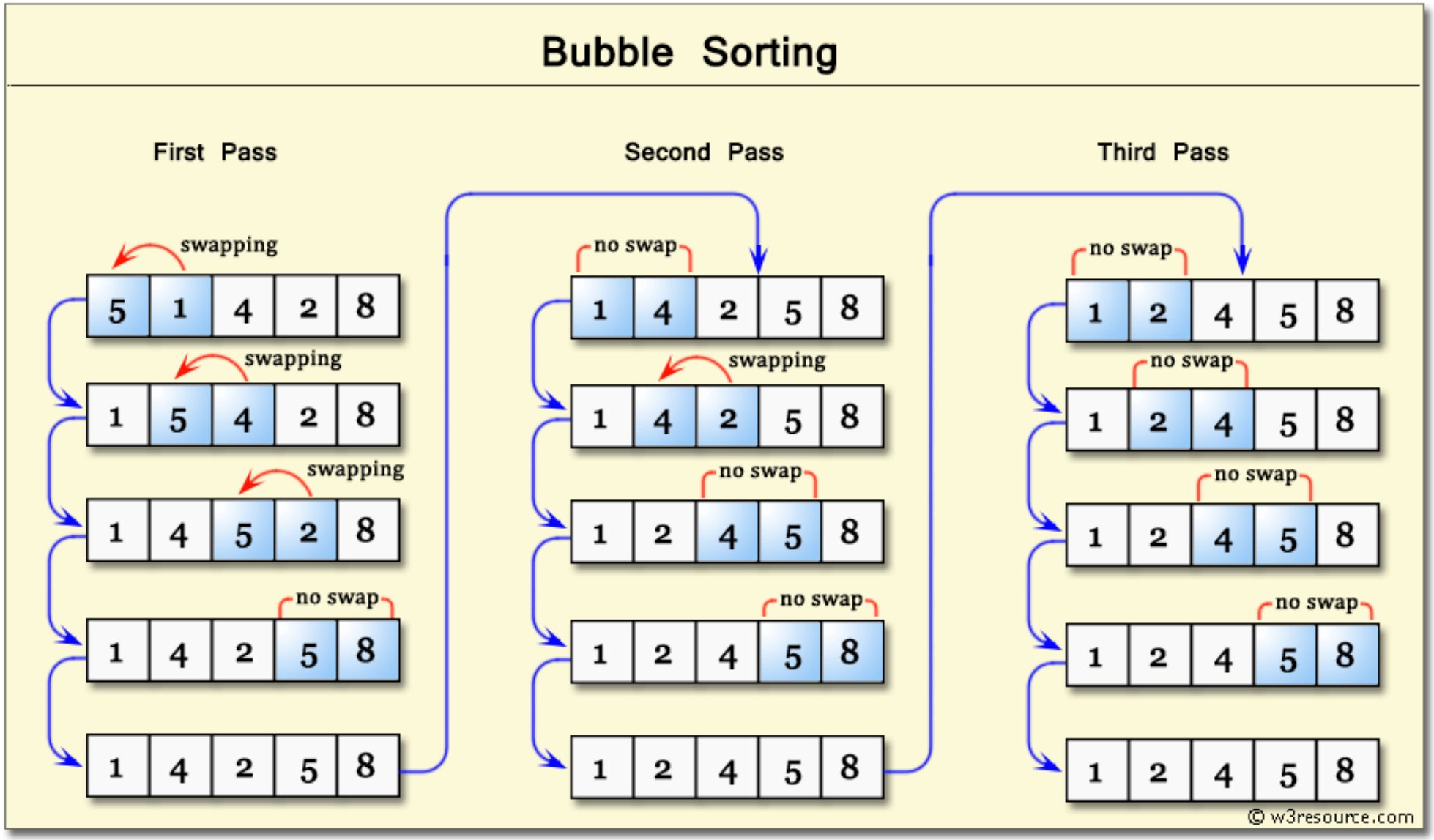
- Bubble Sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya.
- Jika elemen sekarang lebih besar dari elemen berikutnya maka kedua elemen tersebut ditukar, jika pengurutan ascending.
- Jika elemen sekarang lebih kecil dari elemen berikutnya, maka kedua elemen tersebut ditukar, jika pengurutan descending
- Ketika satu proses telah selesai, maka bubble sort akan mengulangi proses, demikian seterusnya.
- Kapan berhentinya? Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan, serta tercapai perurutan yang telah diinginkan.

# BUBBLE SORT ALGORITHM




```
void bubbleSort(int a[])
{
    for(int i=0; i<5;i++)
    {
        for(int j=0; j<(5-i-1); j++)
        {
            if(a[j]>a[j+1])
            {
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
}
```

# ALGORITMA

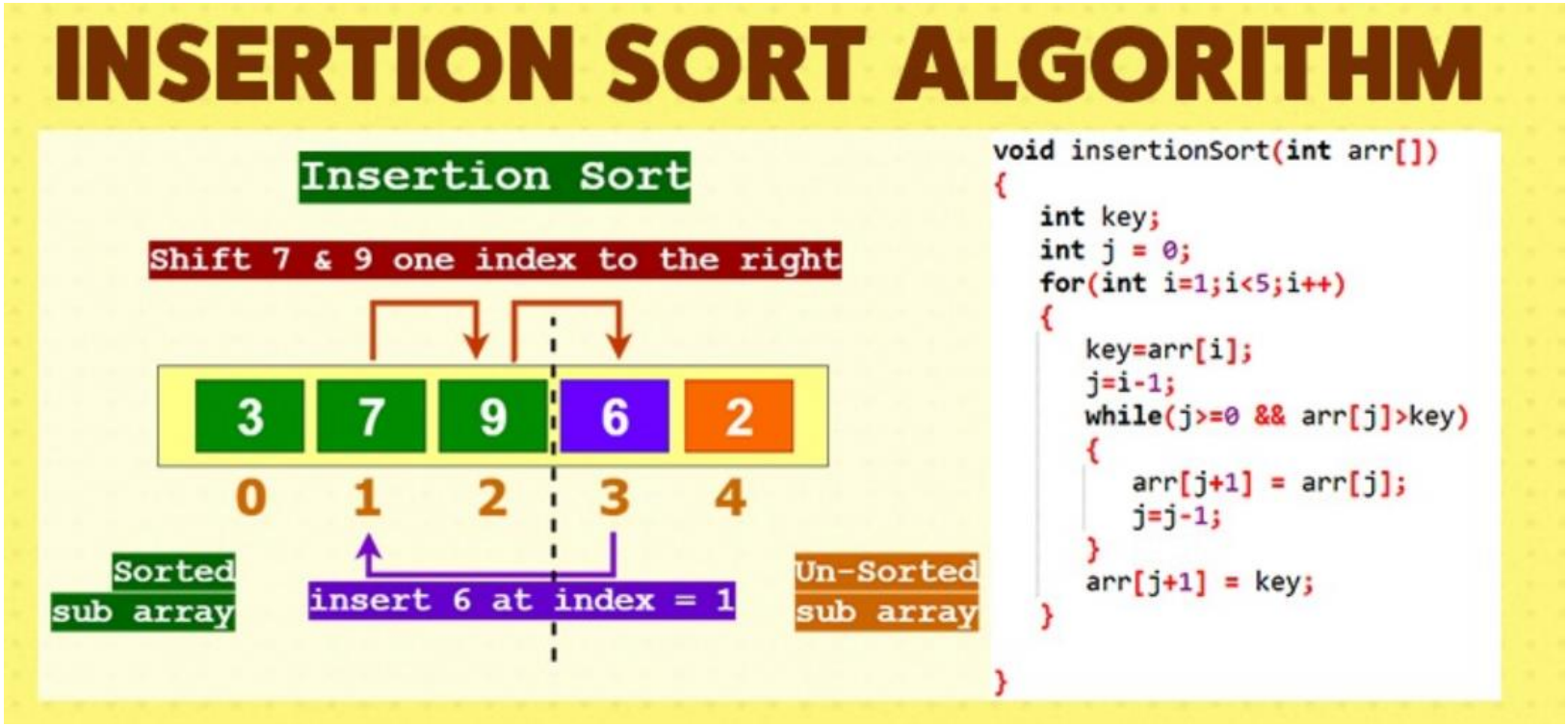

Bubble Sorting
First Pass
Second Pass
Third Pass
swapping
swapping
swapping
no swap
5 1 4 2 8
1 5 4 2 8
1 4 5 2 8
1 4 2 5 8
1 4 2 5 8
no swap
swapping
no swap
no swap
1 4 2 5 8
1 4 2 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
no swap
no swap
no swap
no swap
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
© w3resource.com

# INSERTION SORT

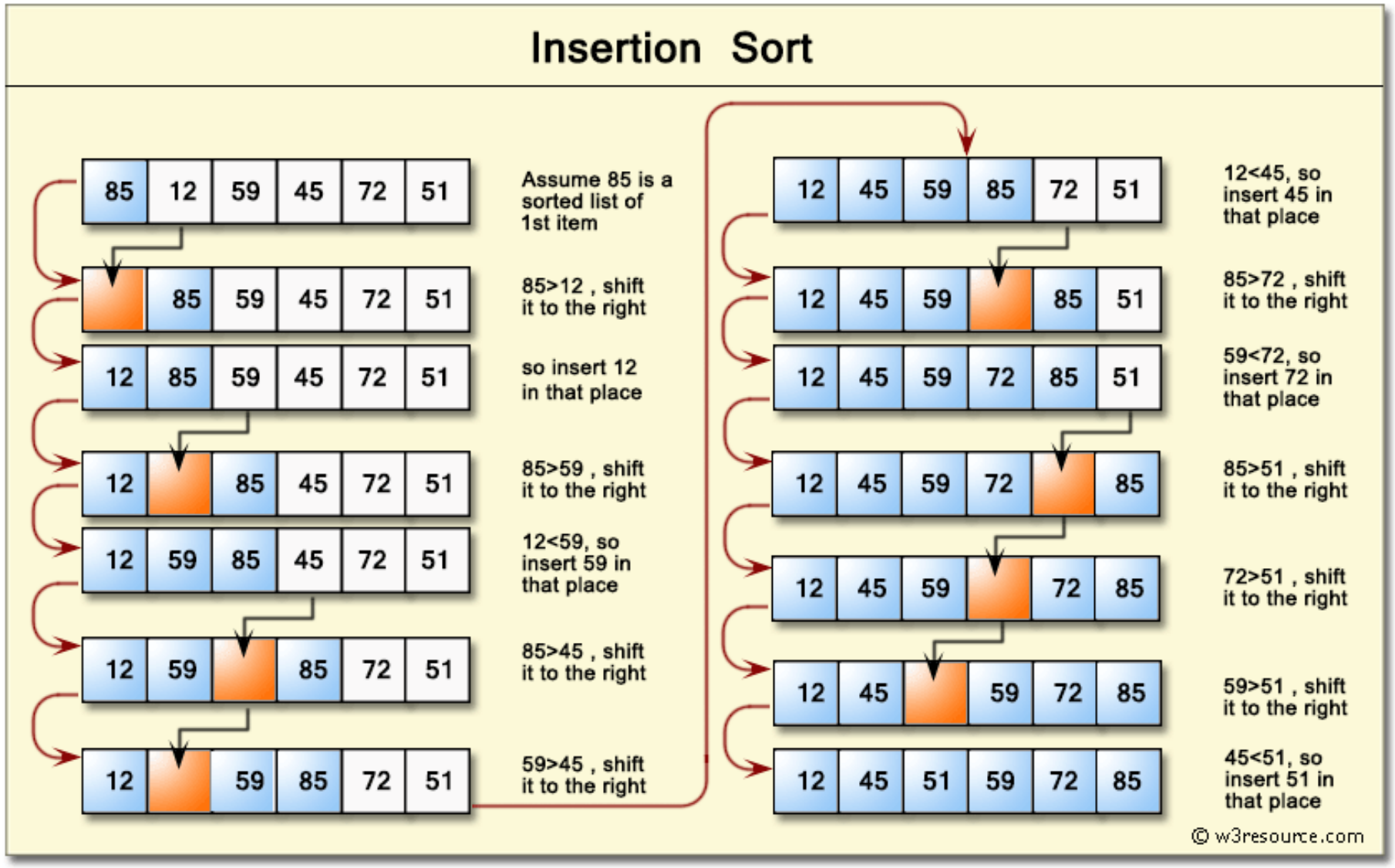
- Algoritma *insertion sort* pada dasarnya memilah data yang akan diurutkan menjadi dua bagian, yang belum diurutkan(meja pertama) danyang sudah diurutkan(meja kedua)
- Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkan sesuai posisinya pada bagian lain dari array yang telah diurutkan. Langkah ini dilakukan secara berulang hingga tidak ada lagi elemen yang tersisa pada bagian array yang belum diurutkan.
- Mengurutkan kartu dari kecils/d besar

- Data pada posisi ke i (x) dibandingkan dengan data pada posisi ke 0 sampai dengan i=1. Jika data ke j lebih besar dari pada x, maka data disisipkan ke posisi j dan data ke j+1 sampai dengan i digeser ke kanan.

(Dimulai dari data ke 2, bandingkan dengan data pertama)


INSERTION SORT ALGORITHM
Insertion Sort
Shift 7 & 9 one index to the right
3
7
9
6
2
0
1
2
3
4
Sorted sub array
insert 6 at index = 1
Un-Sorted sub array
void insertionSort(int arr[])
{
int key;
int j = 0;
for(int i=1;i<5;i++)
{
key=arr[i];
j=i-1;
while(j>=0 && arr[j]>key)
{
arr[j+1] = arr[j];
j=j-1;
}
arr[j+1] = key;
}
}

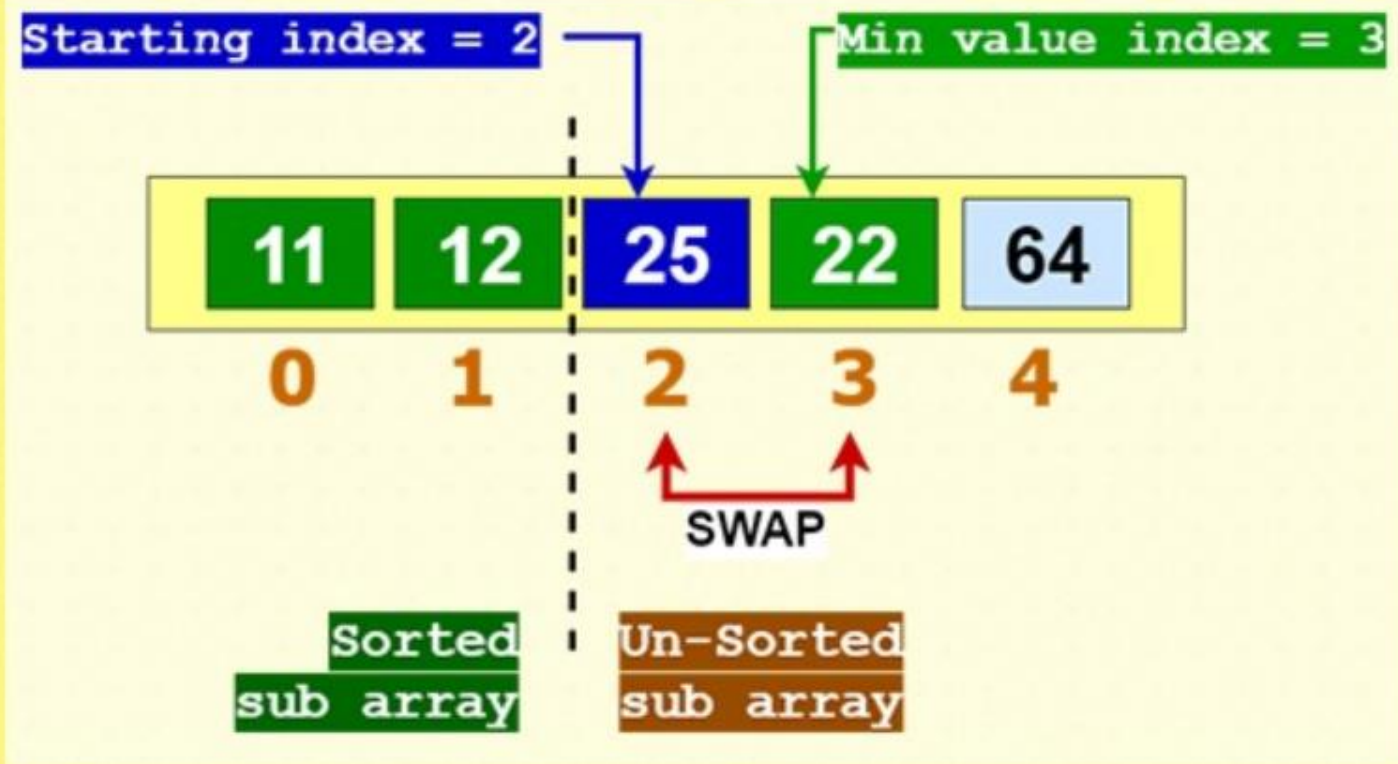
# ALGORITMA


Insertion Sort
85 12 59 45 72 51
Assume 85 is a sorted list of 1st item
85 59 45 72 51
85>12 , shift it to the right
12 85 59 45 72 51
so insert 12 in that place
12 85 45 72 51
85>59 , shift it to the right
12 59 85 45 72 51
12<59, so insert 59 in that place
12 59 85 72 51
85>45 , shift it to the right
12 59 85 72 51
59>45 , shift it to the right
12 45 59 85 72 51
12<45, so insert 45 in that place
12 45 59 85 51
85>72 , shift it to the right
12 45 59 72 85 51
59<72, so insert 72 in that place
12 45 59 72 85
85>51 , shift it to the right
12 45 59 72 85
72>51 , shift it to the right
12 45 59 72 85
59>51 , shift it to the right
12 45 51 59 72 85
45<51, so insert 51 in that place
© w3resource.com

# SELECTION SORT

- Metode ini mulai dengan elemen pertama dan mencari pada seluruh array nilai yang terkecil
- Jika ada yang lebih kecil dari elemen pertama, akan ditukar
- Putaran kedua, akan dimulai dari elemen kedua, demikian seterusnya.
- Variabel `i` menyatakan tempat dimana elemen terkecil ditempatkan.
- Variabel `t` menyatakan elemen terkecil
- Data di dalam larik akan berubah-ubah

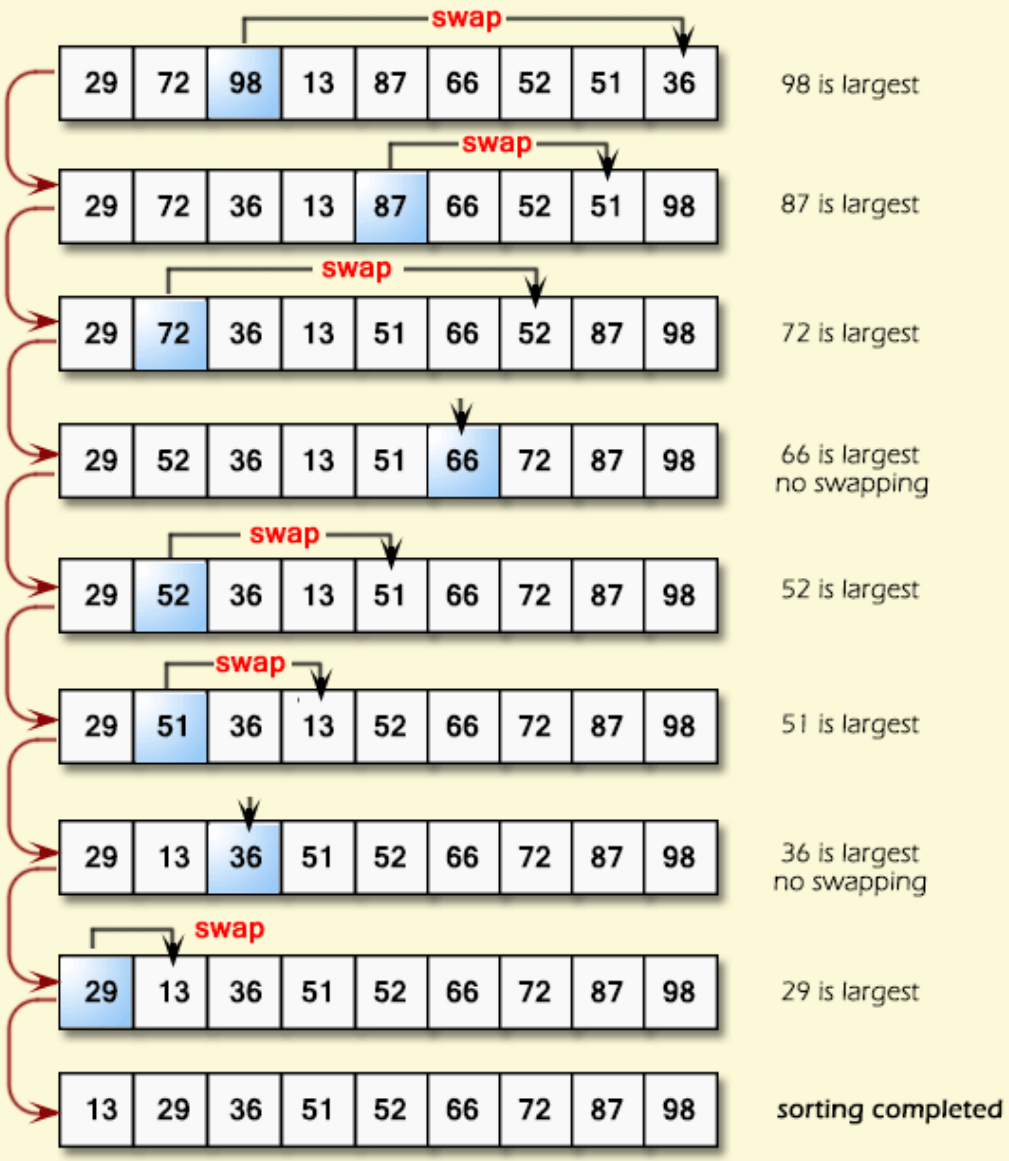
# SELECTION SORT ALGORITHM




```
void selectionSort(int arr[])
{
   for(int i=0;i<4;i++)
   {
      int min = i;

      for(int j=i+1;j<5;j++)
      {
         if(arr[j]<arr[min])
         {
            min=j;
         }
      }
      if(min!=i)
      {
         int temp=arr[min];
         arr[min] = arr[i];
         arr[i] = temp;
      }
   }
}
```

# Selection Sort



----Implementasi Algoritma Bubble Sort

```
private static void bubbleSort(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]){
                    temp = arr[j-1];
                    arr[j-1] = arr[j];
                    arr[j] = temp;
                }
            }
        }
    }
```

----Implementasi Algoritma Selection Sort

```
private static void selectionSort(int[] bilangan){
        for (int i = 0; i < bilangan.length - 1; i++){
            int index = i;
            for (int j = i + 1; j < bilangan.length; j++){
                if (bilangan[j] < bilangan[index]){
                    index = j;
                }
            }
            int smallerNumber = bilangan[index];
            bilangan[index] = bilangan[i];
            bilangan[i] = smallerNumber;
        }
    }
```

----Implementasi Algoritma Insertion Sort

```
private static void insertionSort(int bilangan[]) {
        int n = bilangan.length;
        for (int j = 1; j < n; j++) {
            int key = bilangan[j];
            int i = j-1;
            while ( (i > -1) && ( bilangan [i] > key ) ) {
                bilangan [i+1] = bilangan [i];
                i--;
            }
            bilangan[i+1] = key;
        }
    }
```

# CONTOH PROGRAM

```
2   package SortingALG;
3   public class BubbleSort {
4       private final int [] data = {5, 30, 12, 15, 27, 13, 27, 48, 42, 54, 24, 58, 90};
5           public void tampilData(){
6           for (int i : data) {
7               System.out.print(i+" ");
8           }
9           System.out.println();  //pindah baris
10      }
11
12      public void bubbleSort() {
13          int n = data.length;
            int temp = 0;
15          for(int i=0; i < n; i++){
16              for(int j=1; j < (n-i); j++){
17                  if(data[j-1] > data[j]){
18                      temp = data[j-1];
19                      data[j-1] = data[j];
20                      data[j] = temp;
21                  }
22          } } }
23  }
```

```java

package SortingALG;
/**
 * @author dwrat
 */
public class TesSorting {

    public static void main(String[] args) {
        BubbleSort obj = new BubbleSort();
        obj.tampilData();     //menampilkan data awal
        obj.bubbleSort();
        System.out.println("Sesudah Dilakukan Sorting");
        obj.tampilData();
    }
}
```

# SELESAI