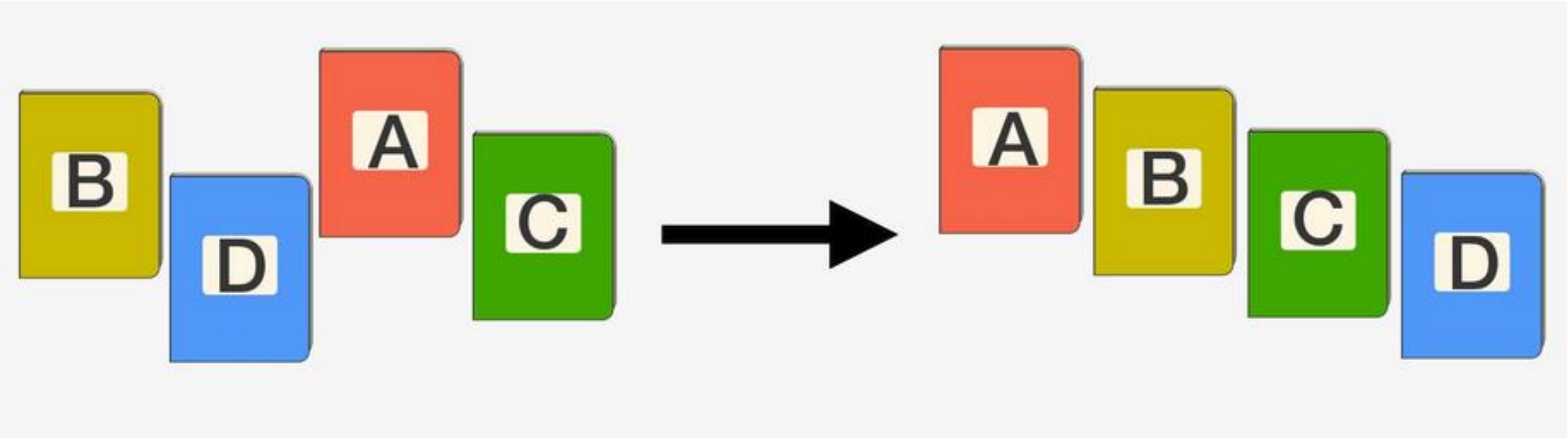
ALGORITMA SORTING

DR. DWI RATNA S., MT



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 Pemrosesan Data

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.

Contoh : 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.

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

4. Merge Sort

Dalam algoritma ini kumpulan data dibagi menjadi subkumpulan, kemudian subkumpulan tersebut diurutkan secara terpisah, dan kemudian digabungkan kembali.

Contohnya adalah : Merge sort.

5. Non-Comparison Sort

Proses pengurutan data pada algoritma ini tidak terdapat pembandingan antardata, data diurutkan sesuai dengan **pigeon hole principle.**

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

TIGA ALGORITMA PENGURUTAN BERBASIS-PEMBANDINGAN

Bubble Sort

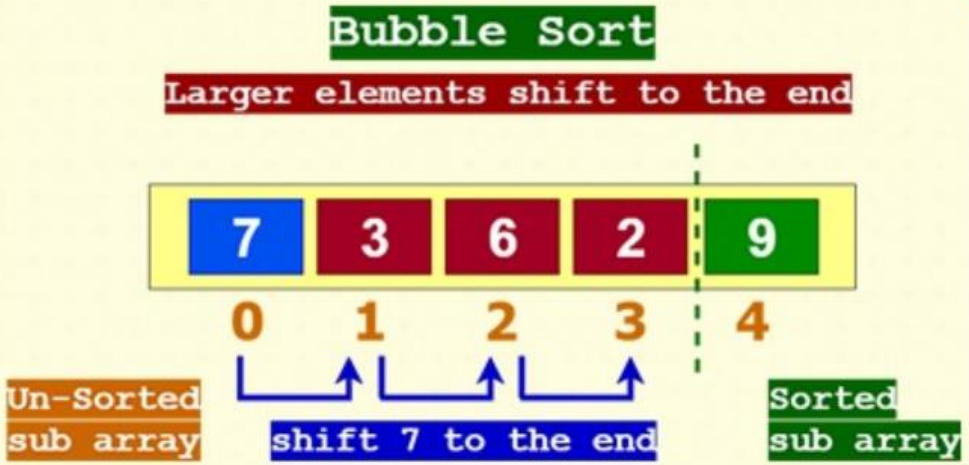
Insertion Sort

Selection Sort

BUBBLE SORT

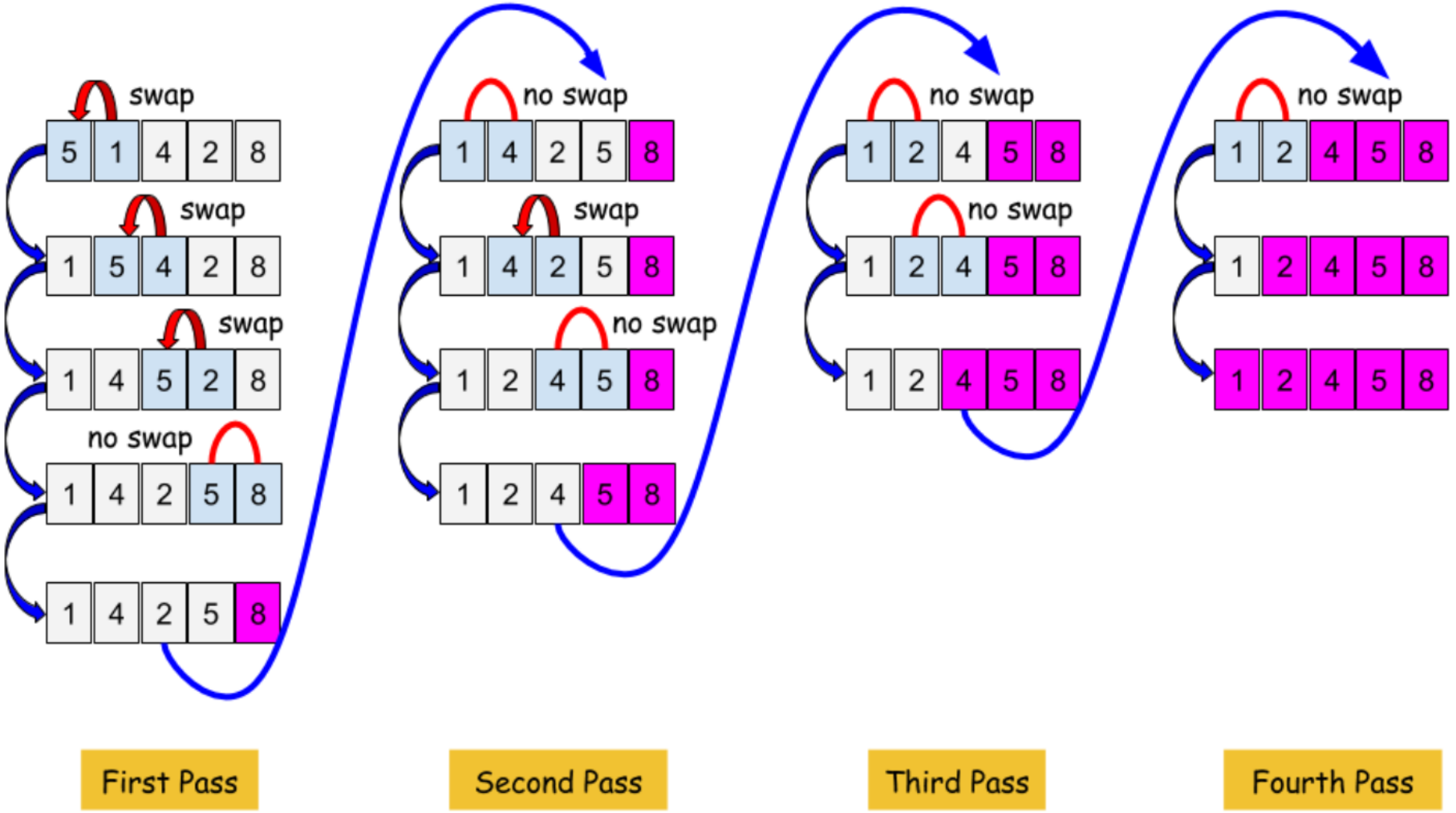
- Bubble Sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya.
- Jika elemen sekarang lebih besar dari elemen berikutnya maka kedua elemen tersebut ditukar, jika pengurutan ascending.
- Jika elemen sekarang lebih kecil dari elemen berikutnya, maka kedua elemen tersebut 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


swap
5 1 4 2 8
swap
1 5 4 2 8
swap
1 4 5 2 8
no swap
1 4 2 5 8
1 4 2 5 8
no swap
1 4 2 5 8
swap
1 4 2 5 8
no swap
1 2 4 5 8
1 2 4 5 8
no swap
1 2 4 5 8
no swap
1 2 4 5 8
1 2 4 5 8
no swap
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
First Pass
Second Pass
Third Pass
Fourth Pass

ALGORITHM *BubbleSort*$(A[0..n-1])$

//Sorts a given array by bubble sort
//Input: An array $A[0..n-1]$ of orderable elements
//Output: Array $A[0..n-1]$ sorted in nondecreasing order
for $i \leftarrow 0$ **to** $n-2$ **do**
    for $j \leftarrow 0$ **to** $n-2-i$ **do**
        if $A[j+1] < A[j]$ swap $A[j]$ and $A[j+1]$

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1 = \sum_{i=0}^{n-2} [(n-2-i)-0+1]$$
$$= \sum_{i=0}^{n-2} (n-1-i) = \frac{(n-1)n}{2},$$

$$S_{worst}(n) = C(n) = \frac{(n-1)n}{2}.$$

KOMPLEKSITAS

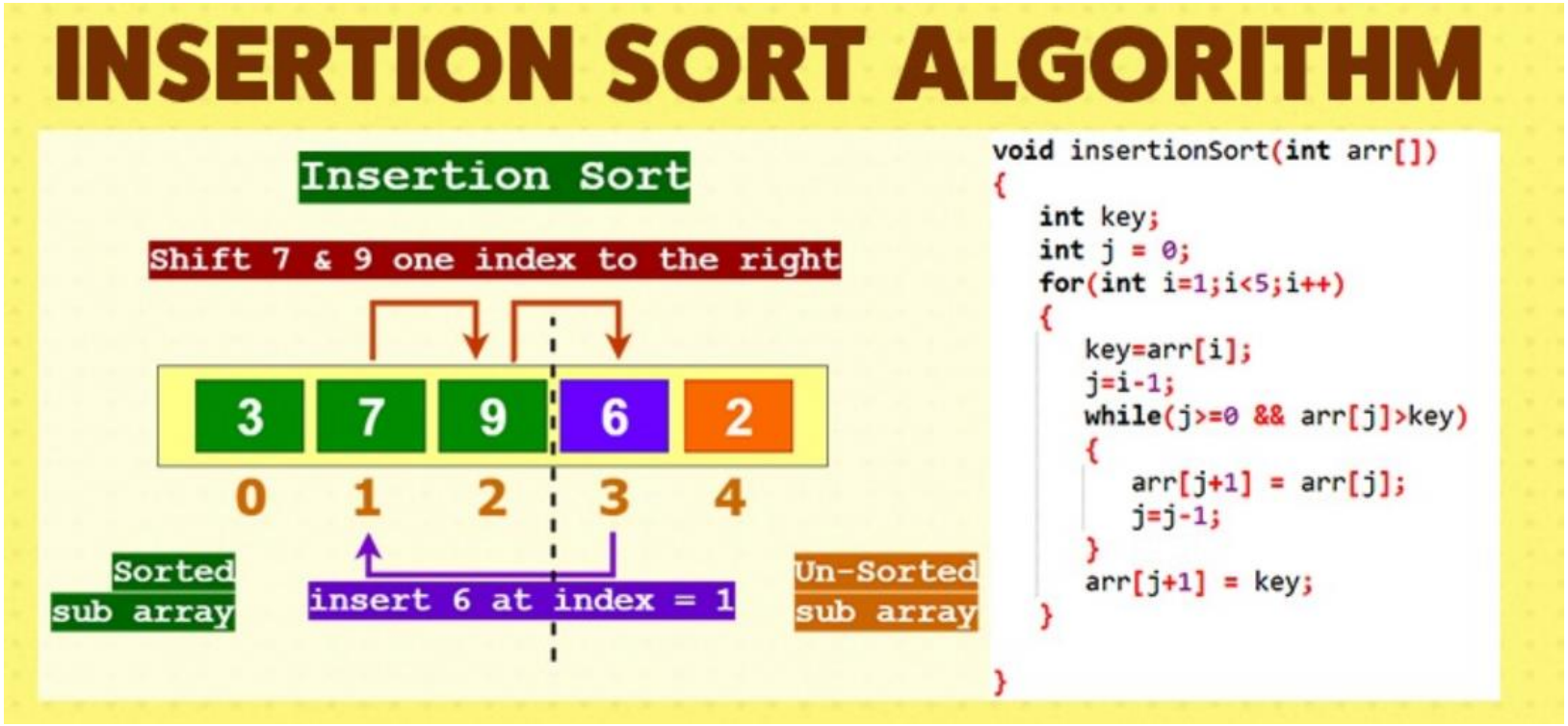
```
package Sorting;
public class BubbleSort {

    public static void main(String[] args) {
        int[] data = {5, 30, 12, 15, 27, 13, 27, 48, 42, 54, 24, 58, 90};
        System.out.println("Data sebelum diurutkan");
        tampilData(data);

        bubbleSort(data);

        System.out.println("Data sebelum diurutkan");
        tampilData(data);
    }
    public 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;
                }
            }
    }      }

    public static void tampilData(int[] arr){
        int n = arr.length;
        for (int i=0; i < n; i++) {
            System.out.print(arr[i]+" ");
        }
        System.out.println();  //pindah baris
    }
```

PROGRAM

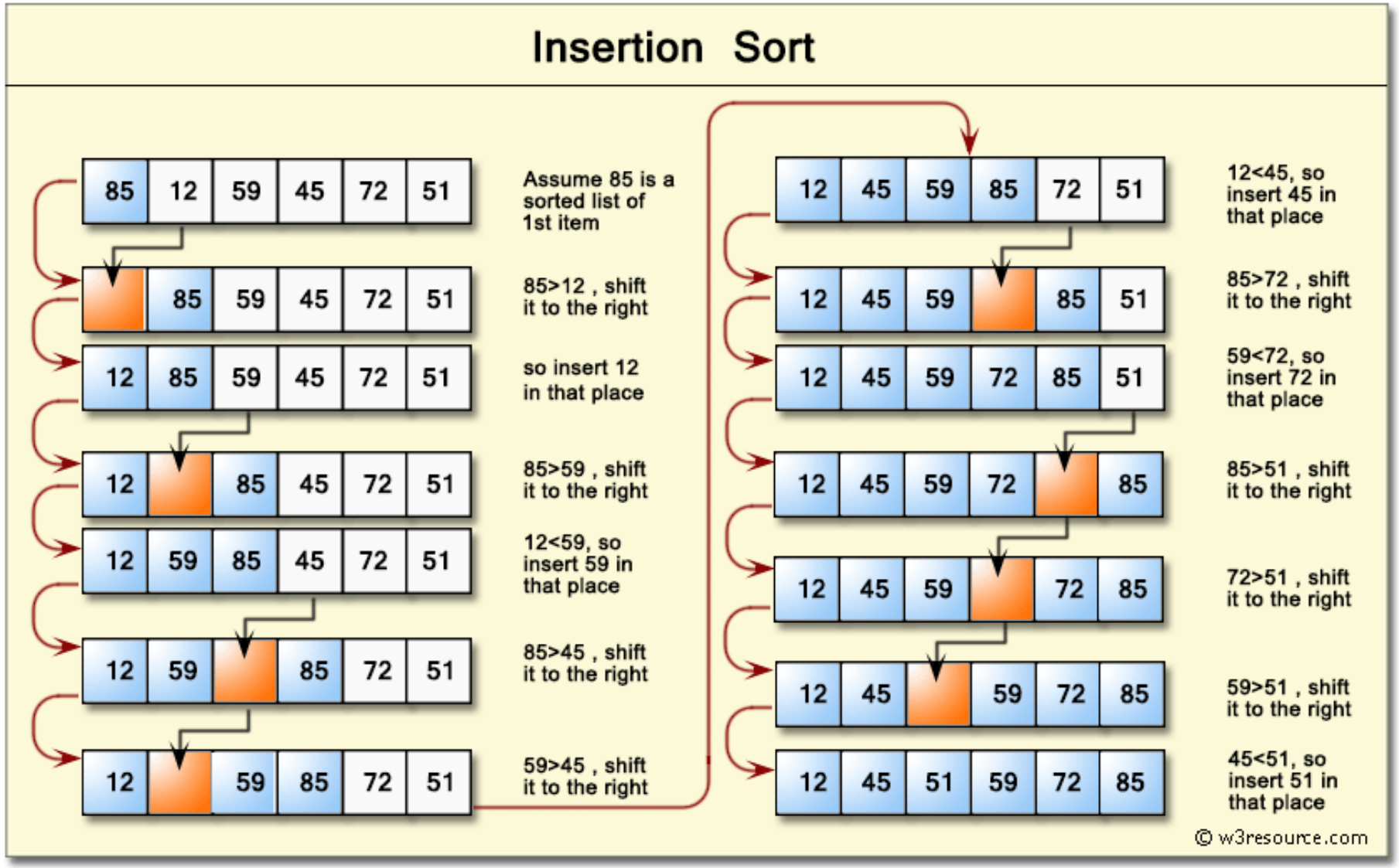
INSERTION SORT

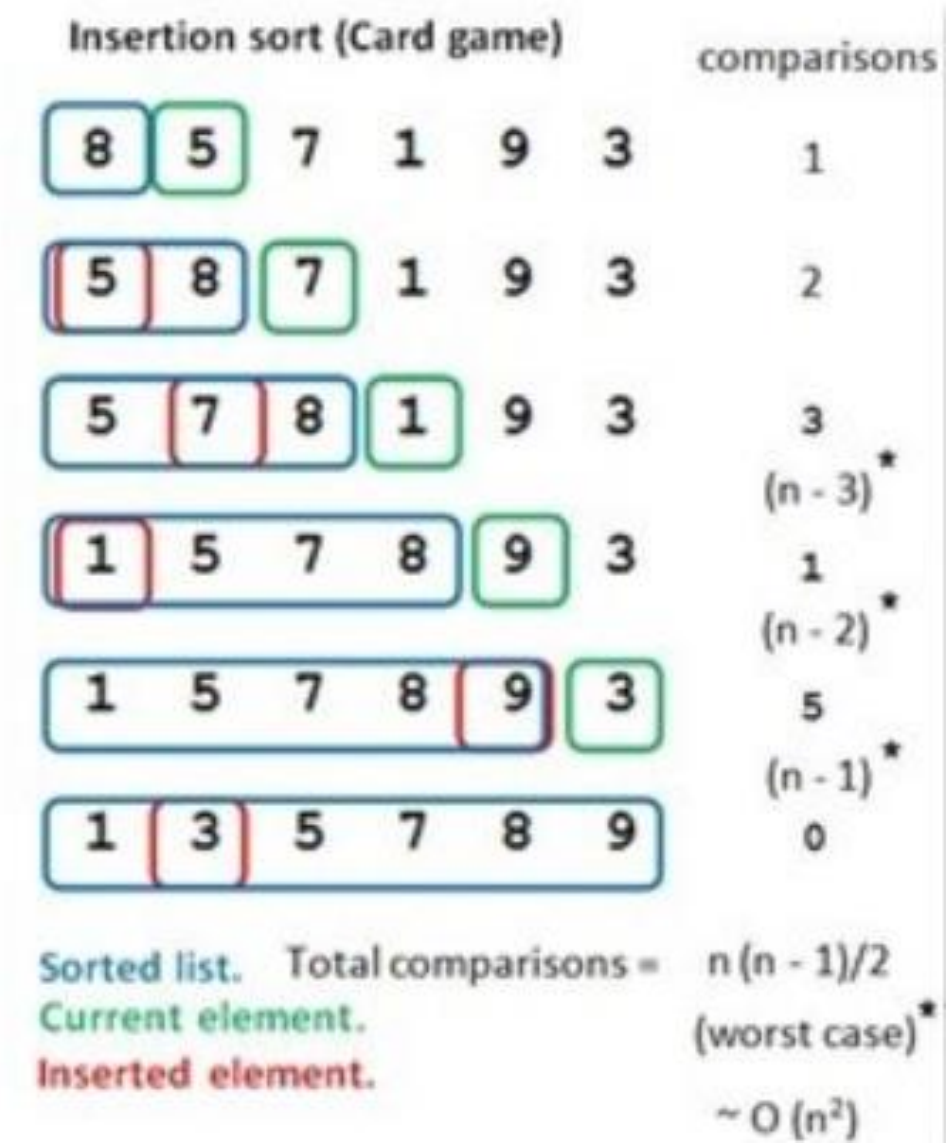
- Algoritma *insertion sort* pada dasarnya memilah data yang akan diurutkan menjadi dua bagian, yang belum diurutkan(meja pertama) dan yang sudah diurutkan(meja kedua)
- Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkan sesuai posisinya pada bagian lain dari array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


Insertion sort (Card game)
comparisons
8 5 7 1 9 3
1
5 8 7 1 9 3
2
5 7 8 1 9 3
3
(n - 3)*
1 5 7 8 9 3
1
(n - 2)*
1 5 7 8 9 3
5
(n - 1)*
1 3 5 7 8 9
0
Sorted list.
Total comparisons =
n (n - 1)/2
Current element.
(worst case)*
Inserted element.
~ O (n²)

ALGORITHM *InsertionSort*$(A[0..n-1])$

//Sorts a given array by insertion sort
//Input: An array $A[0..n-1]$ of n orderable elements
//Output: Array $A[0..n-1]$ sorted in nondecreasing order
for $i \leftarrow 1$ **to** $n-1$ **do**
    $v \leftarrow A[i]$
    $j \leftarrow i-1$
    while $j \geq 0$ **and** $A[j] > v$ **do**
        $A[j+1] \leftarrow A[j]$
        $j \leftarrow j-1$
    $A[j+1] \leftarrow v$

$$C_{worst}(n) = \sum_{i=1}^{n-1} \sum_{j=0}^{i-1} 1 = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}$$

$$C_{best}(n) = \sum_{i=1}^{n-1} 1 = n-1$$

$$C_{avg}(n) \approx \frac{n^2}{4}$$

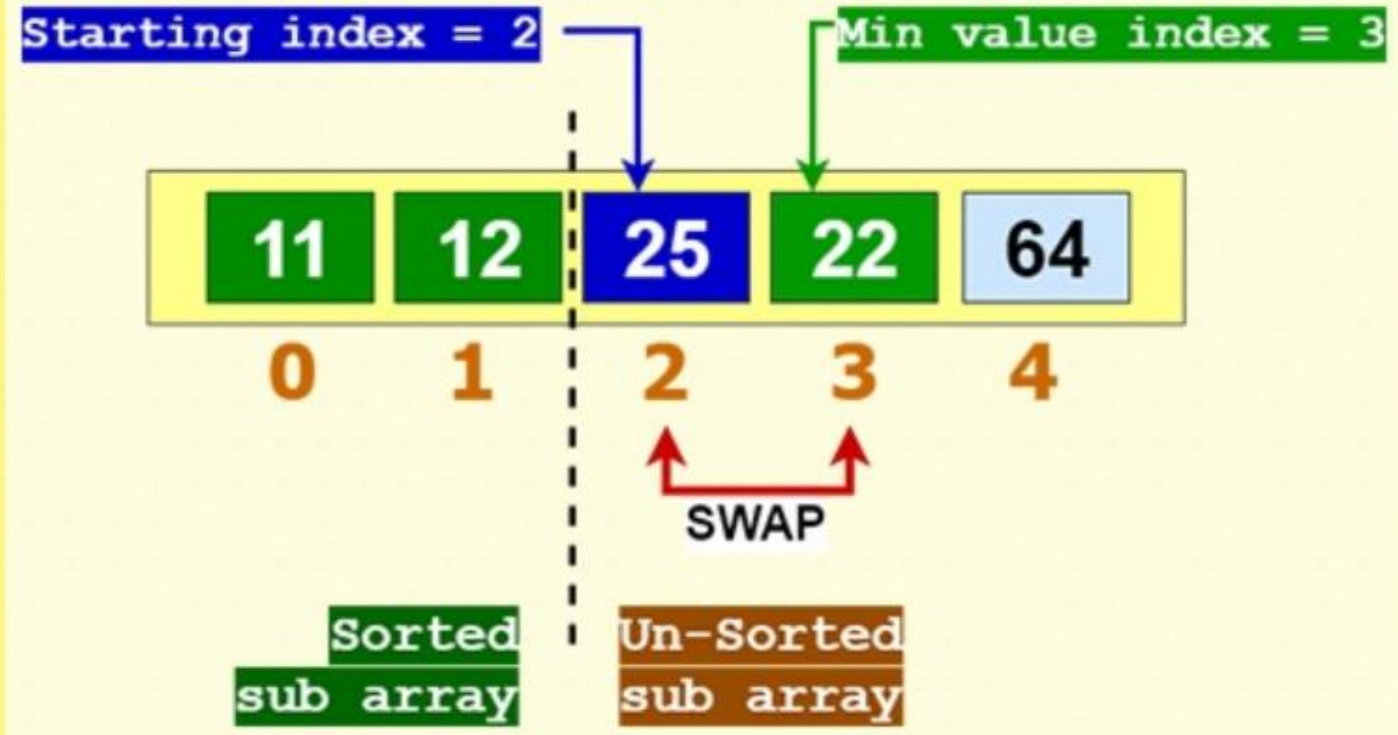
```
package Sorting;
public class InsertionSort {
    public static void main(String[] args) {
        int[] data = {5, 30, 12, 15, 27, 13, 37, 48, 42, 54, 24, 58, 90};
        System.out.println("Data sebelum diurutkan");
        tampilData(data);

        InsertSort(data);

        System.out.println("Data sesudah diurutkan");
        tampilData(data);
    }

    public static void InsertSort(int[] arr) {
        int n = arr.length;
        int v, j;
        for(int i=1; i < n-1; i++){
            v = arr[i];
            j = i-1;
            while (j>0 && arr[j]>v) {
                    arr[j+1] = arr[j];
                    j = j-1;
                }
            arr[j+1]=v;
        }
    }

    public static void tampilData(int[] arr){
        int n = arr.length;
        for (int i=0; i < n; i++) {
            System.out.print(arr[i]+" "); }
        System.out.println();  //pindah baris
    }
}
```

PROGRAM

SELECTION SORT

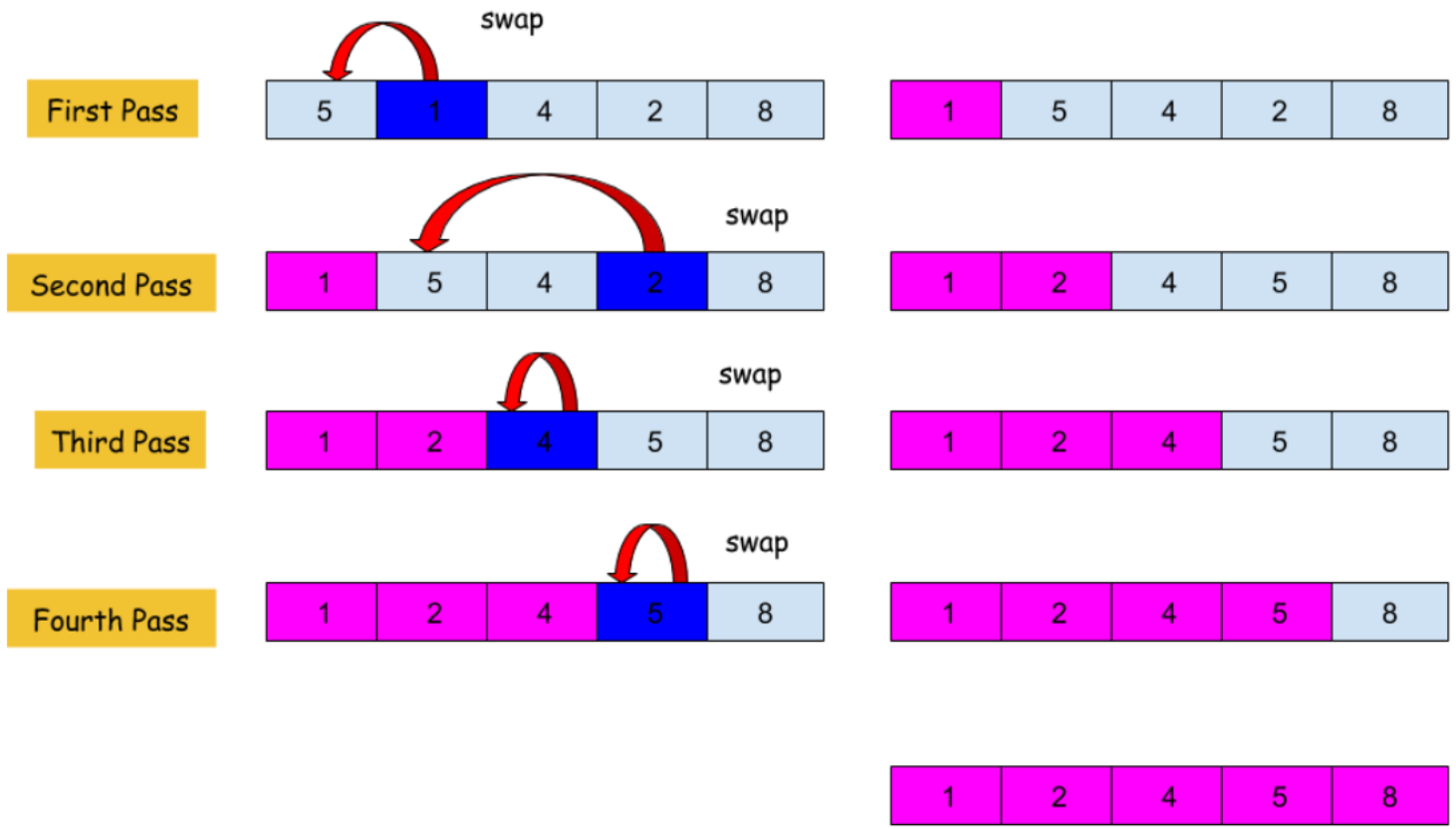
- Metode ini mulai dengan elemen pertama dan mencari pada seluruh array nilai yang terkecil
- Jika ada yang lebih kecil dari elemen pertama, akan ditukar
- Putaran kedua, akan dimulai dari elemen kedua, demikian seterusnya.
- Variabel `i` menyatakan tempat dimana elemen terkecil ditempatkan.
- Variabel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

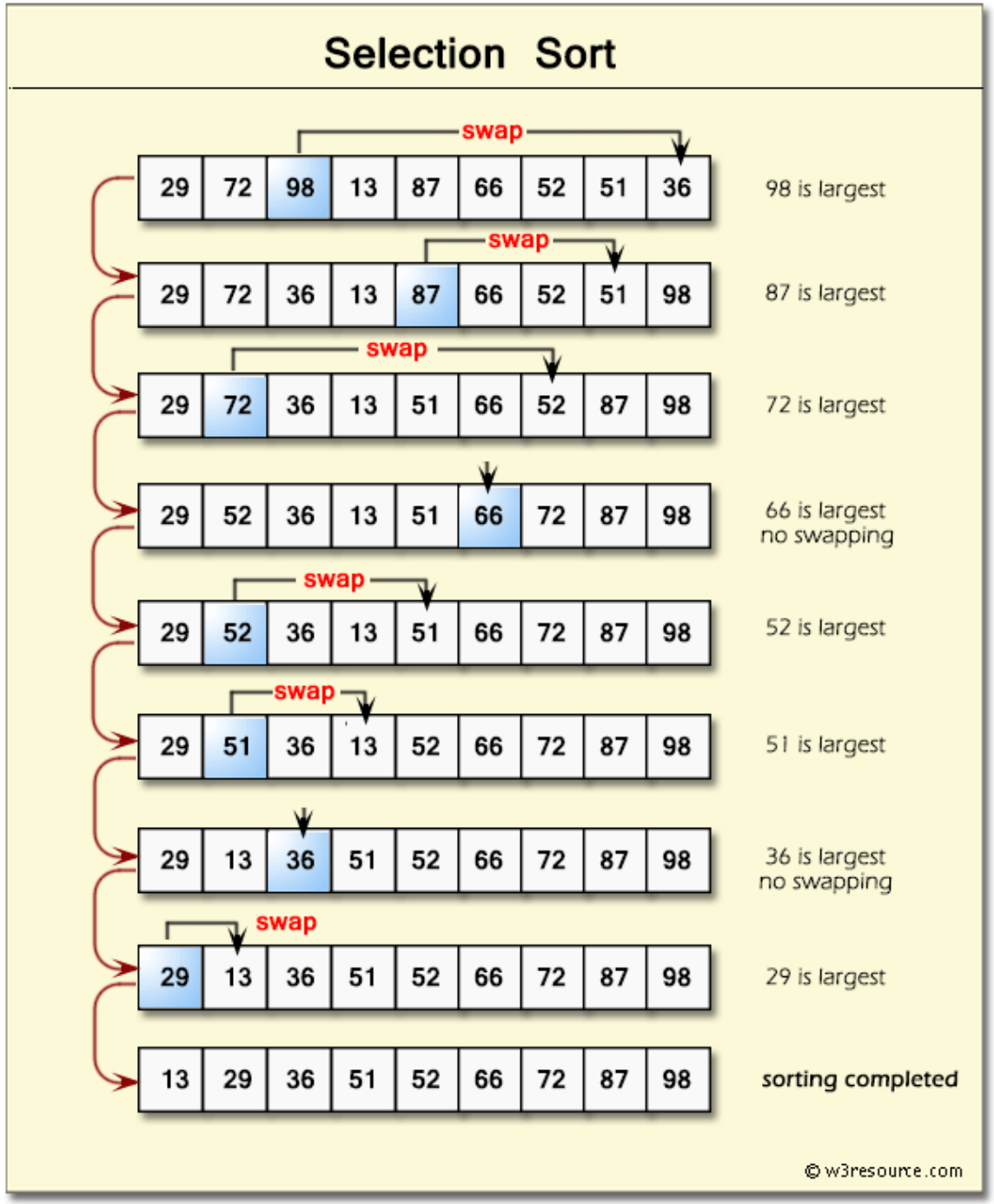
SELECTION SORT ALGORITHM




```
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;
      }
   }
}
```

CONTOH TERURUT NAIK




Selection Sort
swap
29 72 98 13 87 66 52 51 36
98 is largest
swap
29 72 36 13 87 66 52 51 98
87 is largest
swap
29 72 36 13 51 66 52 87 98
72 is largest
29 52 36 13 51 66 72 87 98
66 is largest
no swapping
swap
29 52 36 13 51 66 72 87 98
52 is largest
swap
29 51 36 13 52 66 72 87 98
51 is largest
29 13 36 51 52 66 72 87 98
36 is largest
no swapping
swap
29 13 36 51 52 66 72 87 98
29 is largest
13 29 36 51 52 66 72 87 98
sorting completed
© w3resource.com

ALGORITMA DAN KOMPLEKSITAS

ALGORITHM *SelectionSort*$(A[0..n-1])$

//Sorts a given array by selection sort
//Input: An array $A[0..n-1]$ of orderable elements
//Output: Array $A[0..n-1]$ sorted in nondecreasing order
for $i \leftarrow 0$ **to** $n-2$ **do**
    $min \leftarrow i$
    for $j \leftarrow i+1$ **to** $n-1$ **do**
        if $A[j] < A[min]$ $min \leftarrow j$
    swap $A[i]$ and $A[min]$

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1] = \sum_{i=0}^{n-2} (n-1-i).$$


THE
END