



# **STRUKTUR DATA** **SORTING (1)**



# ***Pengantar pengurutan data***



# Pengurutan Data

- Pengurutan data merupakan salah satu proses yang banyak ditemui dalam aplikasi menggunakan komputer.
- Pengurutan membuat data disusun dalam keadaan urut menurut aturan  
=> **Misal dari nilai terkecil menuju nilai terbesar untuk bilangan A ke Z pada string.**
- Sebagai contoh, data yang urut memudahkan dalam pencarian, praktik ini sering dijumpai menggunakan pengurutan data.



**contoh :**

- Yellow pages : memiliki informasi yang telah di urutkan menurut nama perusahaan / perorangan
- Kamus berisi kata-kata yang diurutkan berdasarkan huruf terkecil ke terbesar
- Laporan penjualan disusun berdasarkan produk terlaris hingga paling banyak terjual
- File-file dalam directory ditampilkan urut berdasarkan (nama, tanggal dll)
- Indeks buku berisi daftar istilah yang memudahkan pembaca mendapat lokasi halaman yang berisi istilah
- Glossary dalam buku teks berisi istilah dan definisi, nama yang disusun urut.



# Pengurutan Data

Ada 2 teknik untuk melakukan pengurutan data (sorting) yaitu *ascending* dan *descending*.

1. **Pengurutan *Ascending***: data diurutkan dari yang terkecil hingga yang terbesar (dikenal juga sebagai urutan nilai naik).

Contoh:

1a)

700	400	900	600	300
-----	-----	-----	-----	-----



Data acak

Data ditampilkan secara horisontal (diurutkan dari kiri ke kanan)

300	400	600	700	900
-----	-----	-----	-----	-----

Data terurut secara ***Ascending***



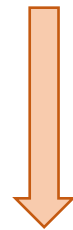
# Pengurutan Data

Ada 2 teknik untuk melakukan pengurutan data (sorting) yaitu *ascending* dan *descending*.

1. **Pengurutan *Ascending***: data diurutkan dari yang terkecil hingga yang terbesar (dikenal juga sebagai urutan nilai naik).

Contoh: 1b)

300
400
600
700
900



Data ditampilkan secara vertikal  
(diurutkan dari atas ke bawah)



# Pengurutan Data

Ada 2 teknik untuk melakukan pengurutan data (sorting) yaitu *ascending* dan *descending*.

2. Pengurutan *Descending*: data diurutkan dari yang terbesar hingga yang terkecil (dikenal juga sebagai urutan nilai turun).

Contoh :

2A.

700	400	900	600	300
-----	-----	-----	-----	-----

2B.



Data acak


Data ditampilkan secara horisontal (diurutkan dari kiri ke kanan)

900	700	600	400	300
-----	-----	-----	-----	-----



Data terurut secara ***Descending***

900
700
600
400
300





# Pengurutan Data

Ada 2 teknik untuk melakukan pengurutan data (sorting) yaitu *ascending* dan *descending*.

Contoh:

**Data yang tidak urut**

Bekasi
Karawang
Cikampek
Purwakarta
Jakarta

**Data teurut ascending**

Bekasi
Cikampek
Jakarta
Karawang
Purwakarta

**Data teurut descending**

Purwakarta
Karawang
Jakarta
Cikampek
Bekasi





# Teknik Pengurutan Data

Ada beberapa cara yang digunakan untuk mengurutkan data, ada beberapa teknik yang terkenal.

No.	Sorting Dasar
1	Bubble Sort
2	Insertion Sort
3	Selection Sort

Tiga yang Pertama : dikenal sebagai cara pengurutan dasar

No.	Sorting Lanjutan
1	Merge Sort
2	Quick Sort
3	Bucket Sort
4	Shell Sort
5	Radix Sort
6	External Sort



# Teknik Pengurutan Data

- Secara umum masing- masing Teknik pengurutan data memiliki keunggulan dan kelemahan, Teknik mana yang yang paling baik sulit dikatakan.
- Ada beberapa cara dalam memilih Teknik, yaitu jumlah data, kompleksitas waktu, dan lama waktu membuat program

No.	Teknik pengurutan	<i>Best case</i>	<i>Average case</i>	<i>worst case</i>
1	<i>Bubble Sort</i>	$O(N)$	$O(N^2)$	$O(N^2)$
2	<i>Insertion Sort</i>	$O(N)$	$O(N^2)$	$O(N^2)$
3	<i>Selection Sort</i>	$O(N)$	$O(N^2)$	$O(N^2)$
4	<i>Merge Sort</i>		$O(N \log N)$	$O(N \log N)$
5	<i>Quick Sort</i>	$O(N)$	$O(N \log N)$	$O(N^2)$
6	<i>Shell Sort</i>		$O(N^{3/2})$	$O(N^2)$



# 1. Metode Bubble Sort

- Bubble Sort merupakan metode pengurutan data yang membandingkan setiap elemen data dengan seluruh elemen di dekatnya dan melakukan penukaran jika memenuhi kriteria
- Metode sorting merupakan metode paling mudah dalam pengurutan data, namun prosesnya paling lambat jika dibandingkan dengan metode sorting lainnya.
- Metode Bubble Sort dengan sifat sangat cepat untuk menangani data yang berjumlah sedikit dan menyita waktu lama untuk data yang sangat banyak.



# 1. Metode Bubble Sort

- Sebagai contoh, lihat pada data berikut :

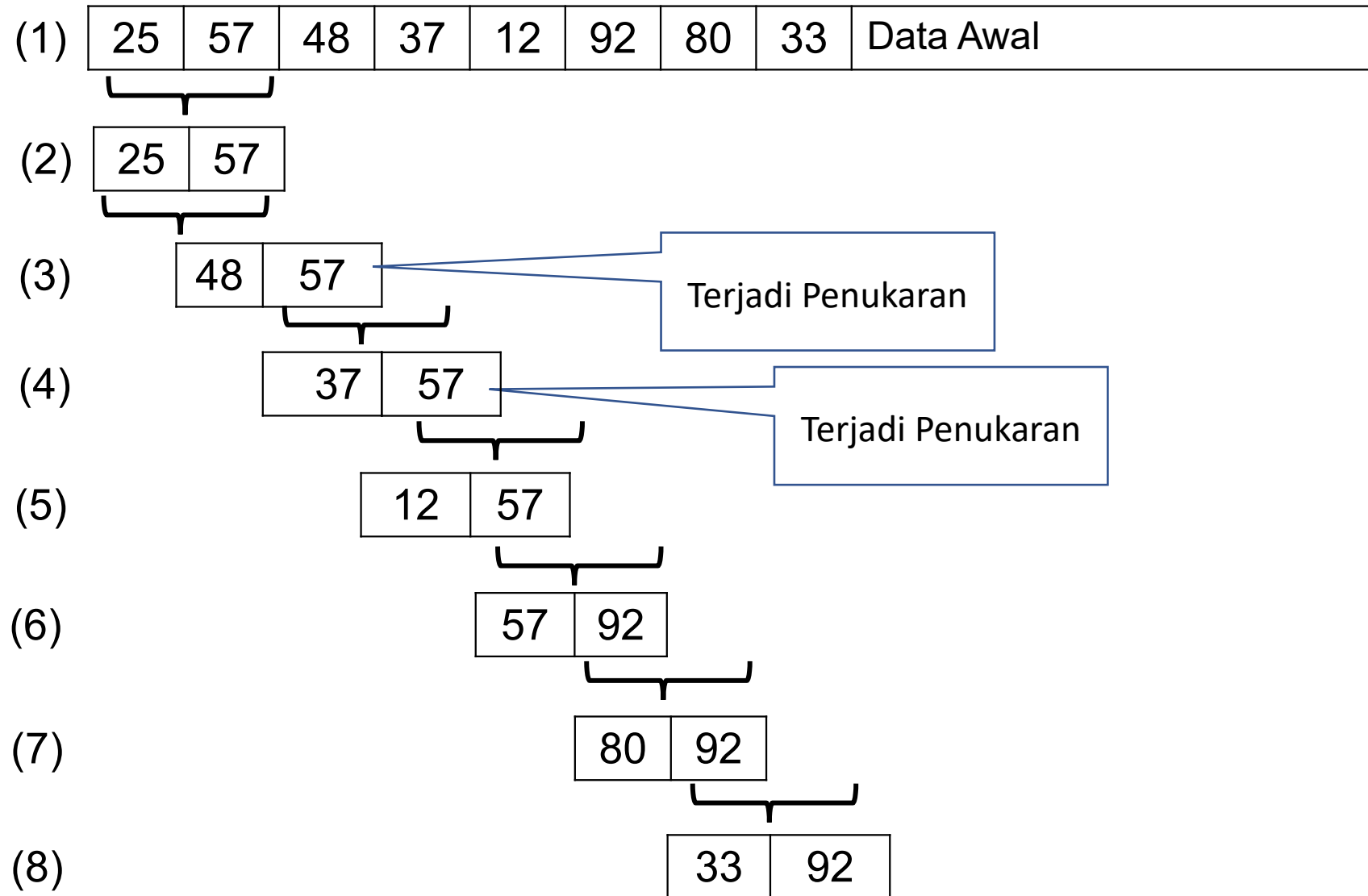
25	57	48	37	12	92	80	33	Data Awal
----	----	----	----	----	----	----	----	-----------

Pengurutan naik Ascending: jika elemen sekarang lebih besar dari pada elemen berikutnya, maka kedua elemen tersebut ditukar tempatnya; tetapi jika elemen sekarang lebih kecil dari pada elemen berikutnya, maka tidak ditukar (tetap).

Amati pengurutan tahap pertama, berikut :

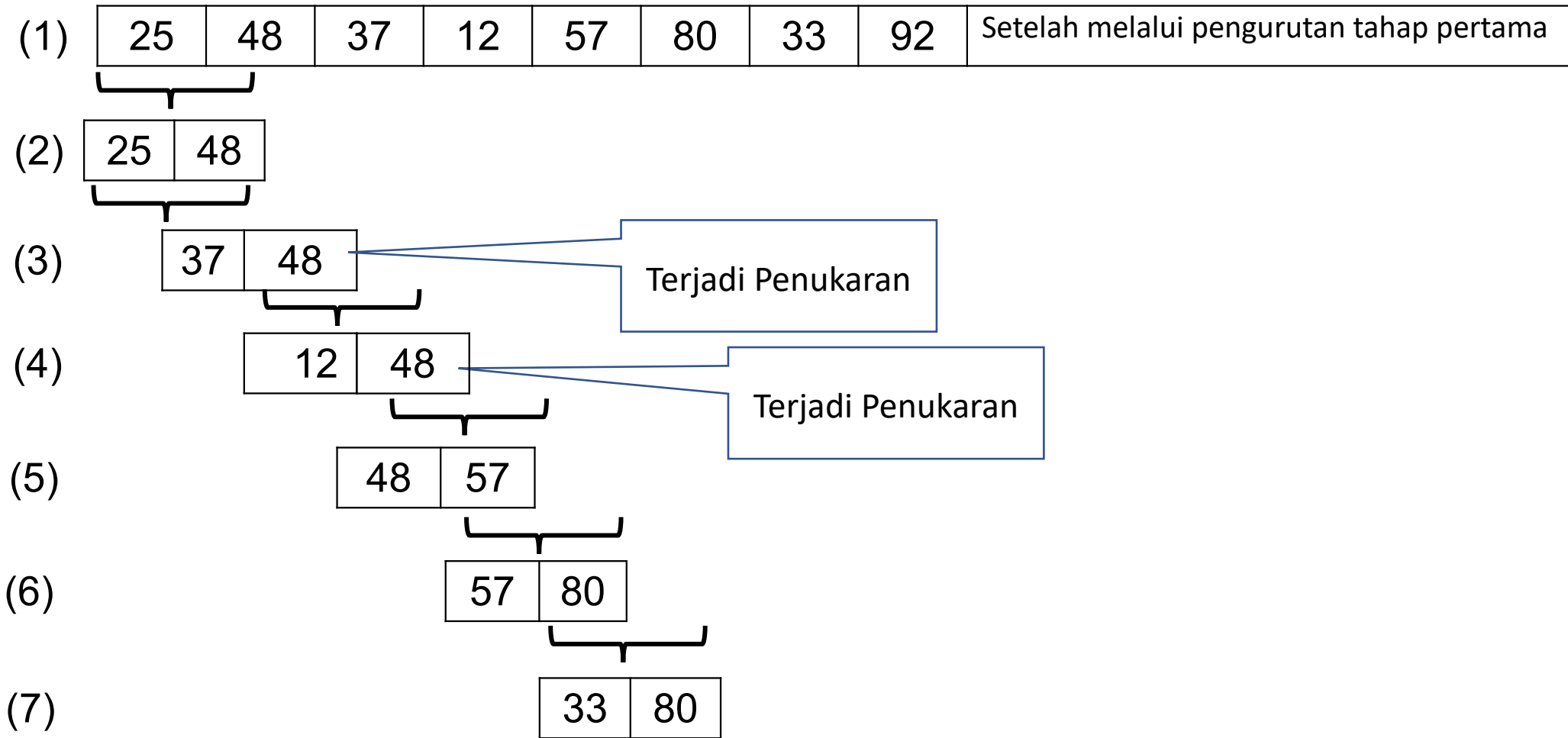


# 1. Metode Bubble Sort





# 1. Metode Bubble Sort



Hasil pengurutan tahap kedua



# 1. Metode Bubble Sort

- Setelah melalui tahap kedua, sebagai contoh, susunan data menjadi berikut :

25	37	12	48	57	33	80	92	Data hasil
----	----	----	----	----	----	----	----	------------

- Jika data sebesar  $n$ , terjadi  $n-1$  tahap pengurutan, lihat keadaan awal hingga akhir

25	57	48	37	12	92	80	33	Awal
25	48	37	12	57	80	33	92	Tahap 1
25	37	12	48	57	33	80	92	Tahap 2
25	12	37	48	33	57	80	92	Tahap 3
12	25	37	33	48	57	80	92	Tahap 4
12	25	33	37	48	57	80	92	Tahap 5
12	25	33	37	48	57	80	92	Tahap 6
12	25	33	37	48	57	80	92	Tahap 7



# 1. Algoritma Bubble Sort

- Algoritma Bubble Sort seperti berikut :

```
Bubble Sort(L, N):  
//L adalah (Larik) array dengan N data (0....N-1)  
For Tahap  $\leftarrow$  1 TO N-1  
  For j  $\leftarrow$  0 TO N – Tahap – 1  
    if  $L[J] > L[J+1]$   
      // Lanjutkan Pertukaran  
      Tmp  $\leftarrow$  L[J]  
      L[J]  $\leftarrow$  L[J+1]  
      L[J+1]  $\leftarrow$  Tmp  
    END-IF  
  END-FOR  
END-FOR
```

Kunci pengurutan ascending  
terletak pada ekspresi  
 $L[J] > L[J+1]$  dalam IF

Kunci pengurutan  
descending bentuk IF  
IF  $L[J] < L[J+1]$





# 1. Efisiensi Bubble Sort

- Berdasarkan Algoritma Bubble Sort, jumlah pembandingan selama pengurutan data sebanyak :  
$$(N-1) + (N-2) + (N-3) + \dots + 1 = N * (N-1) / 2$$
- Hal ini berlaku untuk *best case*, *worst case*, maupun *average case* :  
Untuk  $N = 8$ , jumlah pembandingan data sebanyak 28 kali ( $8 * 7 / 2$ ), **Best Case**
- Kompleksitas waktu Bubble Sort berupa  $O(N * (N-1) / 2)$  atau  $O(N^2)$



## 2. Insertion Sort

- Metode Insertion Sort: mengurutkan seleksi data dengan cara melakukan pengurutan dengan menyisipkan data yang belum urut ke dalam bagian data yang telah diurutkan
- Konsep ini biasa dilakukan oleh pemain kartu, pemain selalu berusaha untuk membuat kartu dalam keadaan terurut, jika pemain dapat kartu baru, kartu akan disisipkan pada lokasi kartu tetap urut.





## 2. Insertion Sort

Sebagai contoh terdapat data sebagai berikut :

**25, 57, 48, 37, 12**

**Pada tahap pertama**, dua elemen dijadikan sebagai data terurut, Elemen kedua akan disisipkan terhadap elemen pertama.

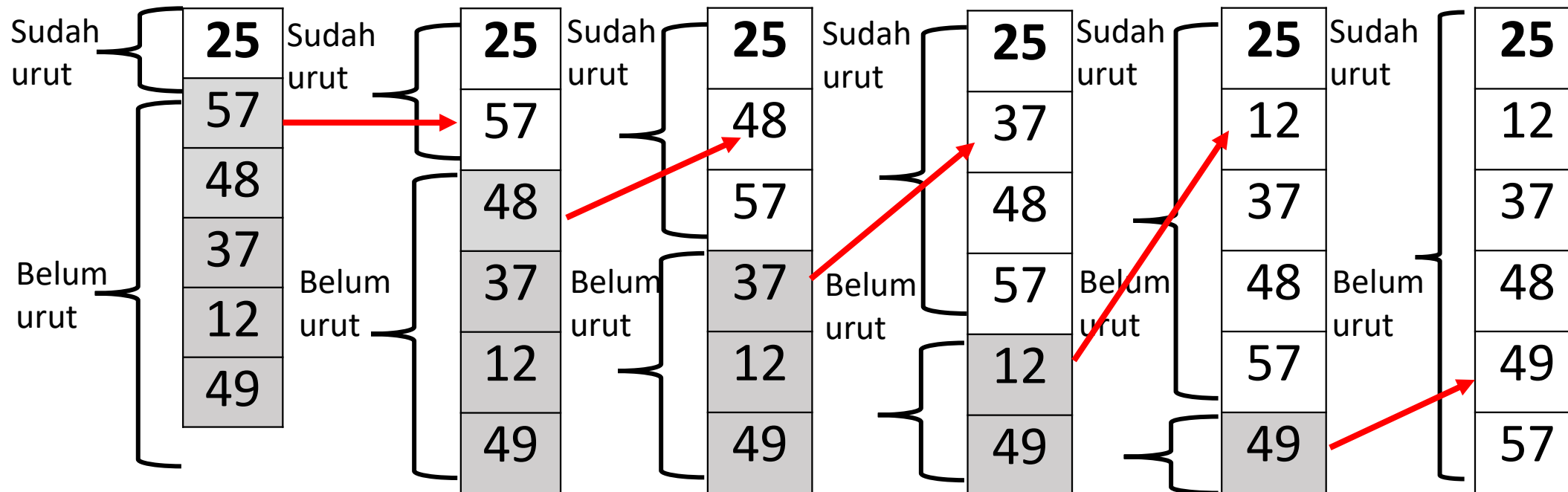
**Pada tahap kedua, tiga** elemen pertama diatu supaya urut, elemen ketiga akan disisipkan terhadap kedua elemen pertama.

**Pada tahap ketiga**, empat elemen pertama diurutkan,dengan elemen keempat akan disisipkan terhadap ketiga elemen pertama, Langkah ini dilakukan terus-menerus sampai semua elemen terurut



## 2. Insertion Sort

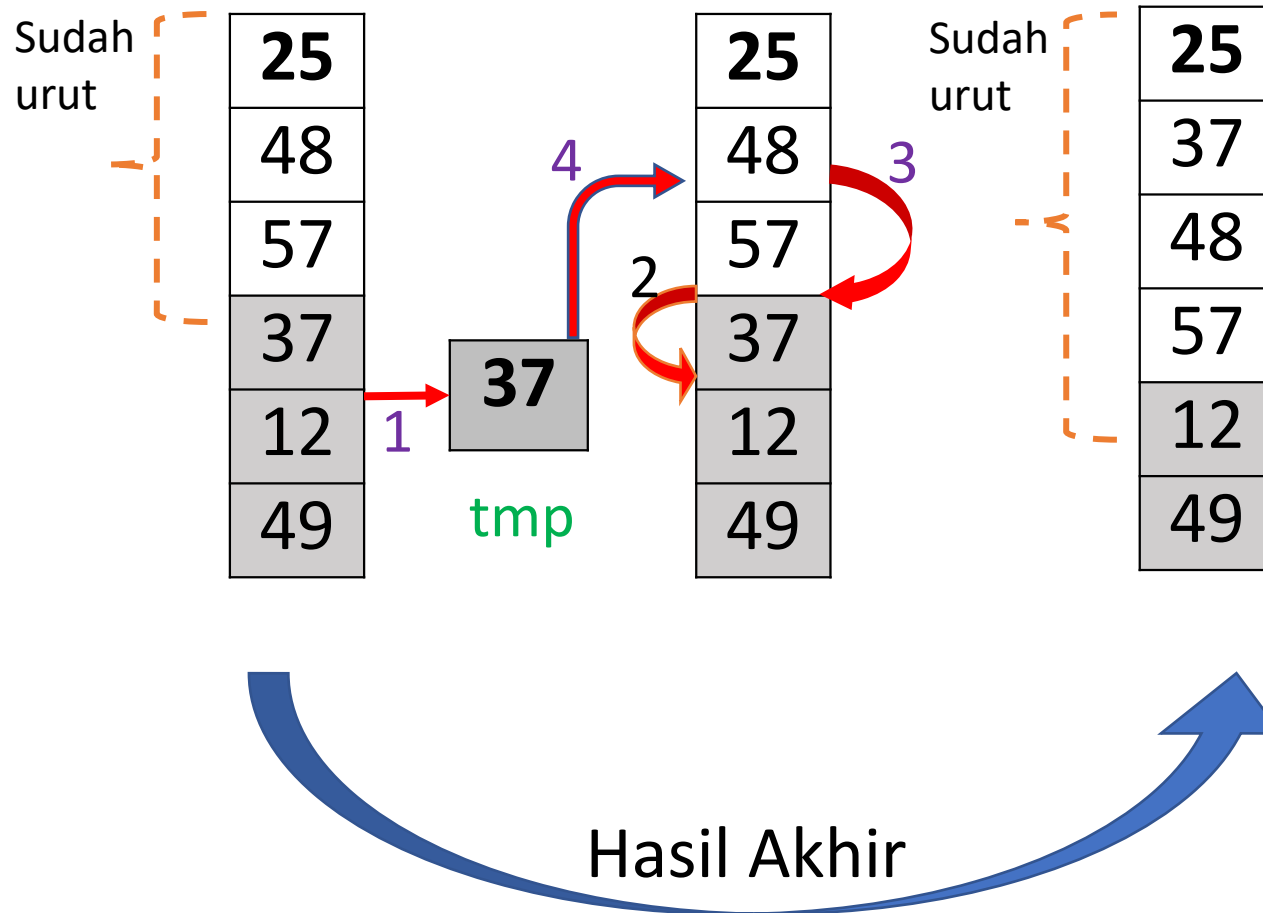
Proses pengurutan sebagai berikut :





## 2. Insertion Sort

Contoh Proses pergeseran dan penyisipan dari suatu tahap berikutnya dilakukan sebagai berikut :





## 2. Insertion Sort

Urutan pengurutan data dilakukan dengan cara :

1. Sebuah data akan diproses untuk diurutkan mula-mula disalin ke suatu variables sementara ( $tmp$ )
2. Nilai pada  $tmp$  akan disisipkan sesudah elemen 25, maka elemen 57 dan 48 perlu di geser satu posisi sesudah posisi semula.
3. Posisi semula berisi 48 yang asli (indeks 1) segera di tempati oleh data yang berada pada  $tmp$



## 2. Algoritma Insertion Sort

### Algoritma Insertion Sort

```
Insertion Sort(L, N):  
//L adalah (Larik) array dengan N data (0....N-1)  
//yang akan diurutkan  
  
FOR J  $\leftarrow$  1 TO N-1  
    tmp  $\leftarrow$  L[J]  
  
    //Sisipkan X kedalam array L dengan indeks 1 s/d K-1  
    K  $\leftarrow$  J  
    WHILE K > 1 AND L[K-1] > Tmp  
        L[K]  $\leftarrow$  L[K-1]  
        K = K-1  
    END- WHILE  
  
    L[K]  $\leftarrow$  Tmp  
END-FOR
```



## 2. Efisiensi Insertion Sort

- *Worst case* Berdasarkan Algoritma *Insertion Sort*, terjadi apabila data yang diurutkan sudah urut dalam keadaan terbalik Misal (5,4,3,2,1). Perbandingan akan berupa  $(N-1) + \dots + 2 + 1 = N * (N-1) / 2$   
Atau sama dengan  $N(N-1) / 2$ , dengan demikian Big O berupa  **$O(N^2)$**
- *Average case* diperkirakan terjadinya  $N(N-1) / 2$  perbandingan pertama, jumlah pembandingnya sebesar  $(N-1) / 2 + \dots + 2 / 2 + 1/2$   
 $= N(N-1)/4$ , Dengan demikian, *Average case* juga berupa  **$O(N^2)$**
- *Best case* terjadi kalua isi array sudah urut sebelum pengurutan terjadi, dengan demikian bagian luar (FOR) dilakuakn sebanyak N-1 dalam tidak dieksekusi sudah urut, best case berupa  **$O(N^2)$**





### 3. Selection Sort

- *Selection Sort* disebut juga push down sort (Tanenbaum, 1981)
- Pada pengurutan *Selection Sort* mula-mula (diberi nama `posAwal`) hal ini menunjuk ke lokasi awal pengurutan.
- Pertama kali penunjuk menunjukan ke indek 0 (awal array), kemudian dicari bilangan terkecil yang berada pada posisi dari `posAwal` hingga indek terkahir.
- Posisi elemen dengan bilangan terkecil dicatat dengan `posMin`
- Apabila nilai `posAwal` tidak sama dengan `posMin`, elemen kedua penunjuk di tukar.
- `posAwal` bernilai 1 hingga N-1 (Menyatakan jumlah data), data dalam array terurut



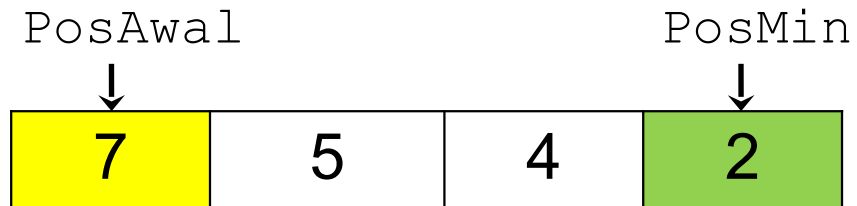
### 3. Selection Sort

Sebagai contoh terdapat data sebagai berikut :

**7, 5, 4, 2**

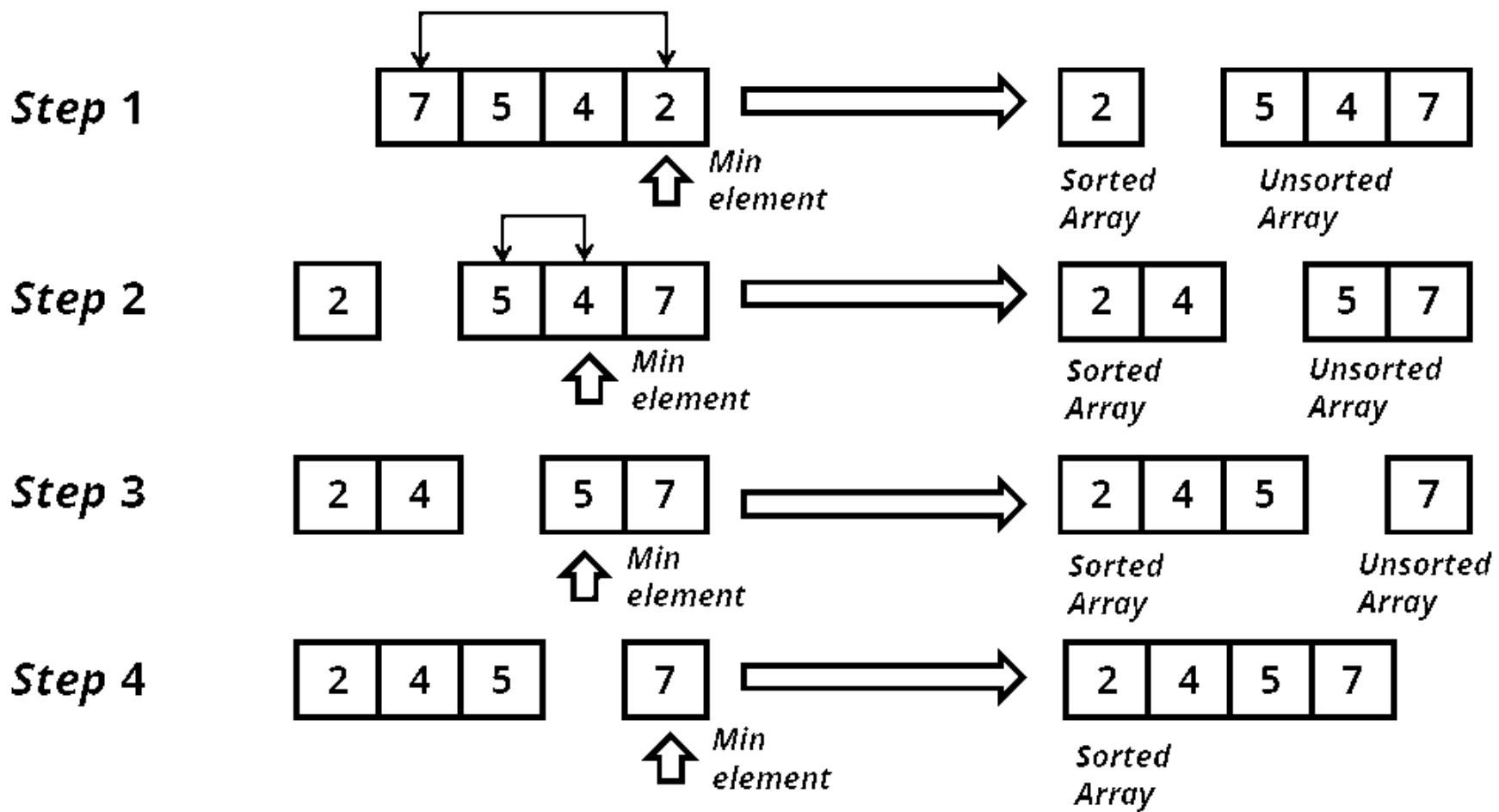
Proses pengurutnya adalah :

Contoh:





# 3. Selection Sort





## 3. Selection Sort

### Algoritma Selection Sort

**Selection Sort(L, N):**

FOR PosAwal  $\leftarrow$  0 TO N-2

PosMin  $\leftarrow$  PosAwal

//Cari data terkecil dan catat index 1 s/d K-1

FOR J  $\leftarrow$  PosAwal + 1 TO N-1

IF L [PosMin] > L[J]

PosMin  $\leftarrow$  J

END- IF

END- FOR

//Kalau PosAwal  $\neq$  PosMin maka tukarkan elemen

IF PosAwal  $\neq$  PosMin

Tmp- L [PosAwal]

L [PosAwal]  $\leftarrow$  L [PosMin]

L [PosMin]  $\leftarrow$  tmp

END- IF

END- FOR



### 3. Efisiensi selection Sort

- Kompleksitas waktu pada selection sort diukur dengan jumlah pembandingan, tahap pertama terdapat  $N-1$  pembandingan, pada tahap kedua  $N-2$  pembandingan tahap ketiga  $N-3$  Pembandingan dan seterusnya , total pembandingan
- $(N-1) + (N-2) + \dots + 2 + 1 = N(N-1) / 2$   
oleh karena itu, kompleksitas waktu berupa  **$O(N^2)$**
- *Worst case* diperoleh Ketika data dalam keadaan terurut terbalik, perbandingan yang terjadi  $N(N-1) / 2$  demikian *Worst case*  **$O(N^2)$**



1. Jelaskan pengertian pengurutan data
2. Berikan contoh pengurutan data didalam kehidupan nyata yang sehari-hari pernah dijumpai
3. Jelaskan mekanisme pengurutan data pada
  - *Bubble Sort*
  - *Insertion Sort*
  - *Selection Sort*



Selesai