Sorting

Tim Ajar

MATA KULIAH ALGORITMA DAN STRUKTUR DATA

Jurusan Teknologi Informasi

Pokok Bahasan

- ✓ Bubble Sort
- ✓ Selection Sort
- ✓ Insertion Sort

Sorting

- *Sorting* / Pengurutan adalah proses mengurutkan deret data yang awalnya tertata secara acak menjadi data yang tertata berurutan sesuai dengan kebutuhan.
- Tujuan pengurutan adalah untuk memudahkan proses lebih lanjut terkait penggunaan data tersebut, misalnya proses pencarian, pengolahan data, penjadwalan dll
- Penyusunan *sorting* ada, secara ***ascending*** dan ***descending*.** *Ascending* mengurutkan dari data kecil ke besar, dan *Descending* mengurutkan dari data besar ke kecil

Bubble Sort

Bubble Sort

- Merupakan algoritma khusus untuk penyelesaian masalah pengurutan (*sorting*)
- Teknis kerja melalui perbandingan pasangan elemen dari *list* yang tidak terurut dan membalikkan urutan jika ditemukan elemen yang tidak memenuhi ketentuan pengurutan
- Layaknya sebuah gelembung, nilai yang besar/kecil akan bergeser dari kiri ke kanan dengan menukar elemen sekarang dengan elemen setelahnya

compare → swap/no swap

Ilustrasi Pengurutan

- Ditentukan *list* dengan data sebagai berikut:

```
{6,5,3,1,8,7,2,4}
```

- Maka skema pengurutan adalah sebagai berikut:

```
{6,5,3,1,8,7,2,4}
{**5,6**,3,1,8,7,2,4} -- 5 < 6 -> swap
```

Ilustrasi Pengurutan (2)

- Ditentukan list dengan data sebagai berikut:

```
{6,5,3,1,8,7,2,4}
```

- Maka skema pengurutan adalah sebagai berikut:

```
{6,5,3,1,8,7,2,4}
{**5,6**,3,1,8,7,2,4} -- 5 < 6 -> swap
{5,**3,6**,1,8,7,2,4} -- 3 < 6 -> swap
```

Ilustrasi Pengurutan (3)

- Ditentukan list dengan data sebagai berikut:

```
{6,5,3,1,8,7,2,4}
```

- Maka skema pengurutan adalah sebagai berikut:

```
{6,5,3,1,8,7,2,4}
{**5,6**,3,1,8,7,2,4} -- 5 < 6 -> swap
{5,**3,6**,1,8,7,2,4} -- 3 < 6 -> swap
{5,3,**1,6**,8,7,2,4} -- 1 < 6 -> swap
```

Ilustrasi Pengurutan (4)

- Ditentukan *list* dengan data sebagai berikut:

```
{6,5,3,1,8,7,2,4}
```

- Maka skema pengurutan adalah sebagai berikut:

```
{6,5,3,1,8,7,2,4}
{**5,6**,3,1,8,7,2,4} -- 5 < 6 -> swap
{5,**3,6**,1,8,7,2,4} -- 3 < 6 -> swap
{5,3,**1,6**,8,7,2,4} -- 1 < 6 -> swap
{5,3,1,**6,8**,7,2,4} -- 8 > 6 -> no swap
```

Ilustrasi Pengurutan (5)

- Ditentukan *list* dengan data sebagai berikut:

```
{6,5,3,1,8,7,2,4}
```

- Maka skema pengurutan adalah sebagai berikut:

```
{6,5,3,1,8,7,2,4}
{**5,6**,3,1,8,7,2,4} -- 5 < 6 -> swap
{5,**3,6**,1,8,7,2,4} -- 3 < 6 -> swap
{5,3,**1,6**,8,7,2,4} -- 1 < 6 -> swap
{5,3,1,**6,8**,7,2,4} -- 8 > 6 -> no swap
{5,3,1,6,**7,8**,2,4} -- 7 < 8 -> swap
```

Ilustrasi Pengurutan (6)

- Ditentukan *list* dengan data sebagai berikut:

```
{6,5,3,1,8,7,2,4}
```

- Maka skema pengurutan adalah sebagai berikut:

```
{6,5,3,1,8,7,2,4}
{**5,6**,3,1,8,7,2,4} -- 5 < 6 -> swap
{5,**3,6**,1,8,7,2,4} -- 3 < 6 -> swap
{5,3,**1,6**,8,7,2,4} -- 1 < 6 -> swap
{5,3,1,**6,8**,7,2,4} -- 8 > 6 -> no swap
{5,3,1,6,**7,8**,2,4} -- 7 < 8 -> swap
{5,3,1,6,7,**2,8**,4} -- 2 < 8 -> swap
```

Ilustrasi Pengurutan (7)

- Ditentukan *list* dengan data sebagai berikut:

```
{6,5,3,1,8,7,2,4}
```

- Maka skema pengurutan adalah sebagai berikut:

```
{6,5,3,1,8,7,2,4}
{**5,6**,3,1,8,7,2,4} -- 5 < 6 -> swap
{5,**3,6**,1,8,7,2,4} -- 3 < 6 -> swap
{5,3,**1,6**,8,7,2,4} -- 1 < 6 -> swap
{5,3,1,**6,8**,7,2,4} -- 8 > 6 -> no swap
{5,3,1,6,**7,8**,2,4} -- 7 < 8 -> swap
{5,3,1,6,7,**2,8**,4} -- 2 < 8 -> swap
{5,3,1,6,7,2,**4,8**} -- 4 < 8 -> swap
```

Skema Bubble Sort dengan Size data = 8

```
[6, 5, 3, 1, 8, 7, 2, 4]
```

Tahap 0
7 langkah

```
> 6 -> 5 swap : [5, 6, 3, 1, 8, 7, 2, 4]
> 6 -> 3 swap : [5, 3, 6, 1, 8, 7, 2, 4]
> 6 -> 1 swap : [5, 3, 1, 6, 8, 7, 2, 4]
> 6 -> 8 no swap : [5, 3, 1, 6, 8, 7, 2, 4]
> 8 -> 7 swap : [5, 3, 1, 6, 7, 8, 2, 4]
> 8 -> 2 swap : [5, 3, 1, 6, 7, 2, 8, 4]
> 8 -> 4 swap : [5, 3, 1, 6, 7, 2, 4, 8]
```

Tahap 1
6 langkah

```
> 5 -> 3 swap : [3, 5, 1, 6, 7, 2, 4, 8]
> 5 -> 1 swap : [3, 1, 5, 6, 7, 2, 4, 8]
> 5 -> 6 no swap : [3, 1, 5, 6, 7, 2, 4, 8]
> 6 -> 7 no swap : [3, 1, 5, 6, 7, 2, 4, 8]
> 7 -> 2 swap : [3, 1, 5, 6, 2, 7, 4, 8]
> 7 -> 4 swap : [3, 1, 5, 6, 2, 4, 7, 8]
```

Tahap 2
5 langkah

```
> 3 -> 1 swap : [1, 3, 5, 6, 2, 4, 7, 8]
> 3 -> 5 no swap : [1, 3, 5, 6, 2, 4, 7, 8]
> 5 -> 6 no swap : [1, 3, 5, 6, 2, 4, 7, 8]
> 6 -> 2 swap : [1, 3, 5, 2, 6, 4, 7, 8]
> 6 -> 4 swap : [1, 3, 5, 2, 4, 6, 7, 8]
```

Tahap 3
4 langkah

```
> 1 -> 3 no swap : [1, 3, 5, 2, 4, 6, 7, 8]
> 3 -> 5 no swap : [1, 3, 5, 2, 4, 6, 7, 8]
> 5 -> 2 swap : [1, 3, 2, 5, 4, 6, 7, 8]
> 5 -> 4 swap : [1, 3, 2, 4, 5, 6, 7, 8
```

Tahap 4
3 langkah

```
> 1 -> 3 no swap : [1, 3, 2, 4, 5, 6, 7, 8]
> 3 -> 2 swap : [1, 2, 3, 4, 5, 6, 7, 8]
> 3 -> 4 no swap : [1, 2, 3, 4, 5, 6, 7, 8
```

Tahap 5
2 langkah

```
> 1 -> 2 no swap : [1, 2, 3, 4, 5, 6, 7, 8]
> 2 -> 3 no swap : [1, 2, 3, 4, 5, 6, 7, 8
```

Tahap 6
1 langkah

```
> 1 -> 2 no swap : [1, 2, 3, 4, 5, 6, 7, 8
```

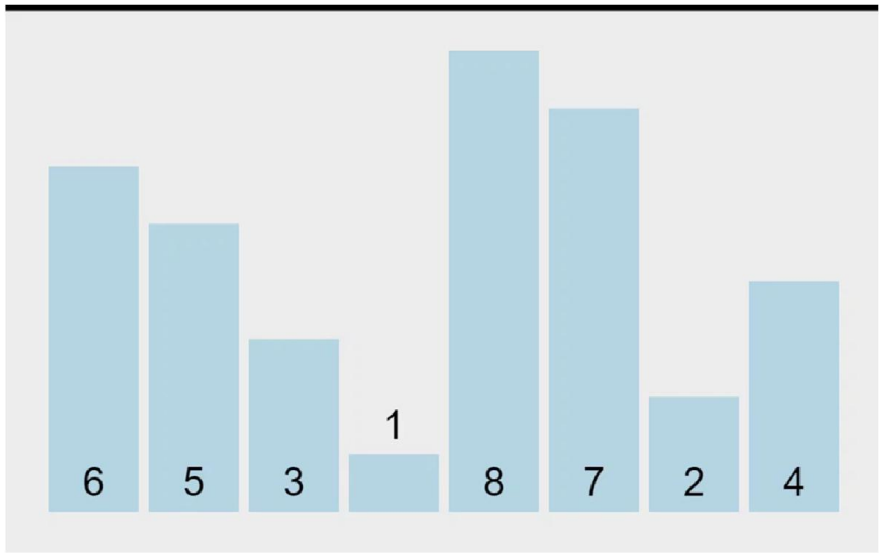
Algoritma Bubble Sort

```
Bubble Sort(arr, size)
for i←0 to size-1
      for j←0 to size-i-1
            if arr[j]>arr[j+1]
                  swap arr[j] and arr[j+1]

return (arr)
```

Keterangan
Nested loop → Outer loop menunjukkan Tahapan dan Inner loop menunjukkan Langkah tiap tahap

Visualisasi Bubble Sort



Selection Sort

Selection Sort

- Proses pengurutan dilakukan melalui pencarian nilai terbesar atau terkecil (tergantung tujuan pengurutan, *ascending* atau *descending*), kemudian terjadi pertukaran (*swapping*) dengan elemen terkiri (awal) yang belum terurut, dan proses berlanjut ke ke elemen berikutnya.
- Berbeda dengan bubble sort yang menukar langsung ketika menemukan elemen yang lebih besar/kecil, selection sort mencari element terkecil/terbesar kemudian menukarnya.

Min / Max → Swap

Ilustrasi Pengurutan

Data = {10,14,27,35,42,19,33,29}

Tahap 0

10	14	27	35	42	19	33	29

Index = 0 ; id = 0
Min awal = 10
- 14 < 10
- 27 < 10
- 35 < 10
- 42 < 10
- 19 < 10
- 33 < 10
- 29 < 10

Swap index 0 dan id 0 {**10**,14,27,35,42,19,33,29}

Tahap 1

10	14	27	35	42	19	33	29

Index = 1 ; id = 1
Min awal = 14
- 27 < 14
- 35 < 14
- 42 < 14
- 19 < 14
- 33 < 14
- 29 < 14

Swap index 1 dan id 1 {**10**,**14**,27,35,42,19,33,29}

Ilustrasi Pengurutan (2)

Tahap 2

10	14	**27**	**35**	**42**	19	**33**	**29**

Index = 2 ; id = 2
Min awal = 27
- 35 < 27
- 42 < 27
- 19 < 27 (min = 19, id = 5)
- 33 < 19
- 29 < 19

Swap index 2 dan id 5 {**10**,**14**,**19**,35,42,27,33,29}

Tahap 3

10	14	19	**35**	**42**	27	**33**	**29**

Index = 3 ; id = 3
Min awal = 35
- 42 < 35
- 27 < 35 (min = 27, id = 5)
- 33 < 27
- 29 < 27

Swap index 3 dan 5 {**10**,**14**,**19**,**27**,42,35,33,29}

Ilustrasi Pengurutan (3)

Tahap 4

10	14	19	27	**42**	**35**	**33**	29

Index = 4 ; id = 4
Min awal = 42
- 35 < 42 (min = 35, id = 5)
- 33 < 35 (min = 33, id = 6)
- 29 < 33 (min = 29, id = 7)

Swap index 4 dan 7 {**10**,**14**,**19**,**27**,**29**,35,33,42}

Tahap 5

10	14	19	27	29	**35**	33	**42**

Index = 5 ; id = 5
Min awal = 35
- 33 < 35 (min = 33, id = 6)
- 42 < 33

Swap index 5 dan 6 {**10**,**14**,**19**,**27**,**29**,**33**,35,42}

Tahap 6

10	14	19	27	29	33	35	**42**

Index = 6 ; id = 6
Min awal = 35
- 42 < 35

Swap index 6 dan 6 {**10**,**14**,**19**,**27**,**29**,**33**,**35**,**42**}

Hasil Akhir Pengurutan

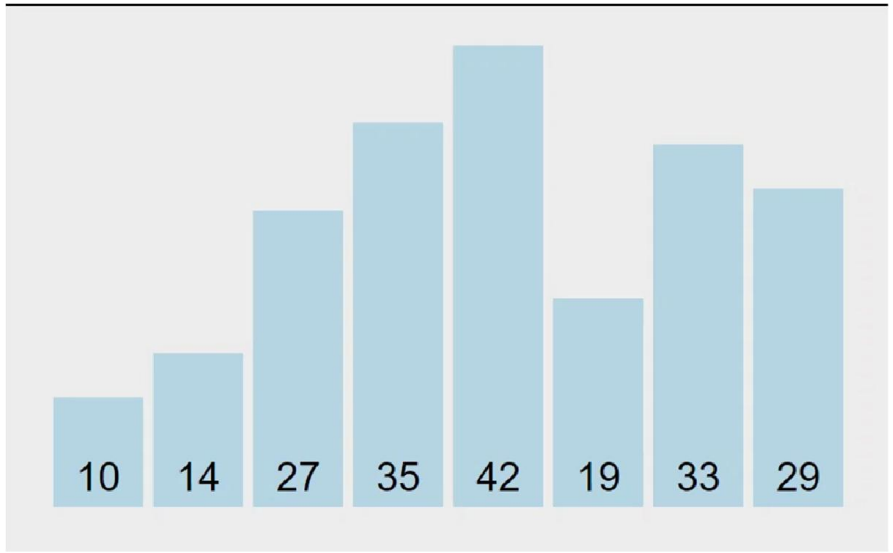
10	14	19	27	29	33	35	42

Algoritma Selection Sort

```
Selection Sort(arr, size)
for i←0 to size-1
      minIndex ← i
      minValue ← arr[i]
      for j←i+1 to size-1
            if arr[j]< minValue
                  minIndex ← j
                  minValue ← arr[j]
      swap arr[i] and arr[minIndex]

return (arr)
```

Visualisasi SelectionSort

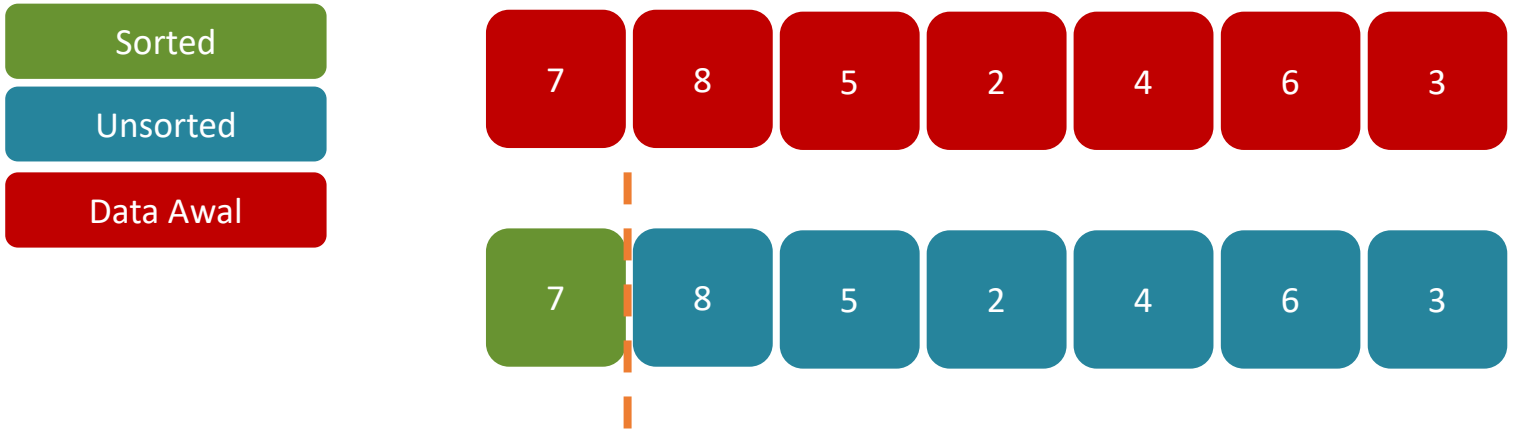


Insertion Sort

Insertion Sort

- Merupakan algoritma yang mengurutkan sederetan angka dengan cara membagi deret angka menjadi dua bagian, bagian *sorted* (terurut) dan bagian *unsorted* (tidak terurut).
- Algoritma ini melakukan penyisipan (*insertion*) nilai di posisi yang tepat pada bagian yang telah terurut

Ilustrasi Pengurutan

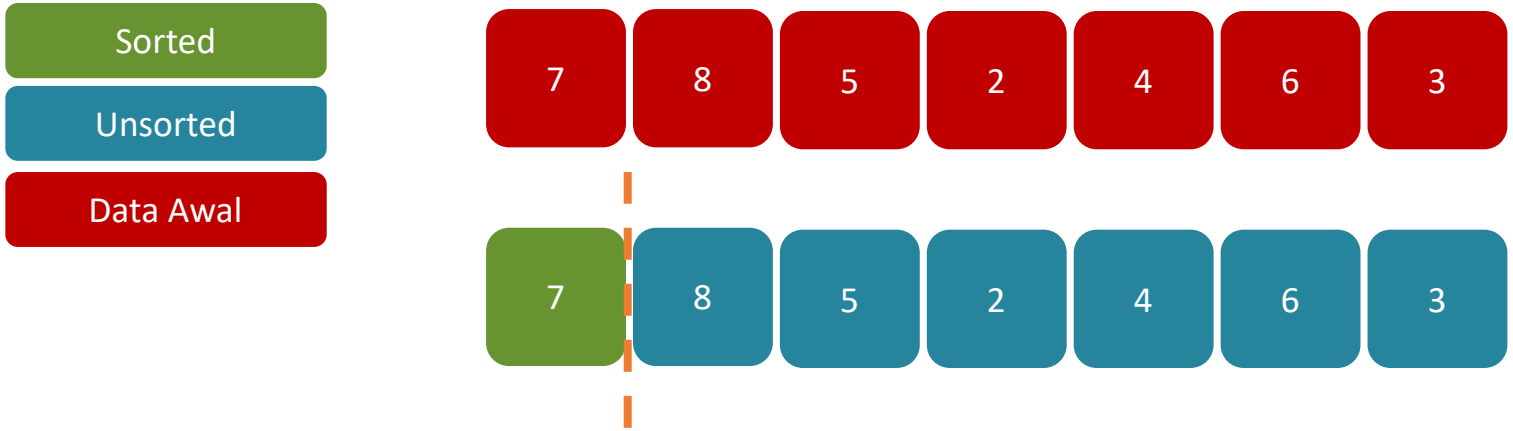


langkah 1 : Data terdiri dari 2 bagian, sorted dan unsorted

Pada langkah pertama item index pertama dari data langsung menjadi bagian sorted

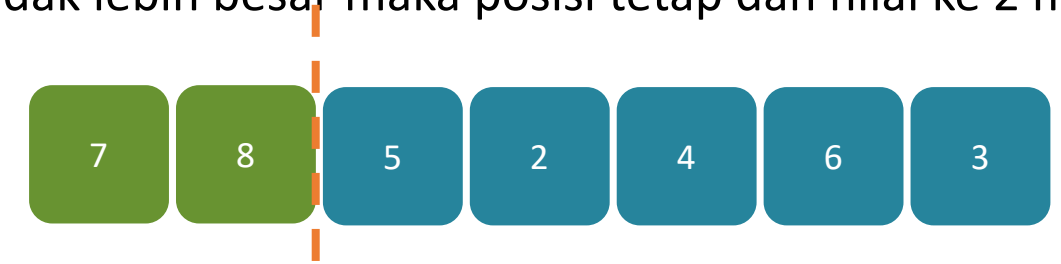
Sisanya menjadi bagian unsorted

Ilustrasi Pengurutan(2)

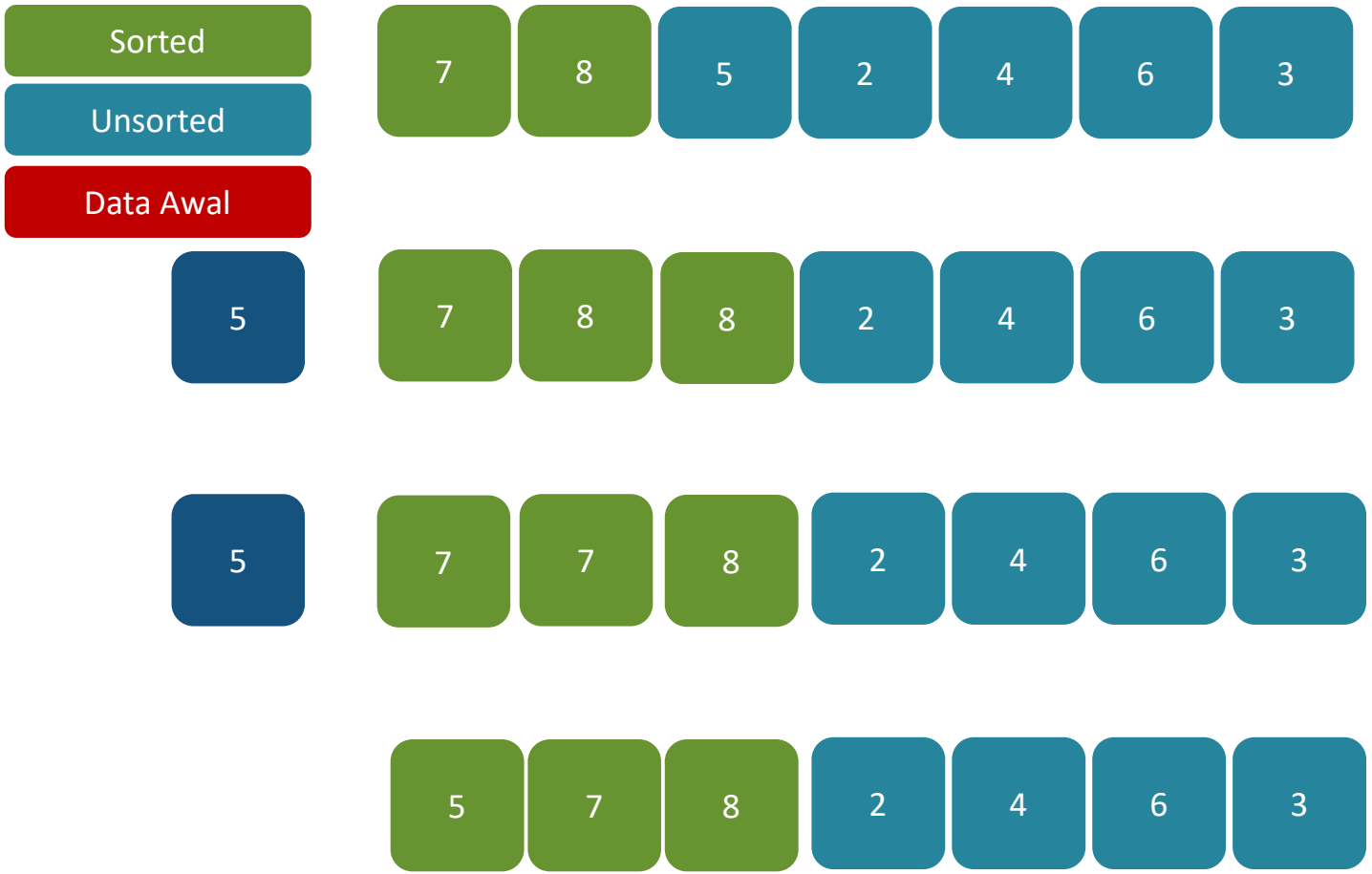


Langkah 2 : dimulai dari nilai ke 2, dibandingkan dengan nilai di kirinya (bagian sorted)

Jika nilai di kirinya tidak lebih besar maka posisi tetap dan nilai ke 2 menjadi bagian sorted

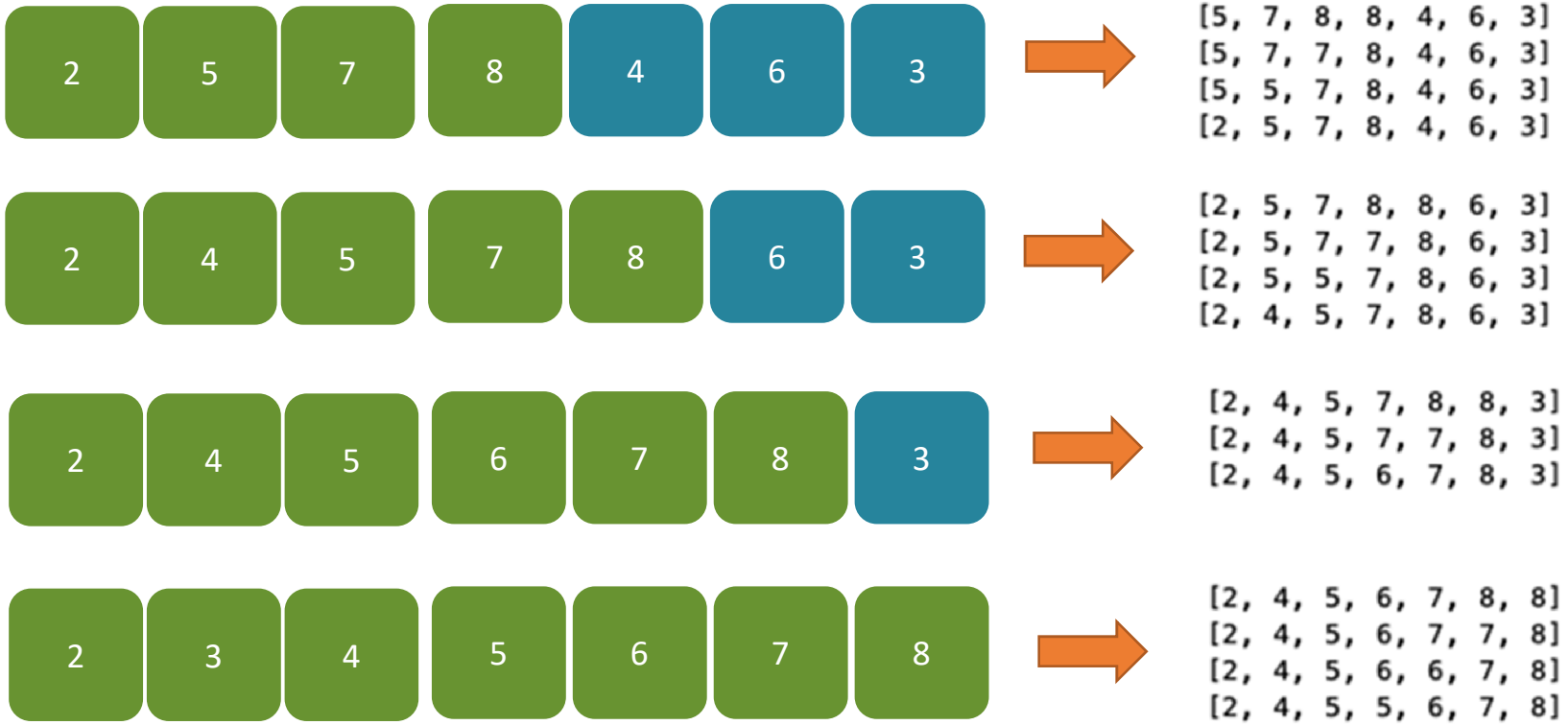


Ilustrasi Pengurutan(3)



Simpan nilai ke 3, dibandingkan dengan masing-masing nilai pada bagian terurut (sorted) mulai dari sebelah kirinya, jika nilai di bagian sorted lebih besar, nilai tersebut bergeser ke kanan, **terus berulang selama di bagian sorted lebih besar dan posisi belum di paling ujung (posisi ke 1)**, **jika ditemukan nilai sorted tidak lebih besar dari nilai yang disimpan atau posisi telah berada di posisi ke 1**, maka berhenti dan **nilai yang disimpan disisipkan di posisi terakhir bergeser**

Ilustrasi Pengurutan(4)




```
[5, 7, 8, 8, 4, 6, 3]
[5, 7, 7, 8, 4, 6, 3]
[5, 5, 7, 8, 4, 6, 3]
[2, 5, 7, 8, 4, 6, 3]

[2, 5, 7, 8, 8, 6, 3]
[2, 5, 7, 7, 8, 6, 3]
[2, 5, 5, 7, 8, 6, 3]
[2, 4, 5, 7, 8, 6, 3]

[2, 4, 5, 7, 8, 8, 3]
[2, 4, 5, 7, 7, 8, 3]
[2, 4, 5, 6, 7, 8, 3]

[2, 4, 5, 6, 7, 8, 8]
[2, 4, 5, 6, 7, 7, 8]
[2, 4, 5, 6, 6, 7, 8]
[2, 4, 5, 5, 6, 7, 8]
[2, 4, 4, 5, 6, 7, 8]
[2, 3, 4, 5, 6, 7, 8]
```

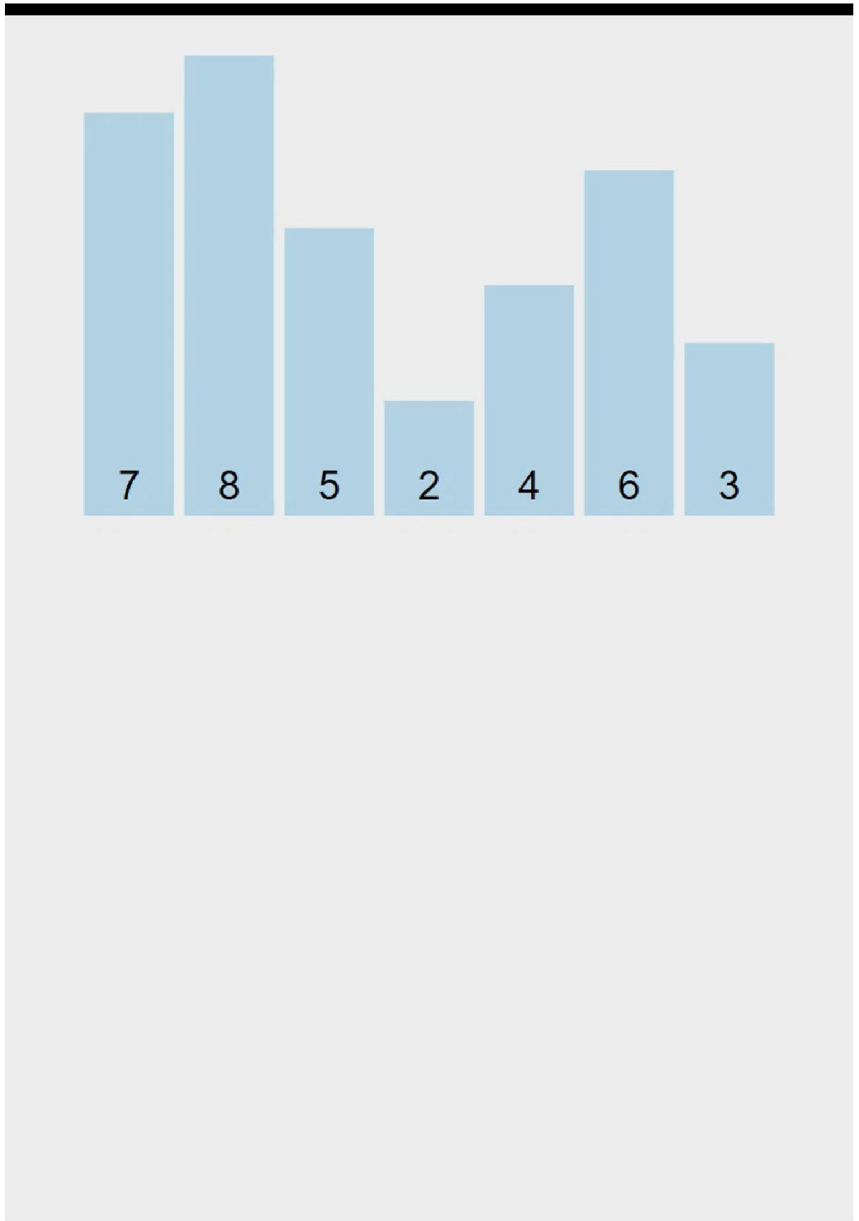
Algoritma Insertion Sort

```
Insertion Sort(arr, size)
for i←0 to size-1
      temp ← arr[i];
      j ← i;
      while (j>0 and arr[j-1]>temp)
            arr[j] ← arr[j-1]

      arr[j] ← temp

return (arr)
```

Visualisasi InsertionSort



Latihan

1. Data = {23,35,14,7,67,89,20}

   Gambarkan proses penyelesaian kasus pengurutan data di atas dengan menggunakan algoritma

   a. Bubble Sort untuk pengurutan descending
   b. Selection Sort untuk pengurutan ascending
   c. Insertion Sort untuk pengurutan descending

2. Jelaskan tindakan yang dilakukan pada algoritma Bubble Sort dan Selection Sort jika menemukan elemen data yang sama nilainya! Contoh = {22,33,45,17,33}