



Bubble Sort dan Selection Sort

Tim Ajar

MATA KULIAH ALGORITMA DAN STRUKTUR DATA

2021/2022

Jurusan Teknologi Informasi

Pokok Bahasan

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



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.



Contoh Skema Bubble Sort

- Ditentukan *list* dengan data sebagai berikut:

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

Contoh Skema Bubble Sort

- 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
```

Contoh Skema Bubble Sort

- 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
```

Contoh Skema Bubble Sort

- 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
```


Contoh Skema Bubble Sort

- 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
```

Contoh Skema Bubble Sort

- 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
```

Contoh Skema Bubble Sort

- 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
```

Contoh Skema Bubble Sort

- 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
```

Contoh Skema Bubble Sort

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

> 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]

> 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]

> 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]


> 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]

> 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]

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

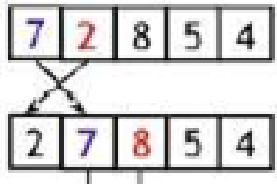
> 1 → 2 no swap : [1, 2, 3, 4, 5, 6, 7, 8]

Contoh Lain

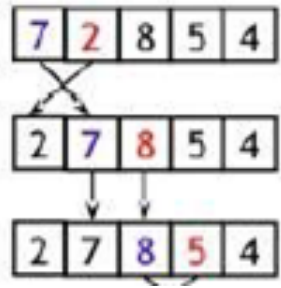


7	2	8	5	4
---	---	---	---	---

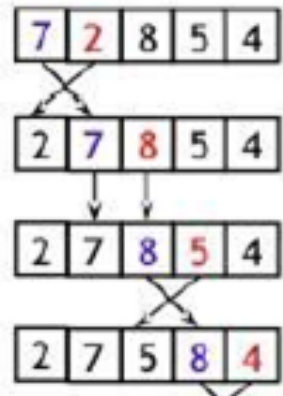
Contoh Lain



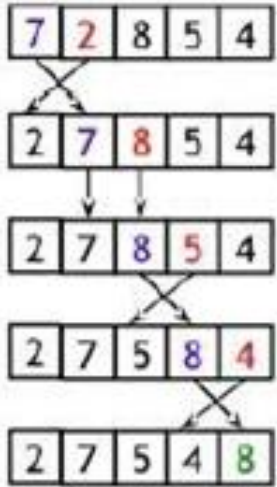
Contoh Lain



Contoh Lain



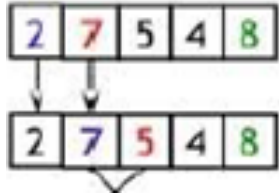
Contoh Lain



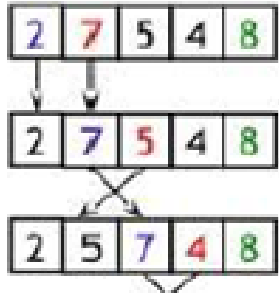
Contoh Lain

2	7	5	4	8
---	---	---	---	---

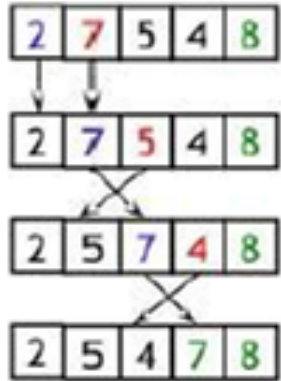
Contoh Lain



Contoh Lain



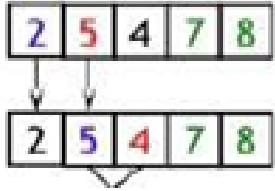
Contoh Lain



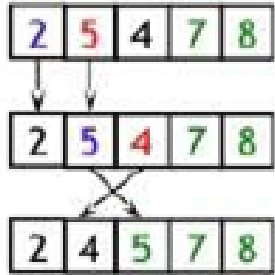
Contoh Lain

2	5	4	7	8
---	---	---	---	---

Contoh Lain

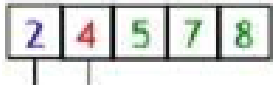



Contoh Lain

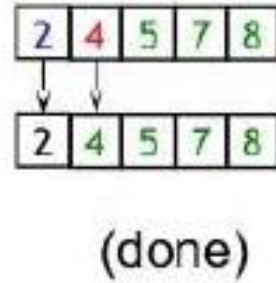
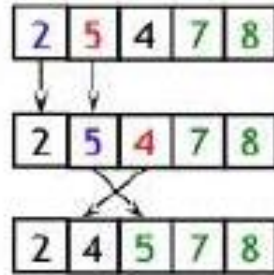
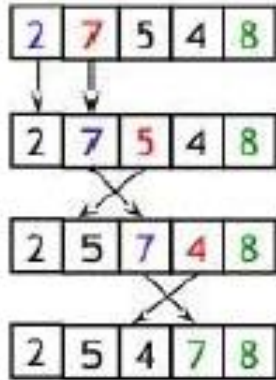
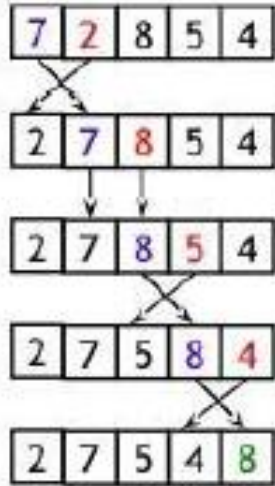


Contoh Lain

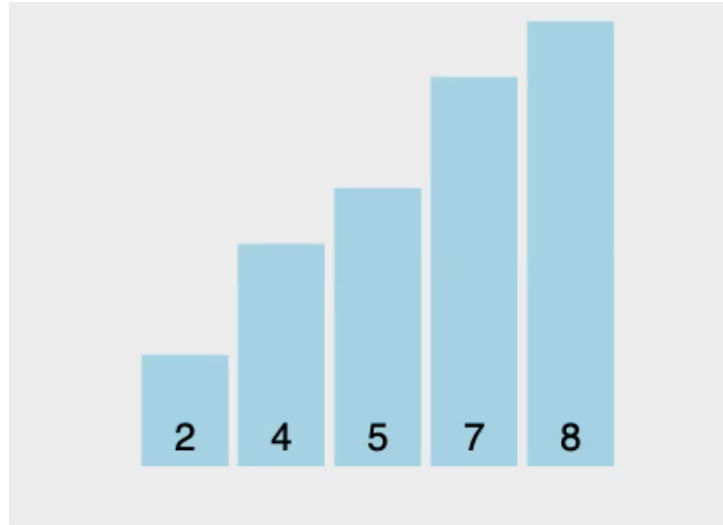
2	4	5	7	8
---	---	---	---	---



Contoh Lain



Visualisasi Bubble Sort



Algoritma Bubble Sort

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



Selection Sort

Selection Sort

- Merupakan algoritma yang digunakan untuk menyelesaikan masalah pengurutan dengan cara membagi input list ke dalam dua bagian dimana sublist telah terurut.
- Proses pengurutan dilakukan melalui pencairan nilai terbesar atau terkecil (tergantung tujuan pengurutan), 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.

Selection Sort (Pseudocode)

```
function select(list[1..n], k)
  for i from 1 to k
    minIndex = i
    minValue = list[i]
    for j from i+1 to n
      if list[j] < minValue
        minIndex = j
        minValue = list[j]
    swap list[i] and list[minIndex]
  return list[k]
```


Ilustrasi Pengurutan

1. Elemen dengan tanda daerah hijau artinya telah terurut, sehingga proses pencarian nilai elemen terkecil tidak dilakukan
2. Nilai elemen = 27 bisa saja masuk ke daerah hijau, namun karena posisinya memungkinkan untuk berubah, maka 27 bertindak sebagai leftmost yang nantinya dibandingkan dengan nilai terkecil dari elemen yang belum terurut



[10, 14, 27, 33, 35, 19, 42, 44]

> 14 → 10

> 27 → 10

> 33 → 10

> 35 → 10

> 19 → 10

> 42 → 10

> 44 → 10

id 0 → 10 : [10, 14, 27, 33, 35, 19, 42, 44]

> 27 → 14

> 33 → 14

> 35 → 14

> 19 → 14

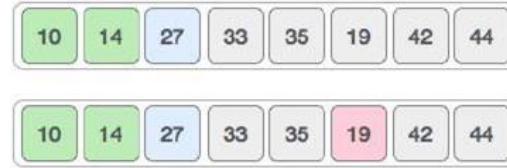
> 42 → 14

> 44 → 14

id 1 → 14 : [10, 14, 27, 33, 35, 19, 42, 44]

Ilustrasi Pengurutan

1. Elemen terkecil ditemukan yaitu 19, maka terjadi perbandingan 19 dengan 27



Ilustrasi Pengurutan

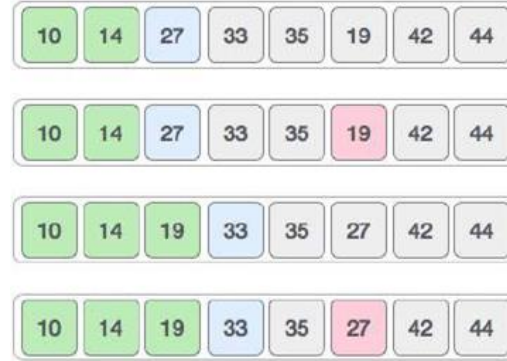
1. Telah terjadi perbandingan dan 19 menempati posisi yang sebelumnya ditempati oleh 27
2. 33 menjadi leftmost



```
> 33 -> 27
> 35 -> 27
> 19 -> 27
> 42 -> 27
> 44 -> 27
id 5 -> 19 : [10, 14, 19, 33, 35, 27, 42, 44]
```

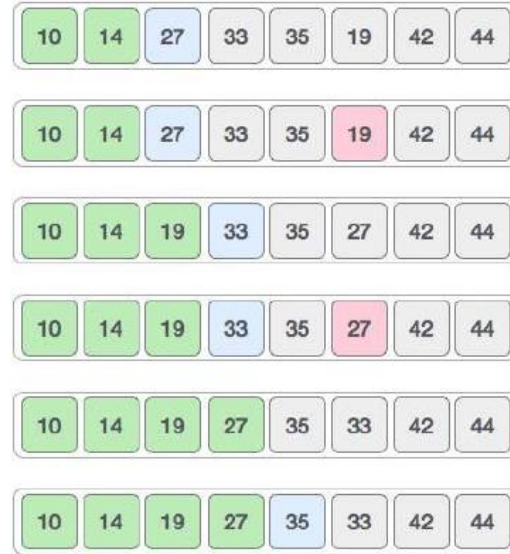
Ilustrasi Pengurutan

1. 33 dibandingkan dengan elemen terkecil yaitu 27



Ilustrasi Pengurutan

1. 35 menjadi leftmost



Ilustrasi Pengurutan

1. 35 dibandingkan dengan elemen terkecil yaitu 33

```
> 35 -> 33  
> 27 -> 33  
> 42 -> 33  
> 44 -> 33  
id 5 -> 27 : [10, 14, 19, 27, 35, 33, 42, 44]
```



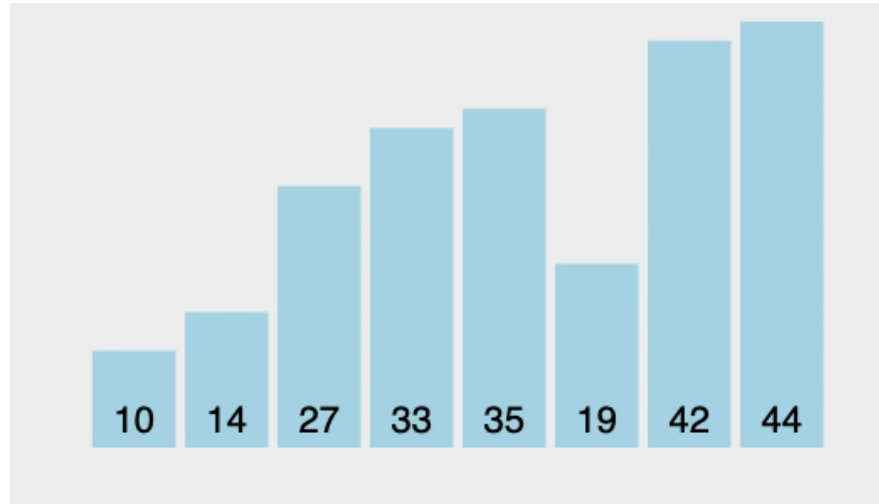
Ilustrasi Pengurutan

1. Proses berlanjut dan telah terurut pada akhirnya

```
> 33 -> 35  
> 42 -> 35  
> 44 -> 35  
id 5 -> 33 : [10, 14, 19, 27, 33, 35, 42, 44]  
  
> 42 -> 35  
> 44 -> 35  
id 5 -> 35 : [10, 14, 19, 27, 33, 35, 42, 44]  
  
> 44 -> 42  
id 6 -> 42 : [10, 14, 19, 27, 33, 35, 42, 44]
```



Visualisasi SelectionSort



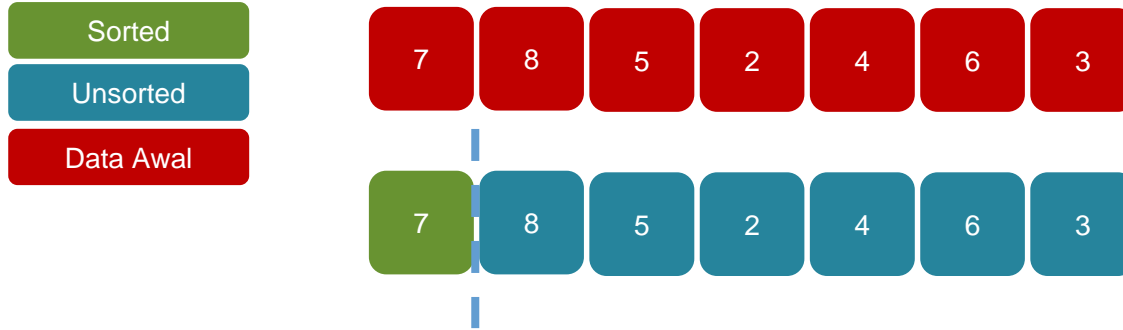


Insertion Sort

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 pertukaran elemen berdasarkan urutan *ascending* / *descending* pada bagian *sorted* (terurut)

Ilustrasi Pengurutan

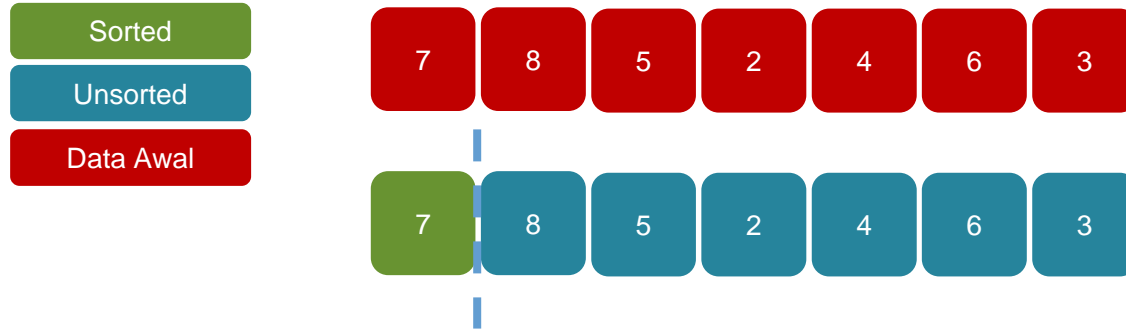


Step 1 : Membagi dua data, menjadi bagian sorted dan unsorted

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

Sisanya menjadi bagian unsorted

Ilustrasi Pengurutan(2)

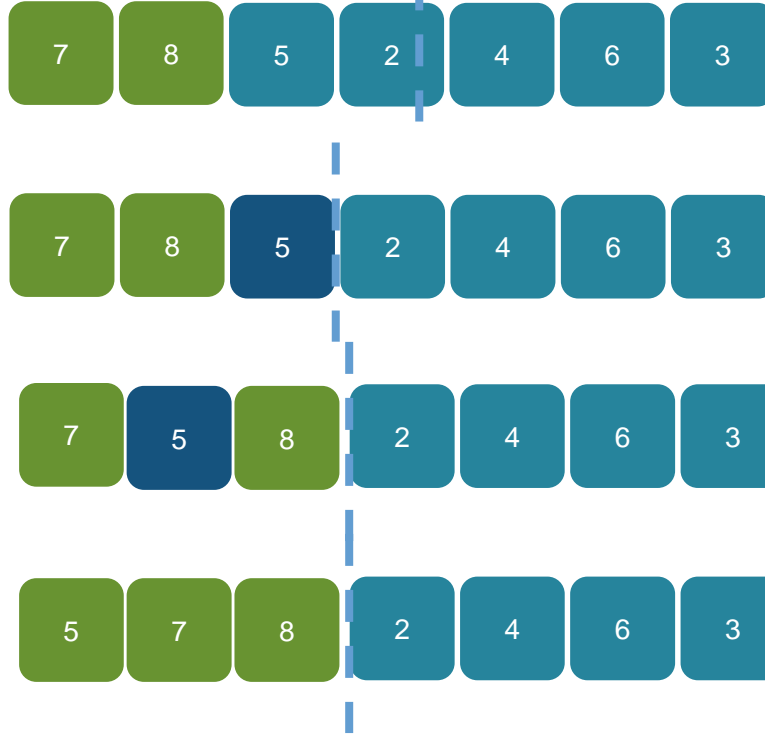


Step 2 : Geser batas bagian sorted satu langkah ke kanan

Jika bagian sorted sudah berurutan sesuai dengan urutan yang diinginkan angka pada bagian sorted dibiarkan, jika belum lakukan pengurutan data di bagian sorted.



Ilustrasi Pengurutan(3)



Ilustrasi Pengurutan(4)



> 8 → 5 : [7, 8, 5, 2, 4, 6, 3]

> 7 → 8 : [7, 8, 8, 2, 4, 6, 3]

> 7 → 8 : [7, 7, 8, 2, 4, 6, 3]

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

> 8 → 2 : [5, 7, 8, 8, 4, 6, 3]

> 7 → 8 : [5, 7, 7, 8, 4, 6, 3]

> 5 → 8 : [5, 5, 7, 8, 4, 6, 3]

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

> 8 → 4 : [2, 5, 7, 8, 8, 6, 3]

> 7 → 8 : [2, 5, 7, 7, 8, 6, 3]

> 5 → 8 : [2, 5, 5, 7, 8, 6, 3]

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

> 8 → 6 : [2, 4, 5, 7, 8, 8, 3]

> 7 → 8 : [2, 4, 5, 7, 7, 8, 3]

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

> 8 → 3 : [2, 4, 5, 6, 7, 8, 8]

> 7 → 8 : [2, 4, 5, 6, 7, 7, 8]

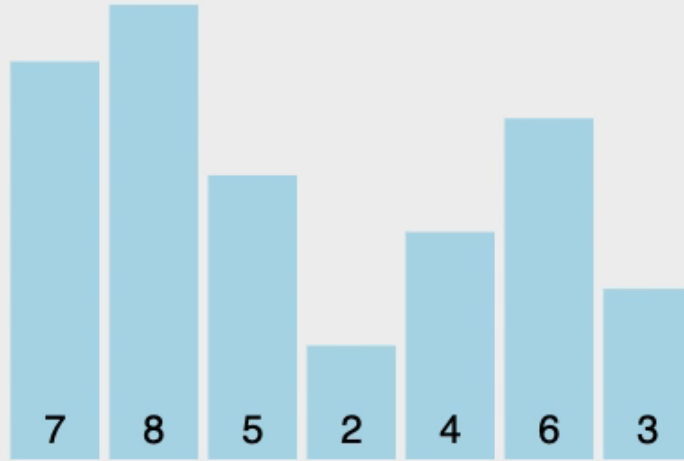
> 6 → 8 : [2, 4, 5, 6, 6, 7, 8]

> 5 → 8 : [2, 4, 5, 5, 6, 7, 8]

> 4 → 8 : [2, 4, 4, 5, 6, 7, 8]

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

Visualisasi InsertionSort



Tugas

1. Gambarkan proses penyelesaian kasus pengurutan data menggunakan Bubble Sort untuk data = {20,35,14,7,67,89,23,46}
2. Gambarkan proses penyelesaian kasus pengurutan data menggunakan Selection Sort untuk data = {39,14,67,29,65}!
3. Gambarkan proses penyelesaian kasus pengurutan data menggunakan Insertion Sort untuk data = {11,13,0,91,11}!
4. Jelaskan tindakan yang dilakukan pada algoritma Bubble Sort dan Selection Sort jika menemukan elemen data yang sama nilainya! Contoh = {22,33,45,17,33}