FAKULTAS
TEKNOLOGI
INFORMASI
Universitas Advent Indonesia

Lecture 5 - Searching and Sorting Algorithms

Friday, September 29, 2023

Data Structure

Introduction to Searching Algorithms

- Algoritma pencarian adalah sebuah metode untuk menemukan item informasi tertentu dalam kumpulan data yang lebih besar.

- Algoritma pencarian dirancang untuk memeriksa elemen atau mengambil elemen dari struktur data dimana elemen tersebut disimpan.

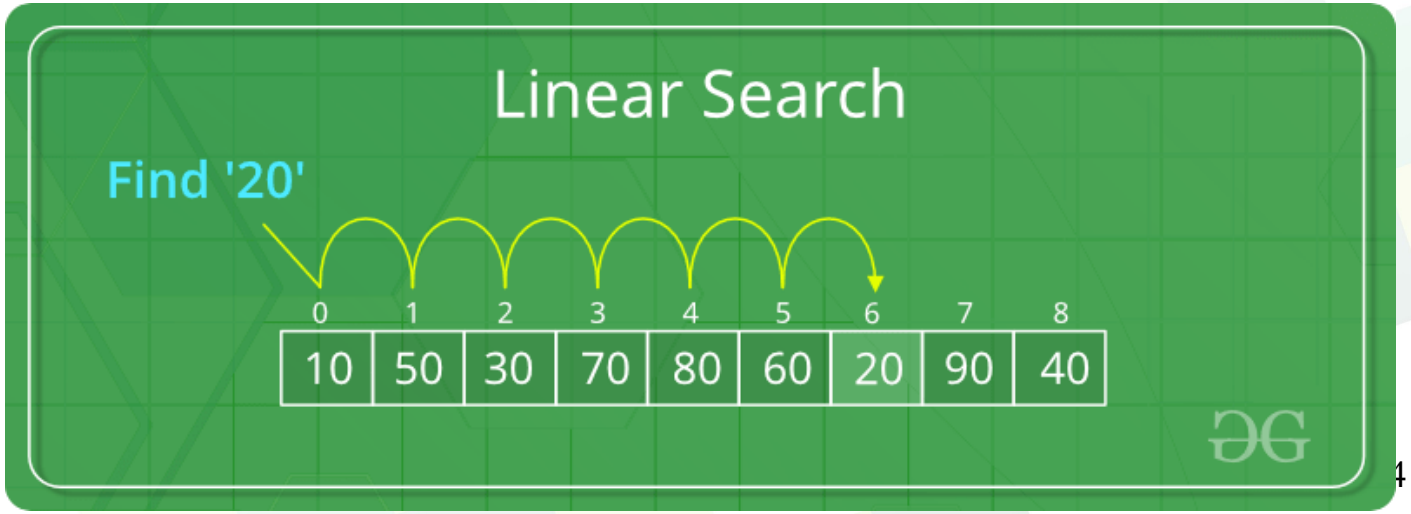
Introduction to Searching Algorithms

Berdasarkan pada jenis operasi pencarian, algoritma ini umumnya diklasifikasikan ke dalam dua kategori:

- **Sequential Search**: Dalam hal ini, daftar atau array dilalui secara berurutan dan setiap elemen diperiksa. Misalnya: **Pencarian Linier/*Linear Search***.
- **Interval Search**: Algoritma ini secara khusus dirancang untuk pencarian dalam struktur data yang diurutkan. Jenis algoritma pencarian ini jauh lebih efisien daripada Pencarian Linier karena mereka berulang kali menargetkan pusat struktur pencarian dan membagi ruang pencarian menjadi dua. Sebagai Contoh: **Pencarian Biner/*Binary Search***.

The Linear Search

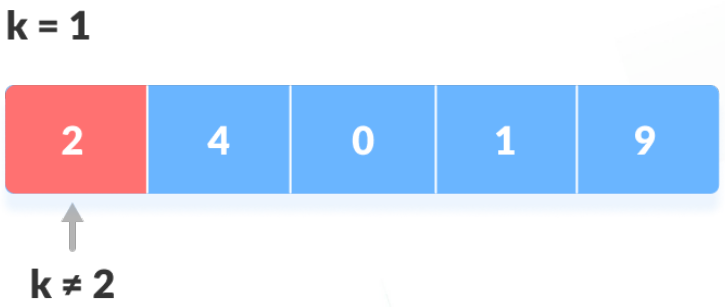
- Ini adalah algoritma yang sangat sederhana.
- Menggunakan loop untuk secara berurutan melangkah melalui array, dimulai dengan elemen pertama.
- Yang selanjutnya dilakukan adalah membandingkan setiap elemen dengan nilai yang dicari dan berhenti Ketika nilai tersebut ditemukan atau sudah sampai pada elemen terakhir dalam array

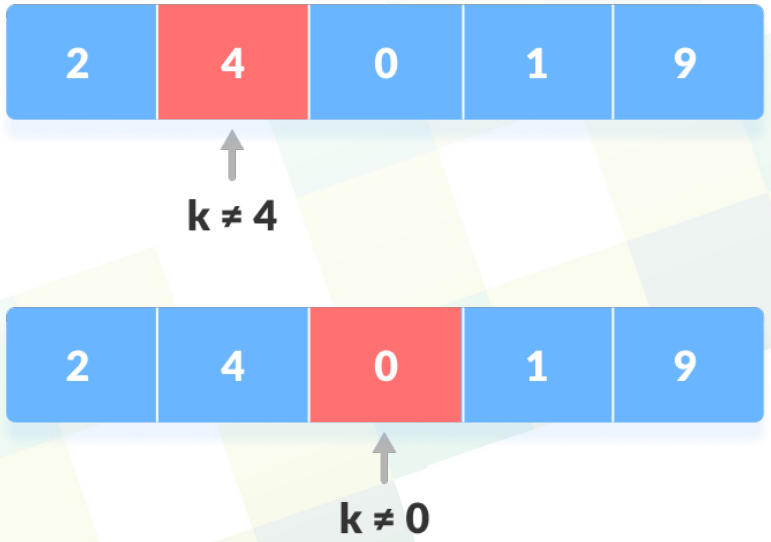


How Linear Search Works

Misalkan, item/elemen yang akan dicari adalah 1 → **k = 1**.

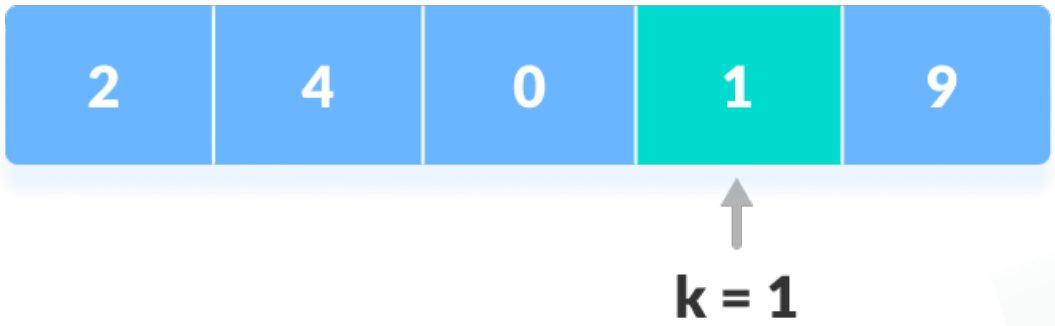
1. Mulai dari elemen pertama, bandingkan **k** dengan setiap elemen dari array.





How Linear Search Works

2. Jika elemen array == k, return index.



3. Else, return ***`tidak ditemukan`***.

The Linear Search Algorithm

```
LinearSearch(array, key)
    for each item in the array
        if item == value
            return its index
```

Efficiency of the Linear Search

- Keuntungan dari algoritma ini adalah kesederhanaannya.
  - Mudah dimengerti
  - Mudah diimplementasikan
  - Tidak membutuhkan array agar berurutan
- Kekurangan algoritma ini adalah inefisiensi/tidak efisien.
  - Jika terdapat 20.000 item dalam array dan apa yang kita cari ada di elemen 19.999, maka kita perlu melakukan pencarian di seluruh array.

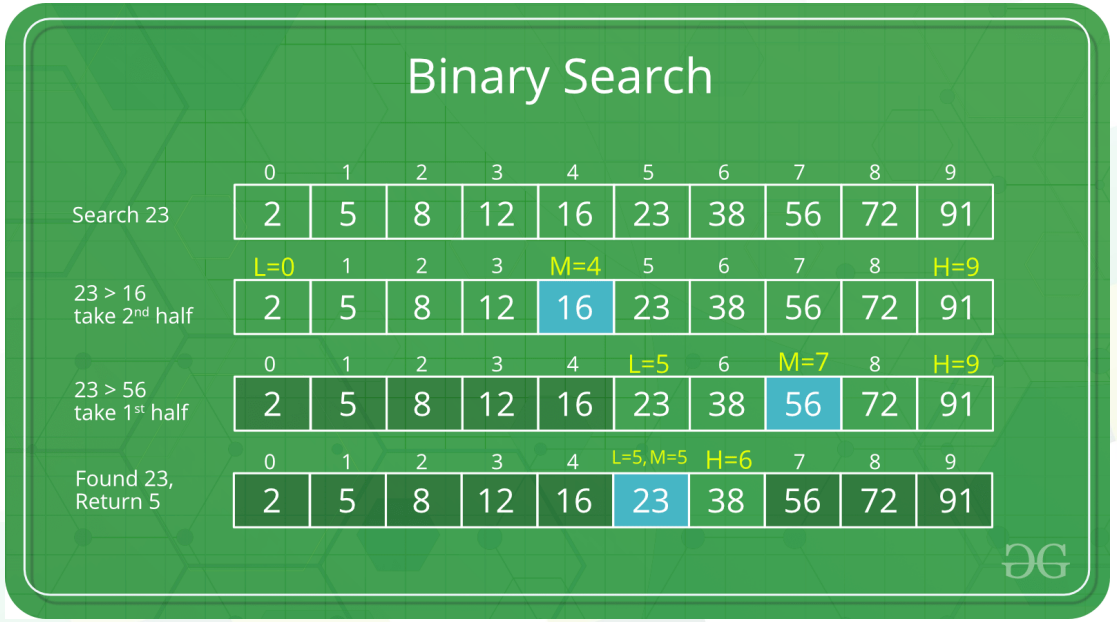
FAKULTAS
TEKNOLOGI
INFORMASI
Universitas Advent Indonesia

Binary Search Algorithm

Data Structure - Searching Algorithm

Binary Search

- Pencarian biner jauh lebih efisien daripada pencarian linear.
- Namun, algoritma ini membutuhkan elemen dalam **array/list berada dalam keadaan terurut**.
- Pencarian akan dilakukan dengan mencari elemen tengah dari array



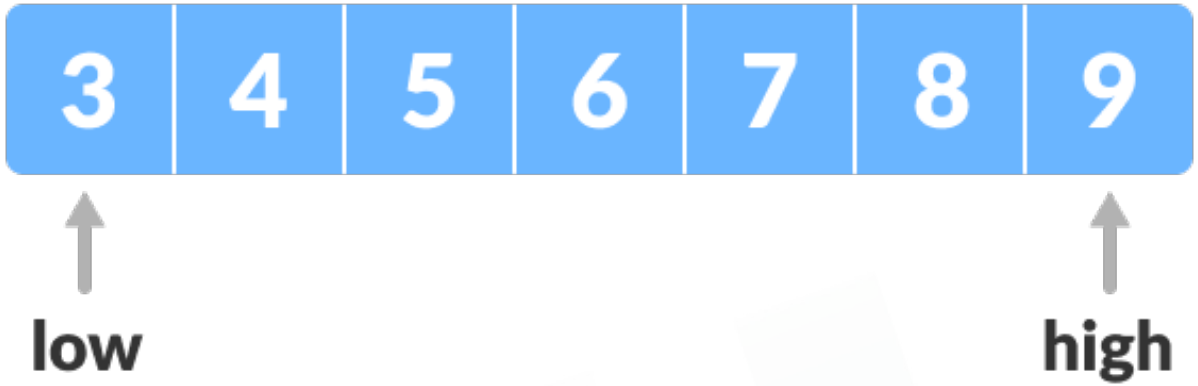
How Binary Search Works

- Algoritma Pencarian Biner dapat diimplementasikan dalam dua cara yakni :
  1. Iterative Method
  2. Recursive Method (cara yang sama dengan metode *Divide and Conquer*)
- Cara umumnya adalah:

1. Misalkan x = 4 adalah elemen yang akan dicari

How Binary Search Works

2. Deklarasikan 2 pointer yakni low dan high untuk menyimpan posisi terendah dan tertinggi.



How Binary Search Works

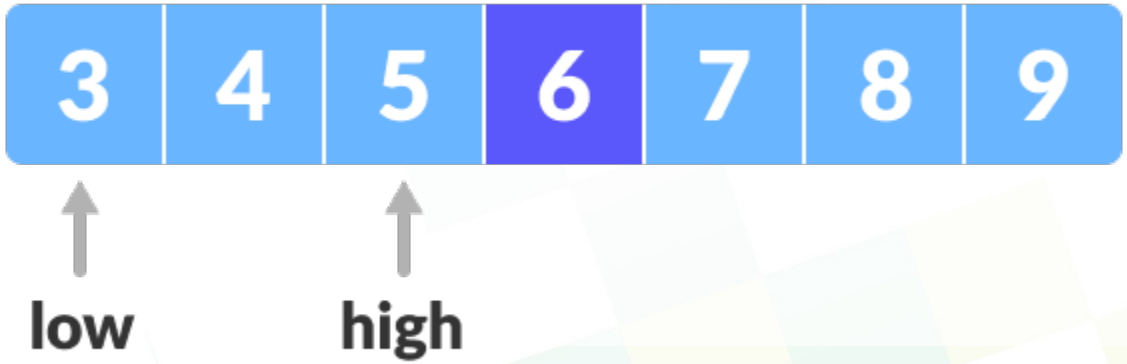
3. Tentukan elemen yang berada di tengah-tengah array. (**mid = (low + high)/2)**

4. Jika x == mid, maka return mid. Else, bandingkan elemen yang akan dicari dengan mid.

How Binary Search Works

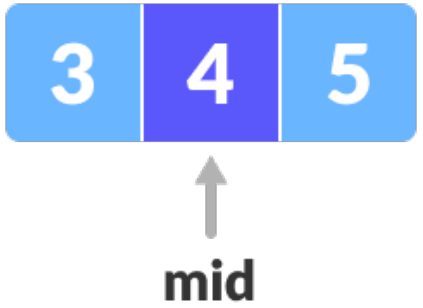
5. Jika **`x > mid`**, bandingkan **`x`** dengan elemen tengah dari sisi kanan **`mid`**. Hal ini dapat dilakukan dengan mengatur **`low = mid + 1`**.

6. Else, bandingkan **`x`** dengan elemen tengah dari sisi kiri **`mid`**. Hal ini dapat dilakukan dengan mengatur **`high = mid - 1`**.

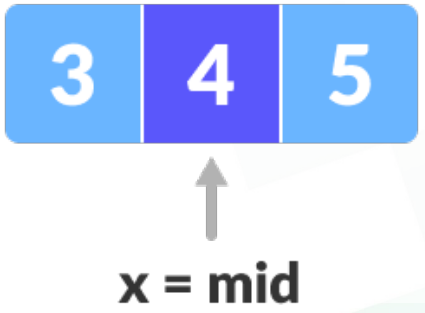


How Binary Search Works

7. Ulangi langkah 3 hingga 6 sampai nilai **`low = high.`**



`8. x = 4` telah ditemukan.



FAKULTAS
TEKNOLOGI
INFORMASI

Universitas Advent Indonesia

Lecture 6 - Sorting Algorithms

Friday, October 6, 2023

Data Structure

Introduction to Sorting Algorithms

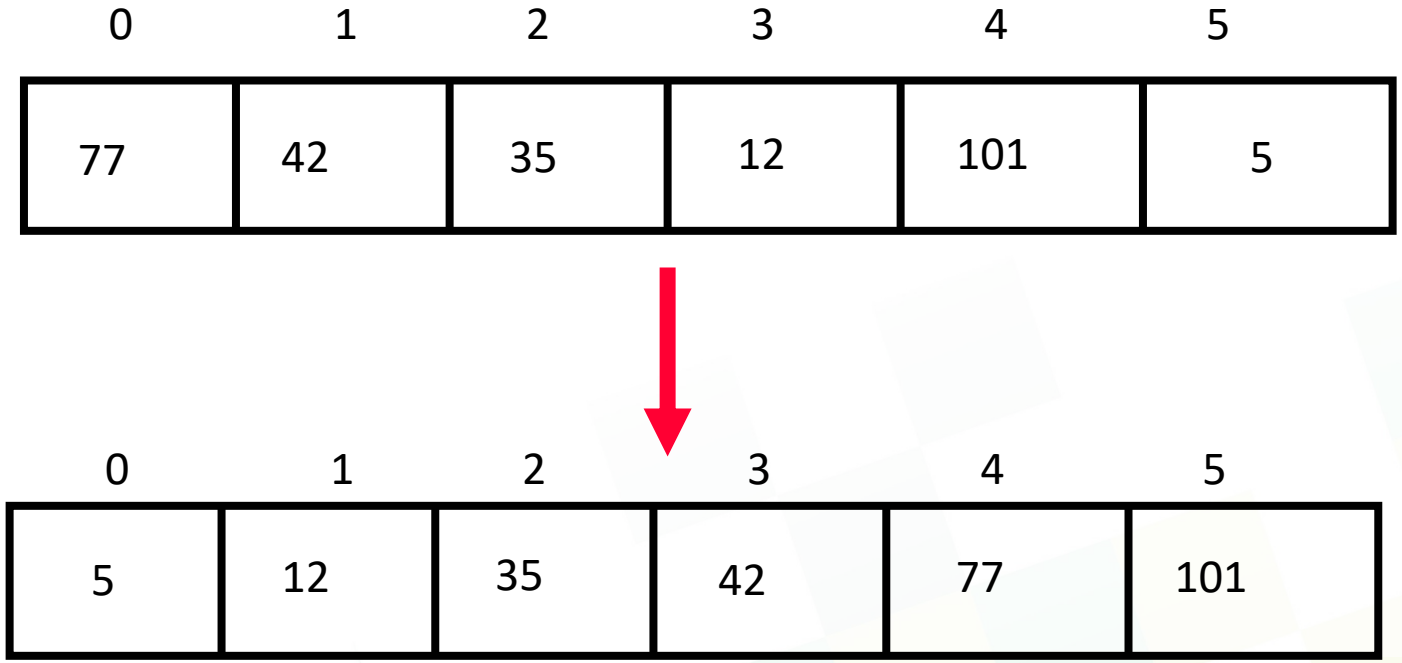
- Pengurutan (Sorting) adalah proses menyusun kembali data yang sebelumnya telah disusun dengan suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu.
- Pengurutan data dalam struktur data sangat penting untuk data yang bertipe data numerik ataupun karakter.
- Pengurutan dapat dilakukan secara ascending (urut naik) dan descending (urut turun).

Contoh:

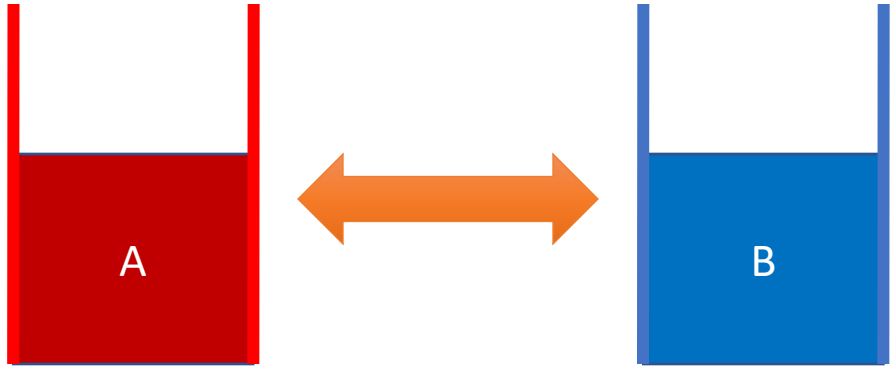
- Data Acak : 5 6 8 1 3 25 10
- Ascending : 1 3 5 6 8 10 25
- Descending : 25 10 8 6 5 3 1

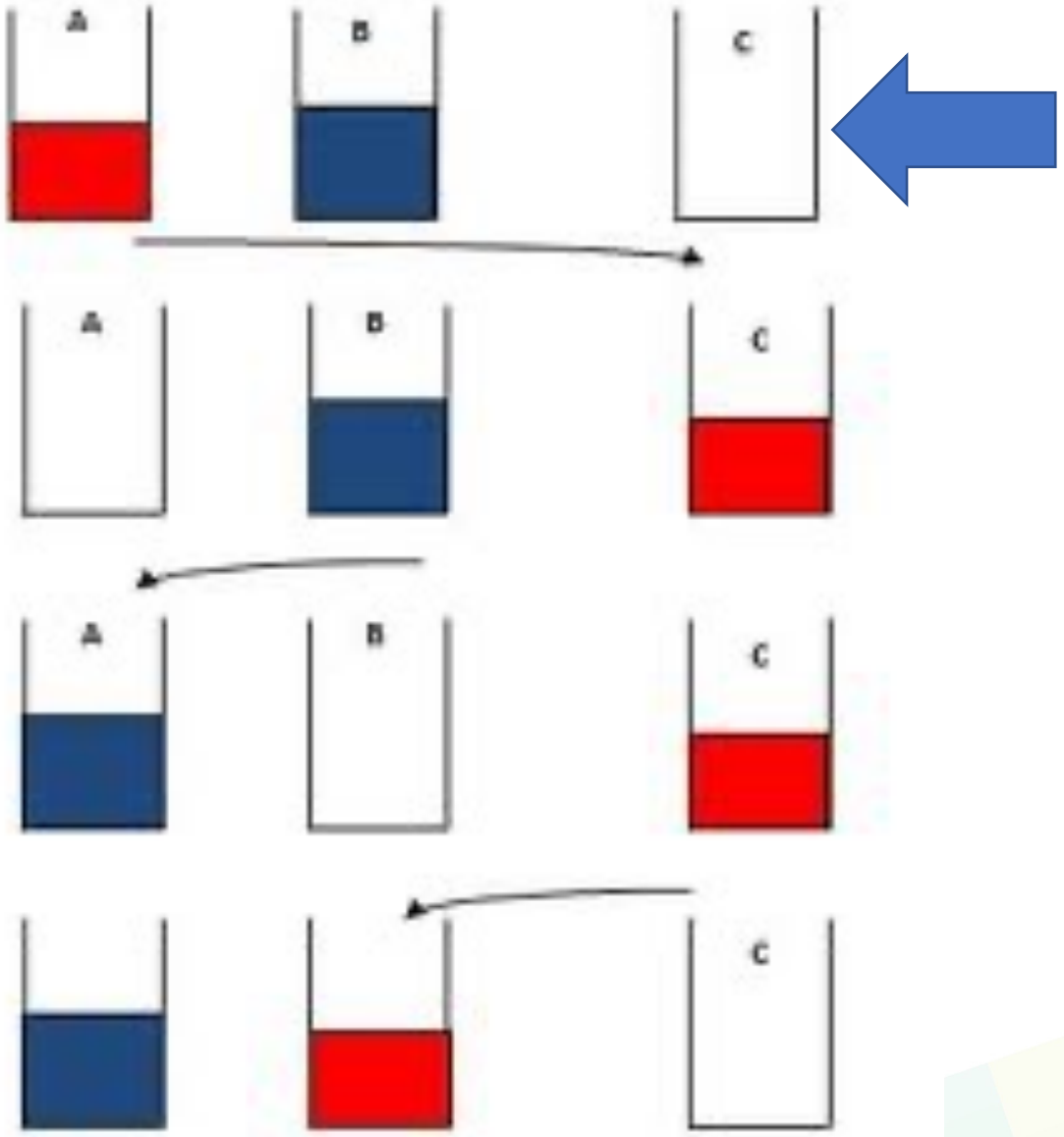
Sorting

- Pengurutan menggunakan koleksi/kumpulan data yang tidak terurut dan membuatnya menjadi terurut.



Swapping Algorithm





Temporary place

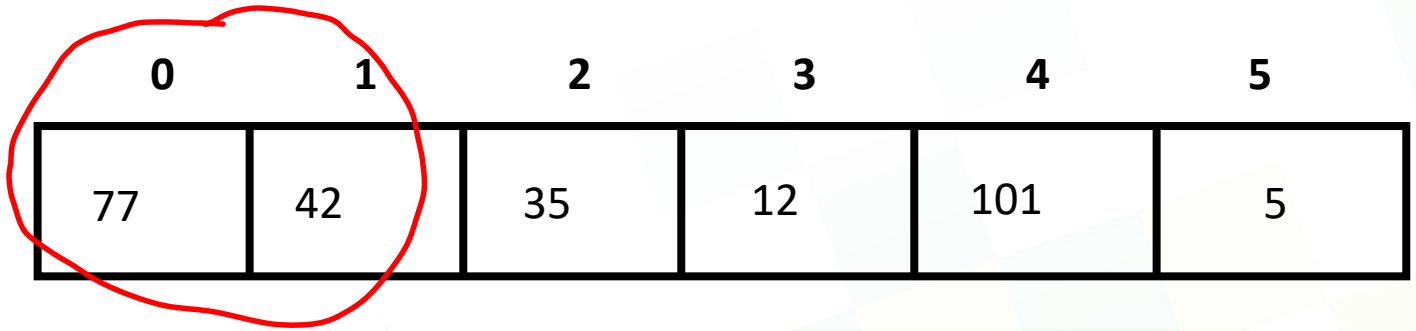
FAKULTAS TEKNOLOGI INFORMASI
Universitas Advent Indonesia

Swapping Algorithm

```
tmp = a;
a = b;
b = tmp;
-------------------------------
void swap(int* a ,int* b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

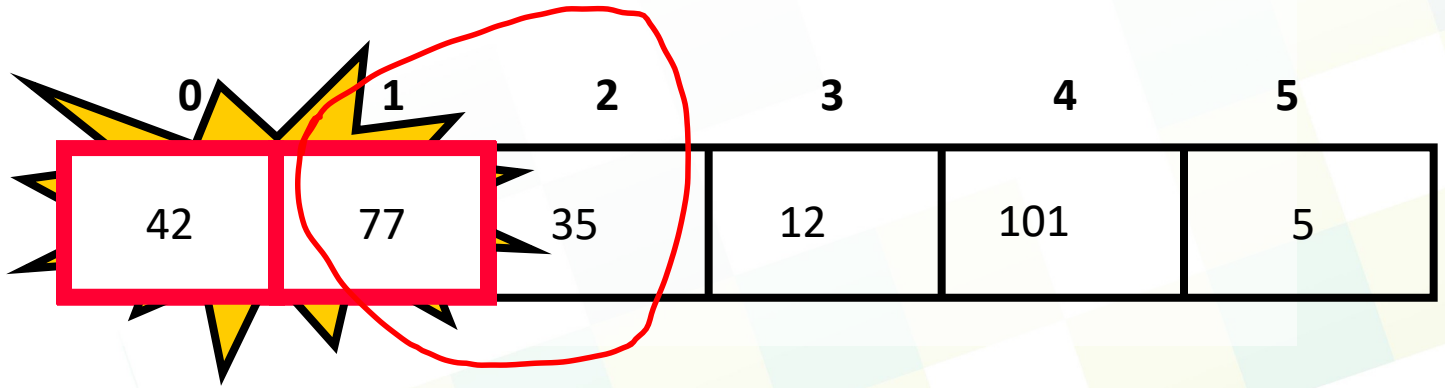
"Bubbling Up" the Largest Element

- Menelusuri kumpulan elemen
  - Bergerak dari depan sampai ke belakang (elemen terakhir/end of array)
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping – algoritma ini melakukan perbandingan berpasangan terlebih dahulu kemudian melakukan pertukaran.



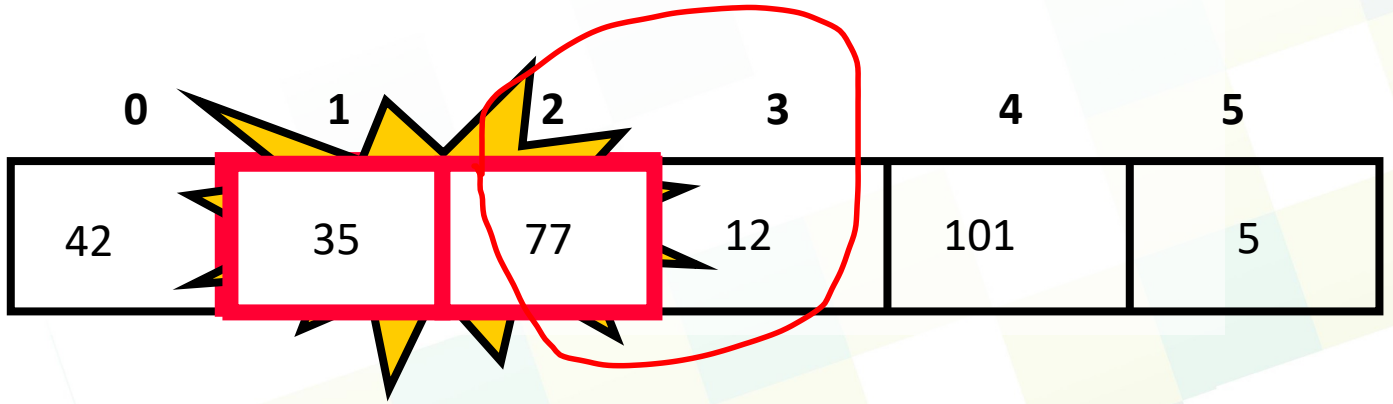
"Bubbling Up" the Largest Element

- Menelusuri kumpulan elemen
  - Bergerak dari depan sampai ke belakang (elemen terakhir/end of array)
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping – algoritma ini melakukan perbandingan berpasangan terlebih dahulu kemudian melakukan pertukaran.



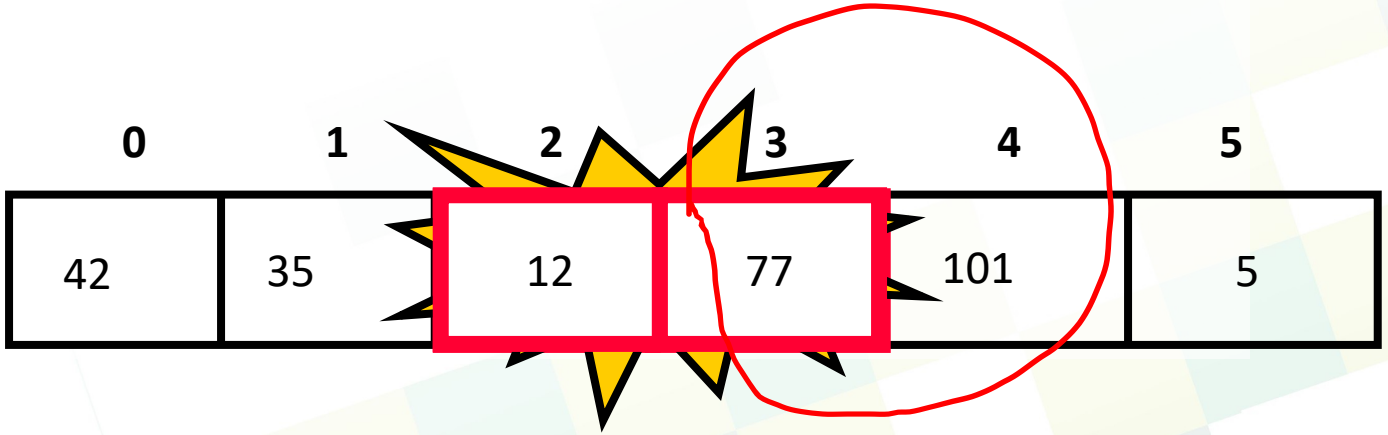
"Bubbling Up" the Largest Element

- Menelusuri kumpulan elemen
  - Bergerak dari depan sampai ke belakang (elemen terakhir/end of array)
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping – algoritma ini melakukan perbandingan berpasangan terlebih dahulu kemudian melakukan pertukaran.



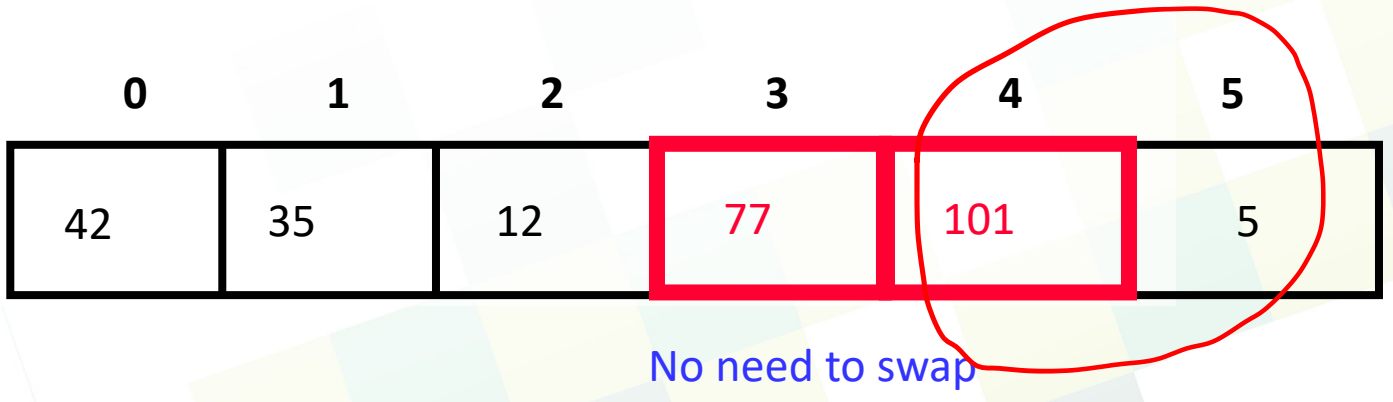
"Bubbling Up" the Largest Element

- Menelusuri kumpulan elemen
  - Bergerak dari depan sampai ke belakang (elemen terakhir/end of array)
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping – algoritma ini melakukan perbandingan berpasangan terlebih dahulu kemudian melakukan pertukaran.



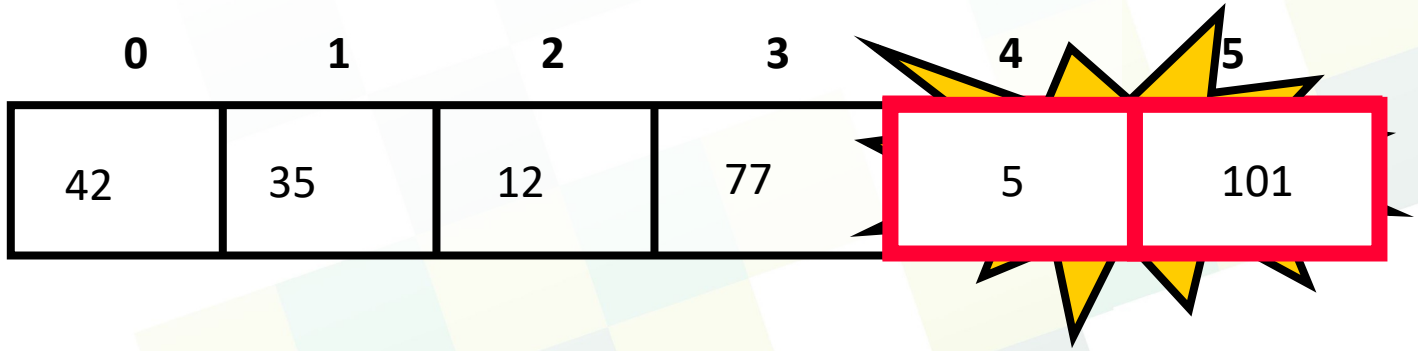
"Bubbling Up" the Largest Element

- Menelusuri kumpulan elemen
  - Bergerak dari depan sampai ke belakang (elemen terakhir/end of array)
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping – algoritma ini melakukan perbandingan berpasangan terlebih dahulu kemudian melakukan pertukaran.



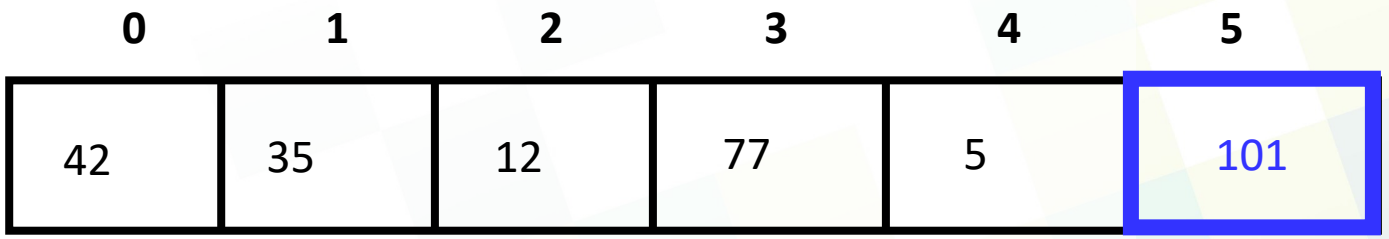
"Bubbling Up" the Largest Element

- Menelusuri kumpulan elemen
  - Bergerak dari depan sampai ke belakang (elemen terakhir/end of array)
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping – algoritma ini melakukan perbandingan berpasangan terlebih dahulu kemudian melakukan pertukaran.



"Bubbling Up" the Largest Element

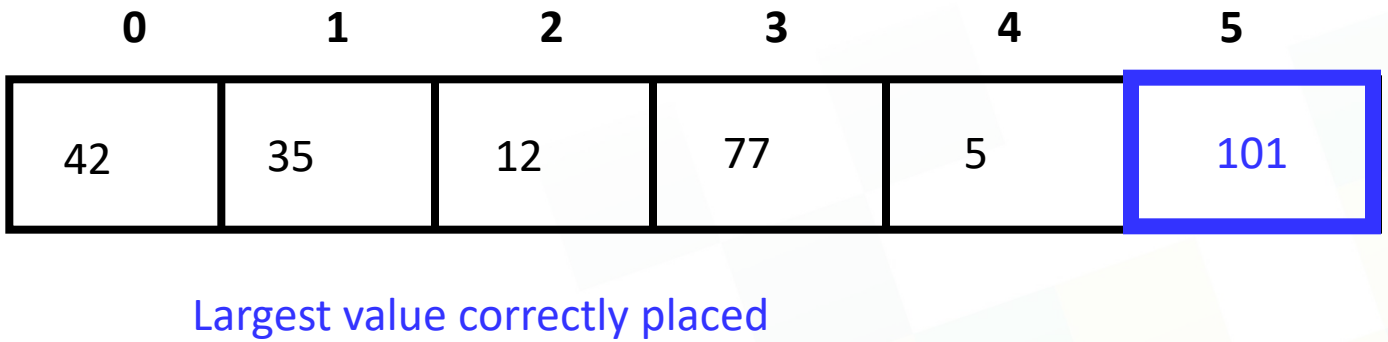
- Menelusuri kumpulan elemen
  - Bergerak dari depan sampai ke belakang (elemen terakhir/end of array)
  - “Bubble” the largest value to the end using pair-wise comparisons and swapping – algoritma ini melakukan perbandingan berpasangan terlebih dahulu kemudian melakukan pertukaran.



Largest value correctly placed

Items of Interest

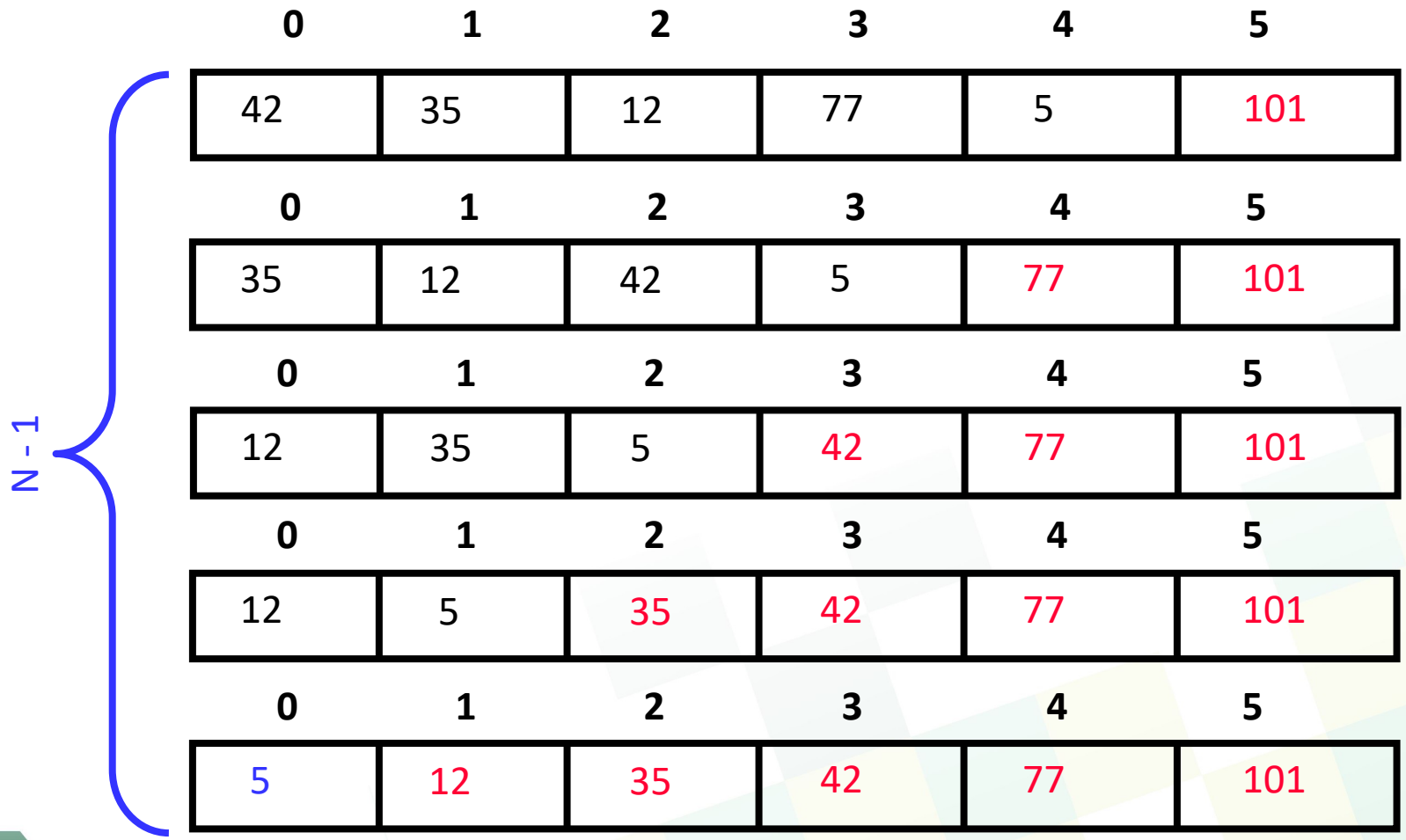
- Perhatikan bahwa hanya nilai terbesar yang ditempatkan dengan benar
- Semua nilai lainnya masih belum terurut
- Untuk itu kita perlu ulangi proses ini



Repeat “Bubble Up” How Many Times?

- Jika kita memiliki n element…
- Dan jika setiap kali kita melakukan “bubble” untuk sebuah elemen, kita menempatkannya di urutan yang benar
- Maka, proses “bubble” akan dilakukan sebanyak n-1 kali
- Dengan demikian, kita akan menempatkan semua n elemen ke urutan yang tepat

“Bubbling” All the Elements



The Selection Sort

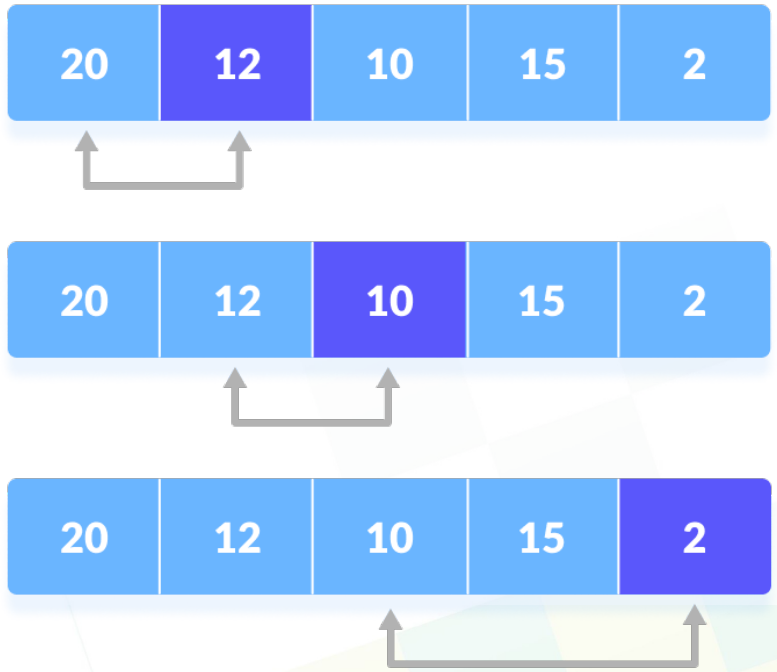
- Selection sort adalah teknik sortir yang sangat mudah karena teknik ini hanya mencari elemen terkecil di setiap pass dan menempatkannya di posisi yang benar.
- Seperti namanya, teknik selection sort pertama-tama memilih elemen terkecil dalam array dan menukarnya dengan elemen pertama dalam array.
- Selanjutnya, menukar elemen terkecil kedua dalam array dengan elemen kedua dan seterusnya. Jadi untuk setiap pass (putaran), elemen terkecil dalam array dipilih dan diletakkan pada posisi yang tepat sampai seluruh array diurutkan.
- Selection sort bekerja secara efisien ketika daftar yang akan disortir berukuran kecil tetapi kinerjanya sangat terpengaruh apabila daftar yang disortir bertambah besar. Karenanya, tidak disarankan untuk menggunakan selection sort untuk daftar (list) yang berukuran besar

How Selection Sort Works

- 1 – set elemen pertama sebagai elemen terkecil (**min**)

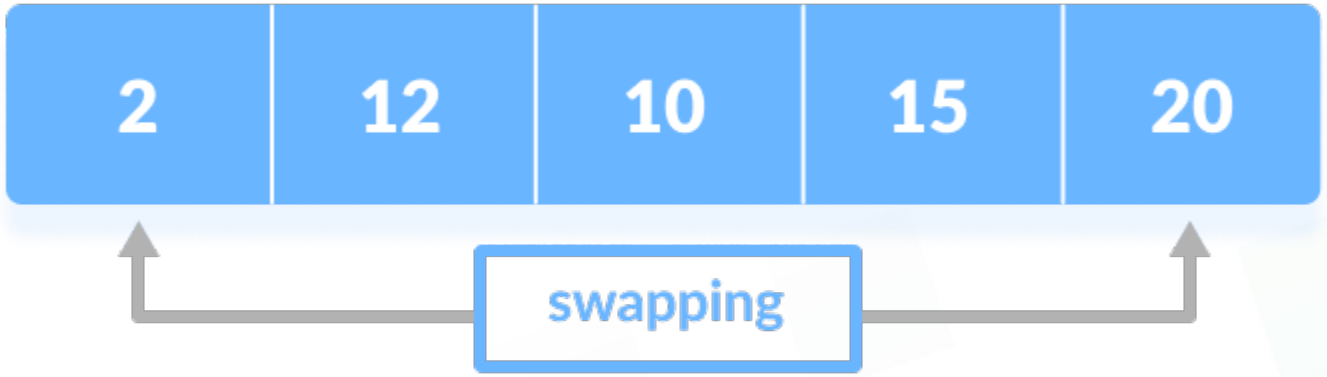
How Selection Sort Works

- 2 – bandingkan **min** dengan elemen kedua. Jika elemen kedua lebih kecil dari **min**, set elemen kedua sebagai **min**. Ulangi step ini dengan elemen selanjutnya.



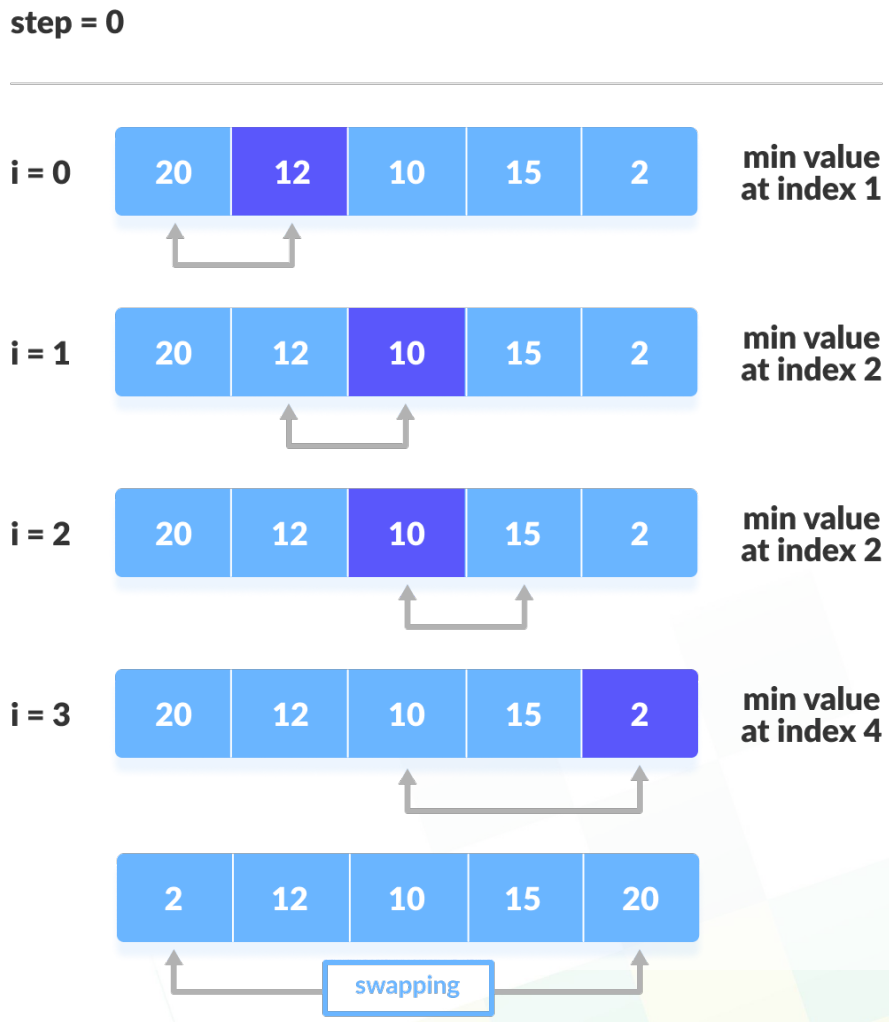
How Selection Sort Works

- 3 – setelah setiap iterasi, **min** akan ditempatkan di depan dari array yang belum terurut.



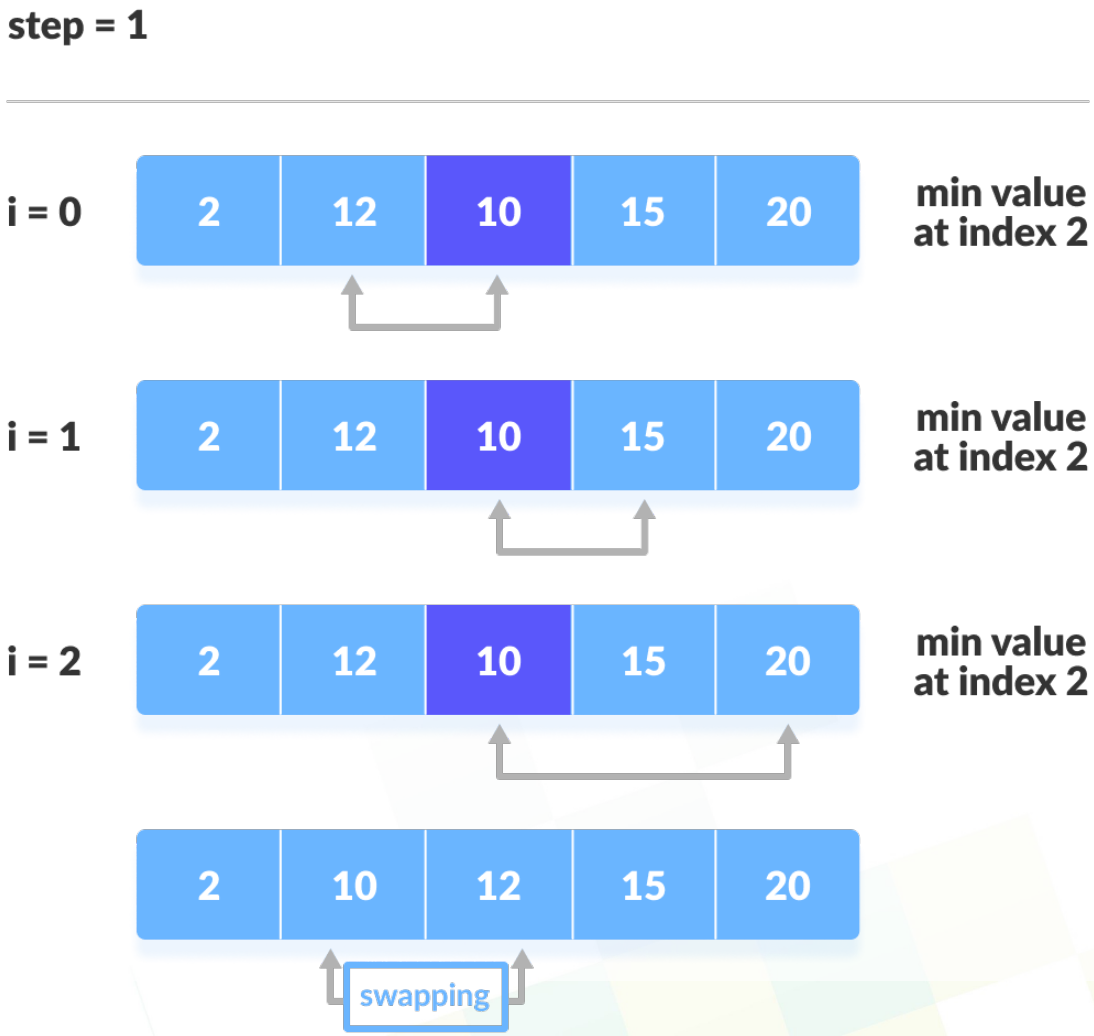
How Selection Sort Works

- Iterasi 1



How Selection Sort Works

- Iterasi 2

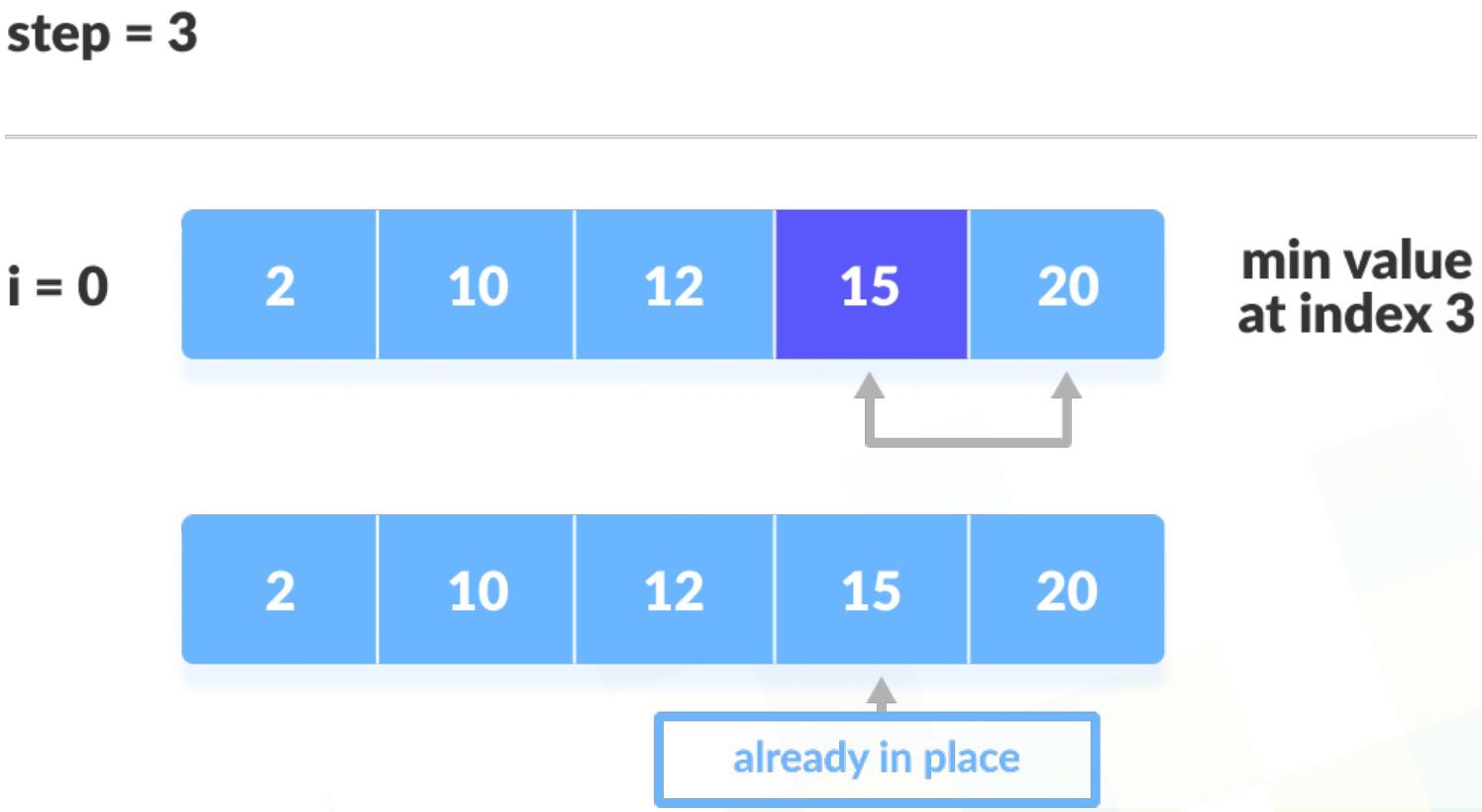


How Selection Sort Works

- Iterasi 3

How Selection Sort Works

- Iterasi 4



Proses 1

0	1	2	3	4	5
32	75	69	58	21	40

Pembanding	Posisi
32 < 75	0
32 < 69	0
32 < 58	0
32 > 21 (**tukar idx**)	4
21 < 40	4

Tukar data ke-0 (**32**) dengan data ke-4 (**21**)

0	1	2	3	4	5
21	75	69	58	32	40

Proses 2

0	1	2	3	4	5
21	**75**	69	58	32	40

Pembanding	Posisi
75 > 69 (**tukar idx**)	2
69 > 58 (**tukar idx**)	3
58 > 32 (**tukar idx**)	4
32 < 40	4

Tukar data ke-1 (**75**) dengan data ke-4 (**32**)

0	1	2	3	4	5
21	32	69	58	75	40

Proses 3

0	1	2	3	4	5
21	32	**69**	58	75	40

Pembanding	Posisi
69 > 58 (**tukar idx**)	3
58 < 75	3
58 > 40	5

Tukar data ke-2 (**69**) dengan data ke-5 (**40**)

0	1	2	3	4	5
21	32	40	58	75	69

Proses 4

0	1	2	3	4	5
21	32	40	**58**	75	69

Pembanding	Posisi
58 < 75	3
58 < 69	3

Tukar data ke-3 (**58**) dengan data ke-3 (**58**)

0	1	2	3	4	5
21	32	40	58	75	69

Proses 5

0	1	2	3	4	5
21	32	40	58	**75**	69

Pembanding	Posisi
75 > 69	5

Tukar data ke-4 (**75**) dengan data ke-5 (**69**)

0	1	2	3	4	5
21	32	40	58	69	75

FAKULTAS TEKNOLOGI INFORMASI
Universitas Advent Indonesia

Selection Sort

Algoritma Selection Sort

```
selectionSort(array, size)
      ulangi (size – 1) kali
      set elemen pertama sebagai min
      untuk setiap elemen yg belum terurut
            if elemen < min
                  set elemen sebagai min yg baru
      swap min dengan posisi elemen pertama
end selectionSort
```