INF218
Struktur Data & Algoritma

Simple Sorting:
Bubble, Selection and Insertion

Alim Misbullah, S.Si., M.S.

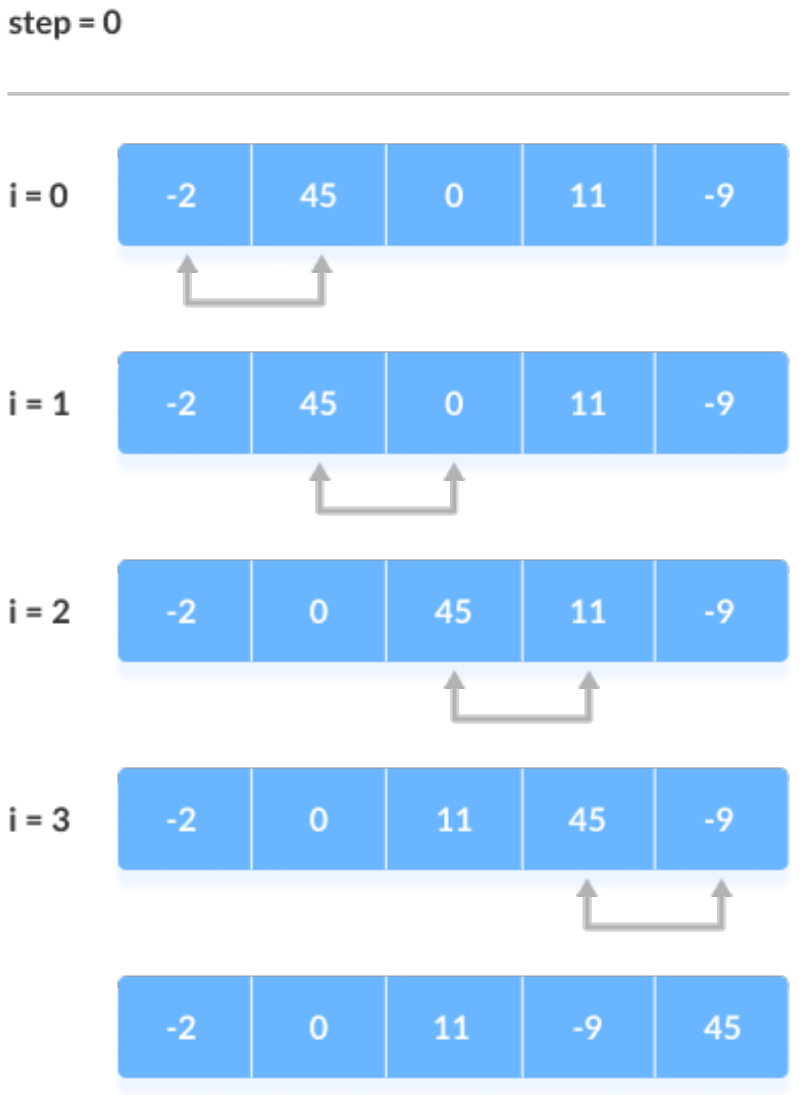
Bubble Sort

Definisi Bubble Sort

- Bubble sort adalah sebuah algoritma pengurutan yang membandingkan 2 nilai yang bersebelahan dan diganti posisinya sampai elemen tersebut terurut secara keseluruhan
- Seperti perpindahan gelembung udara di dalam air yang muncul ke permukaan
- Setiap elemen pada array akan berpindah ke bagian akhir index pada setiap iterasi
- Oleh karena itu, metode pengurutan ini disebut dengan **Bubble Sort**

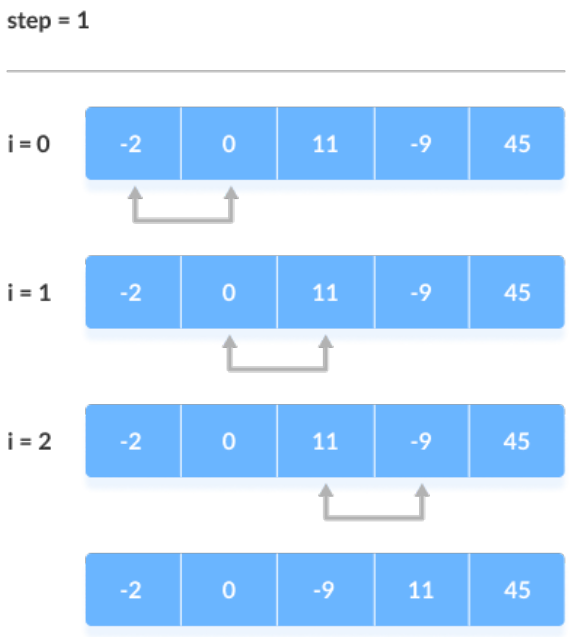
Cara Kerja Bubble Sort

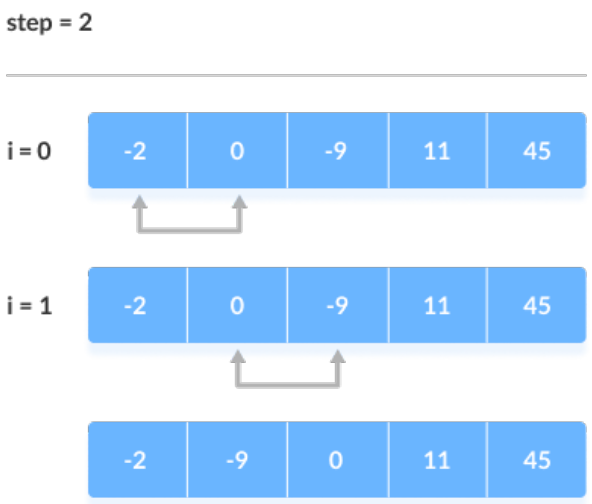
- Misalnya, kita akan melakukan pengurutan data secara **ascending order**
- Iterasi Pertama (Banding dan Tukar Posisi)
  - Mulai dari index pertama, bandingkan elemen pertama dan elemen kedua
  - Jika elemen pertama lebih besar dari elemen kedua, maka tukar posisi
  - Sekarang, bandingkan elemen kedua dan elemen ketiga. Tukar posisi jika kedua elemen tersebut tidak pada posisi yang tepat urutannya
  - Proses tersebut akan dikerjakan sampai elemen terakhir

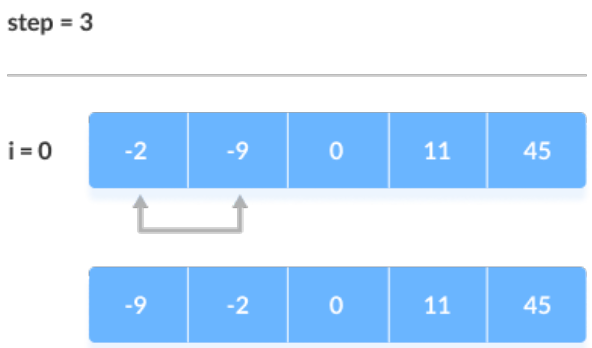


Cara Kerja Bubble Sort

- Iterasi selanjutnya







Algoritma

```
bubbleSort(array)
  for i <- 1 to indexOfLastUnsortedElement-1
    if leftElement > rightElement
      swap leftElement and rightElement
end bubbleSort
```

Implementasi Bubble Sort

- Bahasa C

  https://www.programiz.com/dsa/bubble-sort#c-code
- Bahasa Java

  https://www.programiz.com/dsa/bubble-sort#java-code
- Python

  https://www.programiz.com/dsa/bubble-sort#python-code
- C++

  https://www.programiz.com/dsa/bubble-sort#cpp-code

Optimisasi Bubble Sort

- Pada algoritma bubble sort di atas, proses perbandingan elemen tetap akan dilakukan meskipun array sudah terurut. Hal ini akan meningkatkan waktu eksekusi
- Solusinya adalah dengan menggunakan extra variable yaitu **swapped**.
- Setelah iterasi pertama, variabel swapped tersebut akan selalu bernilai salah ketika tidak ada nilai yang diganti posisinya. Namun, jika ada nilai yang ditukar posisinya, maka variable swapped akan bernilai benar. Artinya, ada nilai di array yang belum terurut
- Jika, tidak ada perubahan maka tidak perlu dilakukan iterasi berikutnya dan array sudah terurut

Optimisasi Bubble Sort

- Algoritma Optimized Bubble Sort

```
bubbleSort(array)
  swapped <- false
  for i <- 1 to indexOfLastUnsortedElement-1
    if leftElement > rightElement
      swap leftElement and rightElement
      swapped <- true
end bubbleSort
```

- Bahasa C
  https://www.programiz.com/dsa/bubble-sort#c-code
- Bahasa Java
  https://www.programiz.com/dsa/bubble-sort#java-code
- Python
  https://www.programiz.com/dsa/bubble-sort#python-code
- C++
  https://www.programiz.com/dsa/bubble-sort#cpp-code

Bubble Sort Complexity

Time Complexity	
Best	O(n)
Worst	$O(n^2)$
Average	$O(n^2)$
Space Complexity	O(1)
Stability	Yes

Bubble Sort Complexity

- Time Complexity
  - Worst Case Complexity
    - Jika kita ingin mengurutkan dalam bentuk ascending order dan arraynya dalam bentuk descending order, maka worst case complexity akan terjadi
  - Average Case Complexity
    - Akan terjadi ketika elemen sebuah array tersusun secara bercamput (bukan ascending atau descending)
  - Best Case Complexity
    - Jika array sudah terurut, maka tidak perlu dilakukan pengurutan kembali
- Space Complexity
  - Space complexity adalah O(1) karena sebuah variable extra digunakan untuk pertukaran nilai
  - Pada Optimized Bubble Sort, dua extra variable digunakan sehingga space complexity menjadi O(2)

Bubble Sort Application

- Bubble Sort digunakan jika
  - Kompleksitas pengurutan bukan menjadi sebuah pertimbangan
  - Kode program yang singkat dan sederhana menjadi sebuah pertimbangan

Selection Sort

Definisi Selection Sort

- Selection Sort merupakan sebuah algoritma pengurutan yang memilih nilai terkecil dari sebuah array yang belum terurut dalam setiap iterasi dan menempatkan nilai terkecil tersebut pada bagian awal dari array yang belum terurut (**jika perngurutannya adalah ascending**)
- Algoritma selection sort akan mengatur 2 subarray di dalam sebuah array yang akan diurutkan
  - Subarray yang sudah terurut
  - Subarray yang belum terurut
- Pada setiap iterasi, elemen terkecil akan diambil dari subarray yang belum terurut dan akan dipindahkan ke bagian subarray yang sudah terurut

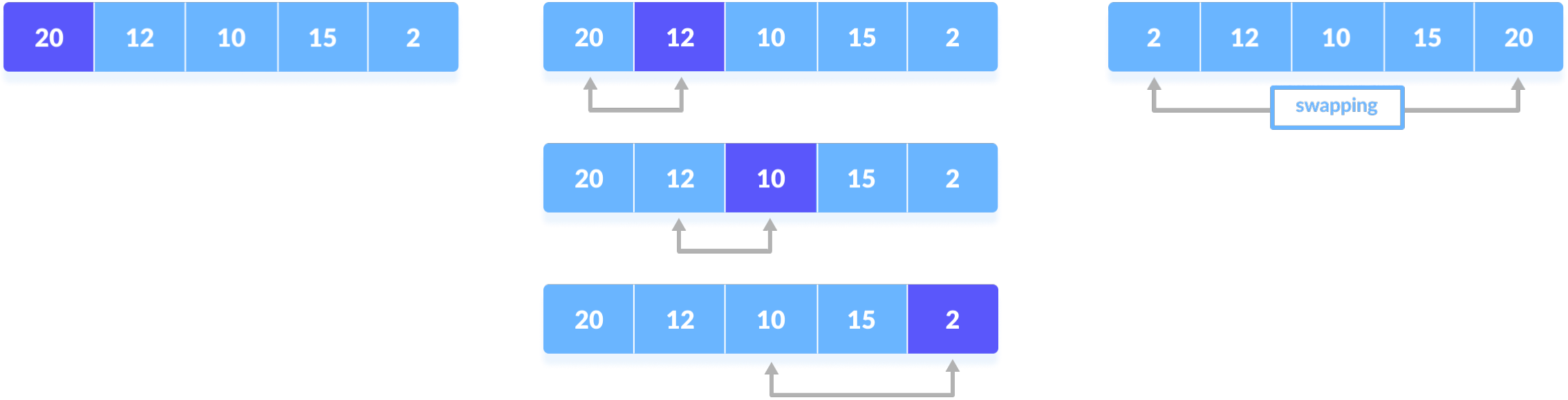
Definisi Selection Sort

```
arr[] = 64 25 12 22 11

// Find the minimum element in arr[0...4]
// and place it at beginning
11 25 12 22 64

// Find the minimum element in arr[1...4]
// and place it at beginning of arr[1...4]
11 12 25 22 64

// Find the minimum element in arr[2...4]
// and place it at beginning of arr[2...4]
11 12 22 25 64

// Find the minimum element in arr[3...4]
// and place it at beginning of arr[3...4]
11 12 22 25 64
```

Cara Kerja Selection Sort

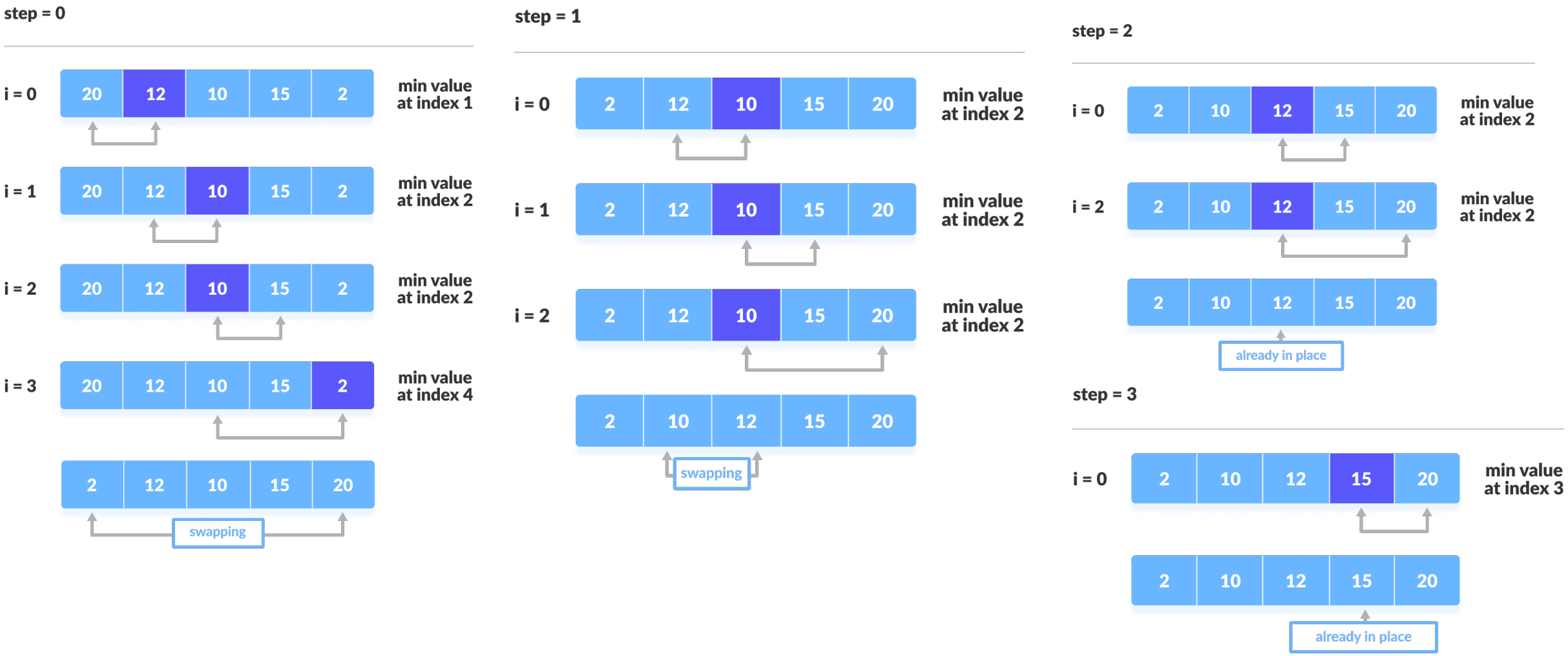
- Jadikan elemen pertama dari sebuah array sebagai **minimum**
- Bandingkan **minimum** dengan elemen kedua. Jika elemen kedua lebih kecil dari **minimum**, maka jadikan elemen kedua sebagai **minimum**
- Bandingkan elemen ketiga dengan **minimum**. Jika elemen ketiga lebih kecil dari **minimum**, maka jadikan elemen ketiga sebagai **minimum**
- Proses tersebut akan berlangsung sampai elemen terakhir dari sebuah array
- Setelah sebuah iterasi selesai, maka elemen terkecil akan ditempatkan pada bagian awal dari array yang belum terurut

Cara Kerja Selection Sort




```
selectionSort(array, size)
  repeat (size - 1) times
  set the first unsorted element as the minimum
  for each of the unsorted elements
    if element < currentMinimum
      set element as new minimum
  swap minimum with first unsorted position
end selectionSort
```

Cara Kerja Selection Sort



Implementasi Selection Sort

- Bahasa C

  https://www.programiz.com/dsa/selection-sort#c-code
- Bahasa Java

  https://www.programiz.com/dsa/selection-sort#java-code
- Python

  https://www.programiz.com/dsa/selection-sort#python-code
- C++

  https://www.programiz.com/dsa/selection-sort#cpp-code

Selection Sort Complexity

Time Complexity	
Best	$O(n^2)$
Worst	$O(n^2)$
Average	$O(n^2)$
Space Complexity	$O(1)$
Stability	No

Selection Sort Complexity

- Time Complexity
  - Worst Case Complexity
    - Jika kita ingin mengurutkan dalam bentuk ascending order dan arraynya dalam bentuk descending order, maka worst case complexity akan terjadi
  - Average Case Complexity
    - Akan terjadi ketika elemen sebuah array tersusun secara bercamput (bukan ascending atau descending)
  - Best Case Complexity
    - Jika array sudah terurut, maka tidak perlu dilakukan pengurutan kembali
- Space Complexity
  - Space complexity adalah O(1) karena sebuah variable extra digunakan untuk pertukaran nilai

Insertion Sort

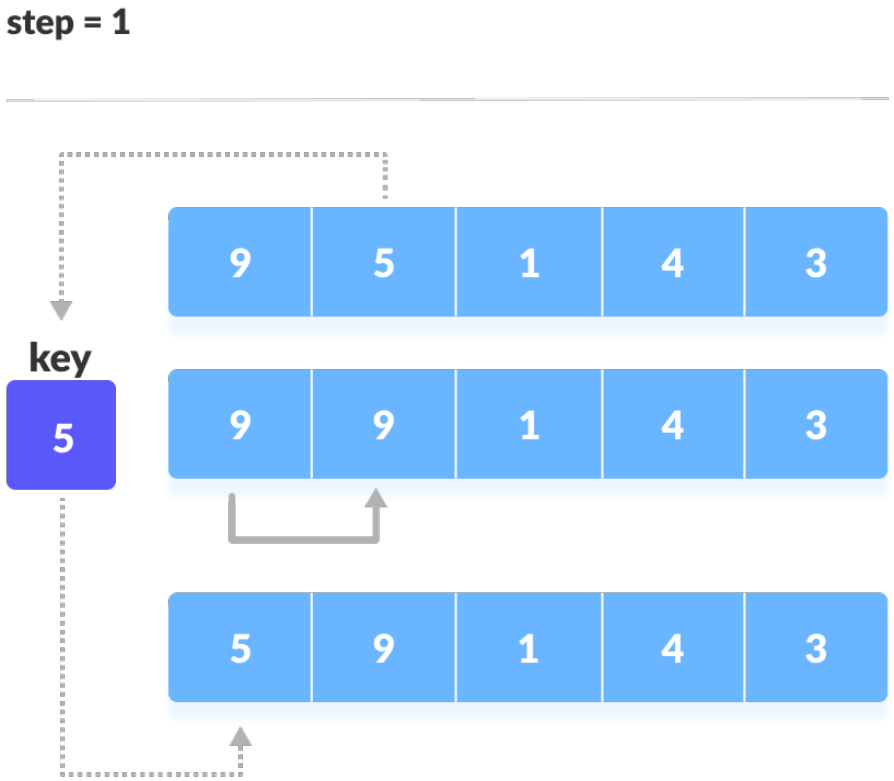
Definisi Insertion Sort

- Insertion sort adalah sebuah algoritma pengurutan yang menempatkan sebuah elemen yang belum terurut pada tempat yang sesuai di setiap iterasi
- Insertion sort bekerja seperti mengurutkan kartu di tangan pada sebuah permainan kartu
- Kita mengasumsikan bahwa kartu pertama sudah terurut, kemudian kita memilih sebuah kartu yang belum terurut. Jika yang dipilih tersebut lebih besar dari kartu yang sudah terurut, maka kita tempatkan disebelah kanan (pada kasus ascending). Namun, jika lebih kecil maka kartu tersebut ditempatkan diposisi sebelah kiri kartu yang sudah terurut
- Cara yang sama juga digunakan pada algoritma **Insertion Sort**

Cara Kerja Insertion Sort

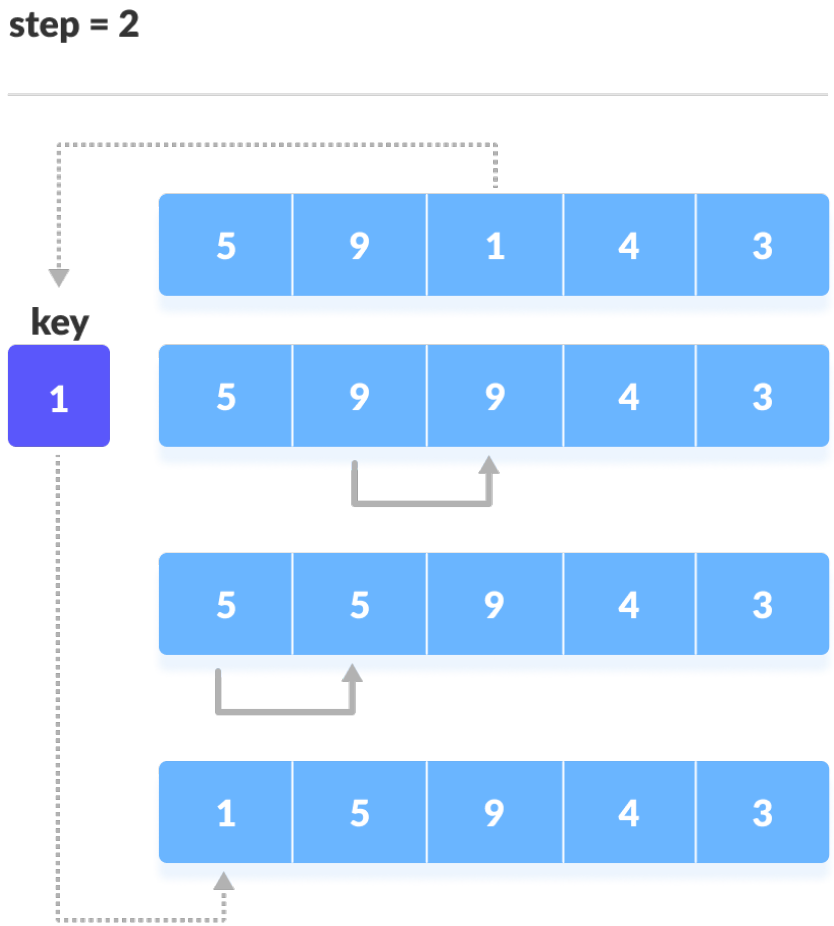
- Misalnya kita ingin mengurutkan array berikut

- Elemen pertama di dalam array diasumsikan sudah terurut
- Ambil elemen kedua dan tempatkan secara terpisah pada sebuah variable **key**
- Bandingkan **key** dengan elemen pertama. Jika elemen pertama lebih besar dari **key**, maka **key** ditempatkan tepat di depan elemen pertama



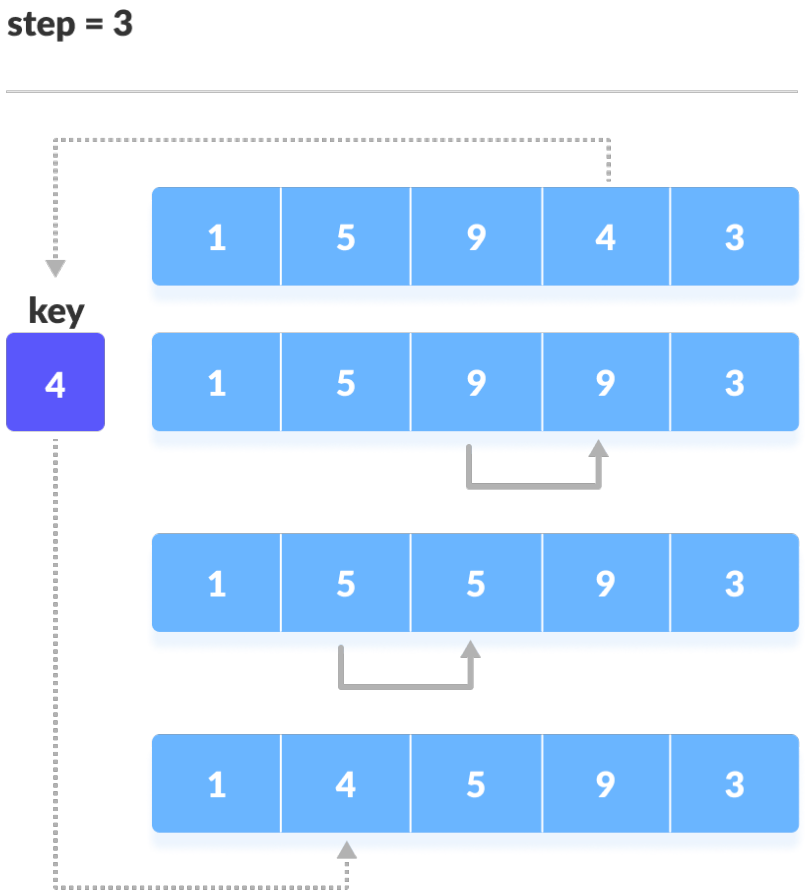
Cara Kerja Insertion Sort

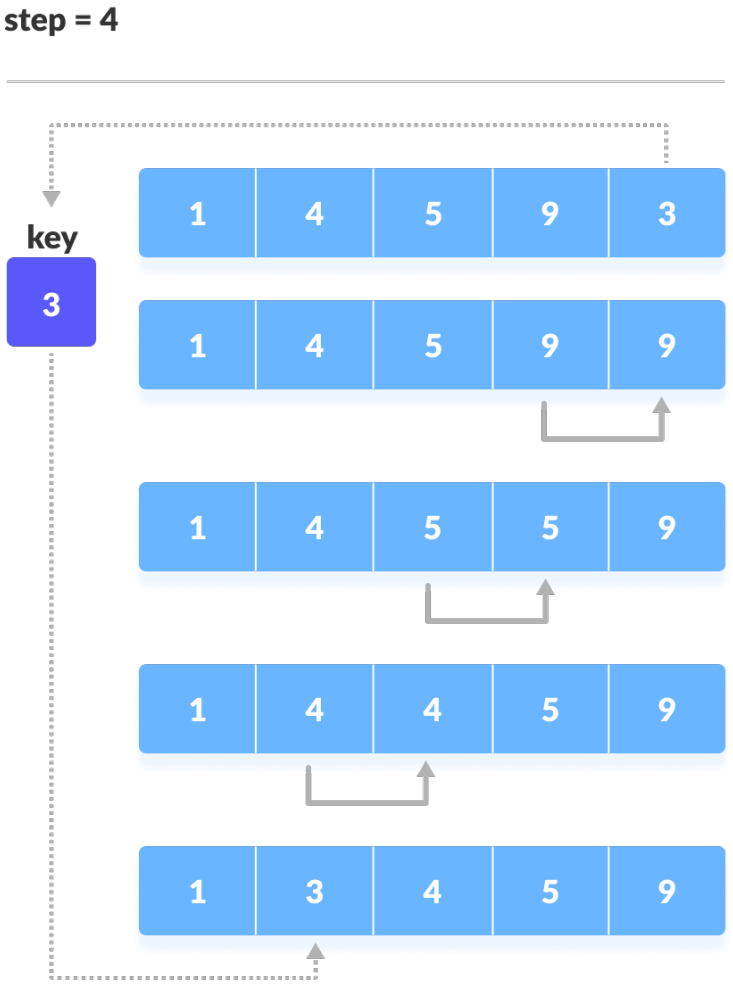
- Sekarang, kedua elemen sudah terurut
- Selanjutnya, bandingkan elemen ketiga dengan elemen sebelah kirinya. Tempatkan elemen yang lebih kecil dibagian kirinya
- Jika tidak ada elemen yang lebih kecil, maka tempatkan elemen tersebut di index pertama array



Cara Kerja Insertion Sort

- Seperti pada langkah sebelumnya, tempatkan elemen yang belum terurut pada posisi yang sesuai





Algoritma Insertion Sort

```
insertionSort(array)
  mark first element as sorted
  for each unsorted element X
    'extract' the element X
    for j <- lastSortedIndex down to 0
      if current element j > X
        move sorted element to the right by 1
    break loop and insert X here
end insertionSort
```

Implementasi Selection Sort

- Bahasa C

  https://www.programiz.com/dsa/insertion-sort#c-code
- Bahasa Java

  https://www.programiz.com/dsa/insertion-sort#java-code
- Python

  https://www.programiz.com/dsa/insertion-sort#python-code
- C++

  https://www.programiz.com/dsa/insertion-sort#cpp-code

Insertion Sort Complexity

Time Complexity	
Best	$O(n)$
Worst	$O(n^2)$
Average	$O(n^2)$
Space Complexity	$O(1)$
Stability	Yes

Insertion Sort Complexity

- Time Complexity
  - Worst Case Complexity
    - Jika kita ingin mengurutkan dalam bentuk ascending order dan arraynya dalam bentuk descending order, maka worst case complexity akan terjadi
  - Average Case Complexity
    - Akan terjadi ketika elemen sebuah array tersusun secara bercamput (bukan ascending atau descending)
  - Best Case Complexity
    - Jika array sudah terurut, maka tidak perlu dilakukan pengurutan kembali
- Space Complexity
  - Space complexity adalah O(1) karena sebuah variable extra digunakan untuk pertukaran nilai

Q & A

Referensi

- https://www.programiz.com/dsa/bubble-sort
- https://www.programiz.com/dsa/selection-sort
- https://www.programiz.com/dsa/insertion-sort