UNIVERSITAS
PENDIDIKAN INDONESIA

Rekayasa
Perangkat
Lunak

Python Programming

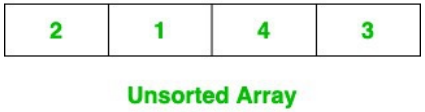
Array Sorting

01

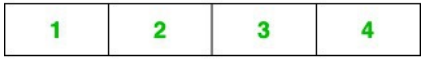
Pengurutan (Sorting)

Pendahuluan Sorting

- Pengurutan (sorting) merupakan hal yang penting saat memanipulasi data.
- Pengurutan diartikan sebagai penataan data dalam urutan tertentu.
- Ketika data diurutkan, maka data akan terlihat rapi dan mudah dibaca. Sehingga memudahkan untuk dianalisa.
- Saat sorting data dilakukan, program akan mengkalkulasi dan membandingkan setiap data agar dapat menemukan mana data yang terbesar dan terkecil.
- Algoritma sorting memungkinkan kita untuk menyusun elemen-elemen data dalam urutan tertentu, yang dapat meningkatkan efisiensi pencarian dan analisis data.
- Pemilihan algoritma sorting juga mempengaruhi cepat atau lambatnya sistem dalam memanipulasi data.
- Beberapa contoh penyortiran dalam kehidupan nyata adalah: buku telepon, kamus, dll.



Unsorted Array



Array sorted in ascending order

Array sorted in descending order

Jenis Pengurutan

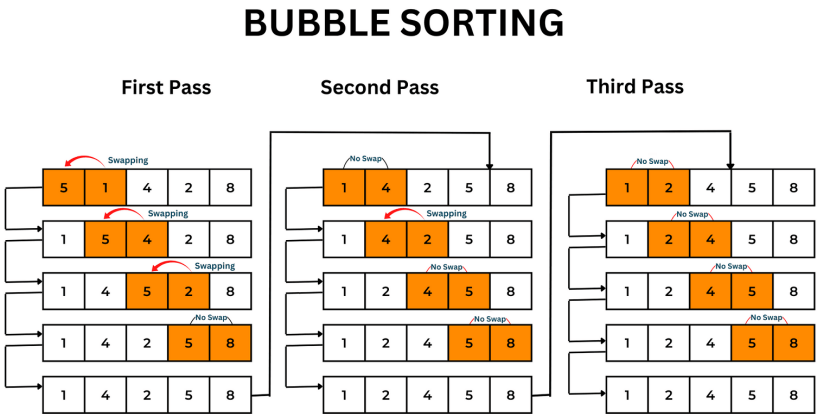
Beberapa jenis pengurutan:

- **Increasing order:** Ketika setiap elemen yang berurutan lebih besar dari elemen sebelumnya. Contoh: 1, 2, 3, 4, 5.
- **Decreasing order:** Jika elemen berikutnya selalu lebih kecil dari elemen sebelumnya. Contoh: 5, 4, 3, 2, 1.
- **Non-Increasing Order:** Jika setiap elemen ke-i yang ada dalam barisan tersebut lebih besar dari atau sama dengan elemen ke-i (i-1). Urutan ini terjadi setiap kali ada angka yang diulang. Contoh: 1, 2, 2, 3, 4, 5. Disini angka 2 diulang sebanyak dua kali.
- **Non-Decreasing Order:** Jika setiap elemen ke-i yang ada dalam barisan tersebut kurang dari atau sama dengan elemen ke-i (i-1). Urutan ini terjadi setiap kali ada angka yang diulang. Contoh: 5, 4, 3, 2, 2, 1. Disini angka 2 diulang sebanyak dua kali.

02
Algoritma Sorting: Bubble Sort

Bubble Sort

- Merupakan algoritma sorting paling populer dan sederhana dibandingkan dengan algoritma lainnya
- Proses pengurutan pada algoritma ini dilakukan **dengan membandingkan masing - masing elemen secara berpasangan lalu menukarnya dalam kondisi tertentu**.
- Proses ini akan terus diulang sampai elemen terakhir atau sampai tidak ada lagi elemen yang dapat ditukar.
- Algoritma ini diberi nama “Bubble”, karena dimana gelembung yang terbesar akan naik ke atas.



Contoh Bubble Sort

```python
def bubbleSort(array):
    n = len(array) # jumlah elemen array
    # perulangan pertama
    for i in range(n):
        # perulangan kedua
        for j in range(n - i - 1):
            # bandingkan masing" elemen
            if array[j] > array[j + 1]:
                # jika lebih besar, tukar.
                array[j], array[j + 1] = array[j + 1], array[j]
    return array
```

✓ 0.0s

```python
unordered = [5, 3, 4, 8, 1, 2, 0, 9, 6, 7]
print(bubbleSort(unordered))
```

✓ 0.0s

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Contoh Bubble Sort

- **def bubbleSort(array):** Deklarasi function yang disebut bubbleSort, yang menerima satu parameter yaitu array. Function ini akan mengurutkan elemen-elemen dalam array menggunakan algoritma bubble sort.
- **n = len(array):** Menghitung jumlah elemen dalam array dan menyimpannya dalam variabel n. Variabel n akan digunakan untuk menentukan jumlah iterasi dalam perulangan.
- **for i in range(n):** Perulangan luar yang akan berjalan sebanyak jumlah elemen dalam array. Setiap iterasi dari perulangan luar ini akan mengurutkan satu elemen terbesar ke posisi yang tepat.
- **for j in range(n - i - 1):** Perulangan dalam yang akan berjalan untuk setiap elemen dalam array, kecuali elemen yang sudah terurut pada iterasi sebelumnya. Hal ini dikarenakan, setelah iterasi pertama, elemen terbesar sudah pasti berada di posisi paling akhir, sehingga tidak perlu dibandingkan lagi.
- **if array[j] > array[j + 1]:** Kondisi untuk memeriksa apakah elemen saat ini lebih besar dari elemen berikutnya. Jika ya, maka kedua elemen tersebut akan ditukar.
- **array[j], array[j + 1] = array[j + 1], array[j]:** Bertugas untuk menukar posisi dua elemen jika kondisi pada langkah sebelumnya terpenuhi. Penggunaan tuple unpacking memungkinkan pertukaran nilai tanpa menggunakan variabel sementara.

Langkah Bubble Sort

Alur Kode:

1. Kita hitung jumlah list menggunakan len(array) sebagai parameter perulangan.
2. Buat dua perulangan untuk membandingkan elemen pada list.
3. Bandingkan elemen pertama dengan elemen kedua menggunakan if.
4. Jika elemen pertama lebih besar daripada elemen kedua maka tukar posisinya.

```
if array[j] > array[j + 1]:
    # jika lebih besar, tukar.
    array[j], array[j + 1] = array[j + 1], array[j]
```

5. Jika elemen kedua lebih besar dari pada elemen pertama maka biarkan saja.
6. Langkah 3 - 5 diulang sampai elemen terakhir (atau perulangan selesai).
7. return array: Setelah seluruh proses pengurutan selesai, fungsi mengembalikan array yang sudah diurutkan.

Bubble Sort

Bagaimana jika kita ingin mengurutkan secara descending (besar ke kecil)?

```python
def bubbleSort(array):
    n = len(array) # jumlah elemen array
    # perulangan pertama
    for i in range(n):
        # perulangan kedua
        for j in range(n - i - 1):
            # bandingkan masing" elemen
            if array[j] < array[j + 1]:
                # jika lebih besar, tukar.
                array[j], array[j + 1] = array[j + 1], array[j]
    return array
```

✓ 0.0s

```python
unordered = [5, 3, 4, 8, 1, 2, 0, 9, 6]
print(bubbleSort(unordered))
```

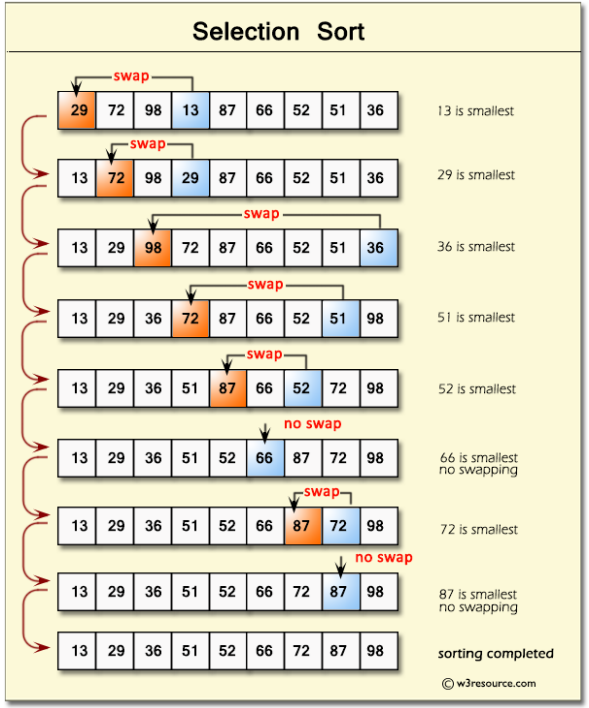
✓ 0.0s

```
[9, 8, 6, 5, 4, 3, 2, 1, 0]
```

03
Algoritma Sorting: Selection Sort

Selection Sort

- Selection Sort adalah algoritma sorting yang mengurutkan data dengan cara mencari elemen paling kecil dari array, lalu menukar elemen tersebut ke urutan paling awal.
- Algoritma ini memiliki konsep yang sama dengan bubble sort, yaitu **membandingkan dan menukar**.
- Tetapi, dalam selection sort ia akan **mencari index dengan elemen paling kecil**, ketika sudah ketemu, **elemen pada index itu akan ditukar dengan elemen pada index pertama**.
- Begitu seterusnya sampai perulangan selesai atau tidak ada lagi elemen yang bisa ditukar.



Contoh Selection Sort

```python
def selectionSort(arr):
    n = len(arr)
    # perulangan list elemen
    for i in range(n):
        # cari nilai elemen terendah
        # yang masih tersedia
        min_idx = i
        for j in range(i+1, n):
            if arr[min_idx] > arr[j]:
                min_idx = j

        # tukar dengan nilai elemen ke-i
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr
```

✓ 0.0s

```python
unordered = [5, 3, 4, 8, 1, 2, 0, 9, 6, 7]
print(selectionSort(unordered))
```

✓ 0.0s

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Contoh Selection Sort

- **def selectionSort(arr):** deklarasi function yang disebut selectionSort, yang menerima satu parameter yaitu arr. Fungsi ini akan mengurutkan elemen-elemen dalam array menggunakan algoritma selection sort.
- **n = len(arr):** Ini menghitung jumlah elemen dalam array dan menyimpannya dalam variabel n. Variabel n akan digunakan untuk menentukan jumlah iterasi dalam perulangan.
- **for i in range(n):** perulangan luar yang akan berjalan sebanyak jumlah elemen dalam array. Setiap iterasi dari perulangan luar ini akan menempatkan elemen terkecil yang belum terurut pada posisi yang benar.
- **min_idx = i:** variabel yang menyimpan indeks elemen terkecil yang sudah ditemukan. Pada awal setiap iterasi luar, diasumsikan bahwa elemen terkecil adalah elemen pada indeks i.
- **for j in range(i+1, n):** perulangan dalam yang akan mencari elemen terkecil yang masih tersedia, yaitu elemen pada indeks j yang lebih besar dari i.
- **if arr[min_idx] > arr[j]:** kondisi untuk memeriksa apakah elemen di min_idx lebih besar dari elemen pada j. Jika ya, maka min_idx akan diupdate menjadi j.
- **arr[i], arr[min_idx] = arr[min_idx], arr[i]:** setelah menemukan elemen terkecil, dilakukan pertukaran dengan elemen pada indeks i. Ini menempatkan elemen terkecil pada posisi yang benar.
- **return arr:** setelah seluruh proses pengurutan selesai, fungsi mengembalikan array yang sudah diurutkan.

Langkah Selection Sort

Alur Kode:

1. Kita cari dulu jumlah elemen pada list dengan len(arr).
2. Lalu lakukan perulangan pertama yang didalamnya terdapat kode untuk mencari nilai minimum dan menukar nilai.
3. min_idx berperan untuk menampung index dengan nilai terendah.
4. Lalu pada perulangan kedua, setiap elemen akan terus dibandingkan menggunakan if untuk mendapatkan index dengan nilai terkecil.

```
for j in range(i+1, n):
    if arr[min_idx] > arr[j]:
        min_idx = j
```

5. Pada perulangan kedua, semua elemen setelah elemen ke-i atau i+1, saling dibandingkan untuk mencari nilai terkecil.
6. Setelah menemukan elemen dengan nilai terkecil, index tersebut ditukar dengan nilai ke-i.

```
arr[i], arr[min_idx] = arr[min_idx], arr[i]
```

7. Hal tersebut diulang sampai perulangan pertama selesai (atau tidak elemen tuk diulang).

Selection Sort

Bagaimana jika kita ingin mengurutkan secara descending (besar ke kecil)?

```python
def selectionSort(arr):
    n = len(arr)
    # perulangan list elemen
    for i in range(n):
        # cari nilai elemen terendah
        # yang masih tersedia
        min_idx = i
        for j in range(i+1, n):
            if arr[min_idx] < arr[j]:
                min_idx = j

        # tukar dengan nilai elemen ke-i
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr
```

✓ 0.0s

```python
unordered = [5, 3, 4, 8, 1, 2, 0, 9, 6, 7]
print(selectionSort(unordered))
```

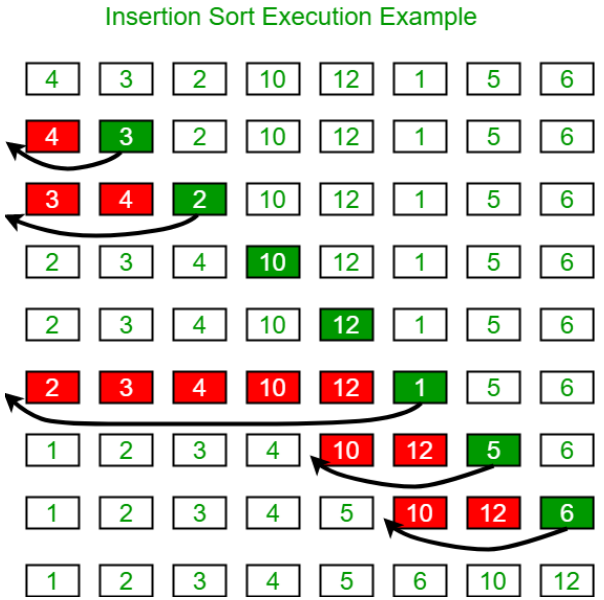
✓ 0.0s

```
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

04
Algoritma Sorting: Insertion Sort

Insertion Sort

- Merupakan algoritma yang melakukan pengurutan dengan **membandingkan elemen satu dengan elemen lainnya dalam sebuah array**.
- Elemen yang dibandingkan akan **ditempatkan ke posisi yang sesuai (urut) pada array**.
- Analoginya seperti mengurutkan kumpulan kartu. Setiap kartu yang diambil, dibandingkan terlebih dahulu ke kumpulan kartu yang sudah diurutkan. Dan ketika tahu urutan ke berapa, selipkan kartu itu ke tumpukan kartu agar terurut.



Contoh Insertion Sort

```python
def insertionSort(array):
    # perulangan pertama
    for i in range(1, len(array)):
        # ini elemen yang akan kita posisikan
        key_item = array[i]
        # kunci index posisi
        j = i - 1
        # lakukan perulangan kedua
        while j >= 0 and array[j] > key_item:
            # menggeser elemen yang lain
            array[j + 1] = array[j]
            j -= 1
        # memposisikan elemen
        array[j + 1] = key_item
    return array
```

✓ 0.0s

```python
unordered = [5, 3, 4, 8, 1, 2, 0, 9, 6, 7]
print(insertionSort(unordered))
```

✓ 0.0s

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Contoh Insertion Sort

- **def insertionSort(array):** deklarasi fungsi yang disebut insertionSort, yang menerima satu parameter yaitu array. Fungsi ini akan mengurutkan elemen-elemen dalam array menggunakan algoritma insertion sort.
- **for i in range(1, len(array)):** perulangan pertama yang akan berjalan mulai dari indeks kedua [1] (indeks pertama dianggap sudah terurut). Setiap iterasi dari perulangan ini akan memproses satu elemen untuk ditempatkan pada posisi yang benar.
- **key_item = array[i]:** elemen yang akan ditempatkan pada posisi yang benar. Elemen ini akan dibandingkan dengan elemen-elemen di bagian terurut dari array.
- **j = i - 1:** j adalah variabel yang menyimpan indeks posisi saat ini dari elemen yang akan dipindahkan. Ini dimulai dari indeks sebelum elemen yang akan ditempatkan.
- **while j >= 0 and array[j] > key_item:** perulangan kedua yang akan berjalan selama elemen di posisi j lebih besar dari key_item dan j tidak mencapai indeks negatif. Perulangan ini bertujuan untuk mencari posisi yang benar untuk key_item.
- **array[j + 1] = array[j]:** langkah untuk menggeser elemen ke kanan jika elemen di posisi j lebih besar dari key_item.
- **j -= 1:** Mengurangi nilai j agar perulangan dapat berlanjut ke posisi sebelumnya dalam array yang sudah terurut.
- **array[j + 1] = key_item:** Setelah menemukan posisi yang benar, key_item ditempatkan pada posisi tersebut.
- **return array:** Setelah seluruh proses pengurutan selesai, fungsi mengembalikan array yang sudah diurutkan.

Langkah Insertion Sort

Alur Kode:

1. Kita mulai dari perulangan dari elemen kedua (1) sampai elemen terakhir. Kenapa tidak dari elemen pertama? karena elemen pertama adalah elemen yang dibandingkan oleh elemen yang lain.
2. Variabel key_item, adalah tempat untuk menampung elemen yang akan diposisikan.
3. Sedangkan variabel j dengan nilai i - 1 adalah tujuan index yang nantinya elemen key_item akan ditempatkan.
4. Menggunakan perulangan while yang memiliki kondisi selama j >= 0 dan elemen index j lebih besar dari key_item, Maka elemen yang lain akan digeser dan index j diperbarui.

```
9       while j >= 0 and array[j] > key_item:
10          # menggeser elemen yang lain
11          array[j + 1] = array[j]
12          j -= 1
```

5. Setelah tidak ada lagi elemen yang lebih besar dari key_item, maka perulangan while akan berhenti.
6. Lalu tempatkan elemen pada key_item ke index j + 1. Kenapa j + 1? karena sebelumnya kita memberikan nilai i - 1 yang seharusnya tidak sesuai.

Insertion Sort

Bagaimana jika kita ingin mengurutkan secara descending (besar ke kecil)?

```python
def insertionSort(array):
    # perulangan pertama
    for i in range(1, len(array)):
        # ini elemen yang akan kita posisikan
        key_item = array[i]
        # kunci index posisi
        j = i - 1
        # lakukan perulangan kedua
        while j >= 0 and array[j] < key_item:
            # menggeser elemen yang lain
            array[j + 1] = array[j]
            j -= 1
        # memposisikan elemen
        array[j + 1] = key_item
    return array
```

✓ 0.0s

```python
unordered = [5, 3, 4, 8, 1, 2, 0, 9, 6, 7]
print(insertionSort(unordered))
```

✓ 0.0s

```
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

05

Studi Kasus

Studi Kasus 1 (Kelompok)

- Buatlah kelompok yang terdiri dari 3-4 orang
- Pilihlah satu dari algoritma sorting di bawah ini dan buat penjelasan serta contoh kodingannya:
  - Quick sort
  - Merge sort
  - Tim sort
  - Tree sort
  - Radix sort
  - Heap sort
  - Bucket sort
  - Counting sort
  - Cube sort, dll.
- Presentasikan di depan kelas

Studi Kasus 2 (Individu)

- Dari algoritma yang telah anda pilih pada studi kasus 1, buatlah perbandingan running program (execution time) dengan 2 algoritma yang ada di slide (pilih: bubble sort, insertion sort, atau selection sort).
- Buatlah dengan menggunakan array acak berikut:

[7, 1, 36, 26, 63, 93, 55, 16, 19, 38, 74, 65, 18, 59, 8, 43, 24, 79, 49, 35, 23, 78, 51, 2, 46, 28, 60, 76, 10, 85, 66, 29, 82, 58, 69, 75, 48, 100, 5, 32, 40, 33, 34, 90, 81, 42, 57, 44, 41, 77]

- Tentukan manakah dari ke 3 algoritma tersebut yang paling cepat?
- Kumpulkan file dalam bentuk copy kodingan dan screenshot hasil program dan simpan ke dalam file dengan format .pdf

UNIVERSITAS
PENDIDIKAN INDONESIA

Rekayasa
Perangkat
Lunak

Terimakasih