FTI

TEKNIK ELEKTRO

EIE5023

ALGORITMA DAN STRUKTUR DATA + PRAK

9

ENDANG SRI RAHAYU

Outlines:

SORTING:

- Selection Sort
- Buble Sort
- Insertion Sort
- Quick Sort

Endang Sri Rahayu

ALGORITMA DAN STRUKTUR DATA

Selection Sort

Ide dasar :

- Menemukan elemen terkecil dari list dan meletakkannya pada posisi pertama, dilanjutkan dengan mencari elemen terkecil berikutnya dan diletakkan di posisi ke dua. Proses terus dilanjutkan sampai posisi terakhir terisi.

Membandingkan 2 elemen, jika sudah sesuai lanjutkan proses perbandingan, jika tidak sesuai tukar empat.

Contoh :

Data yang belum terurut

A[1] : 5 A[2] : 13 A[3] : -2 A[4] : 10 A[5] : 2

Pass No. 1 : i =1

	Comparison 1 (j=2)	Comparison 2 (j=3)	Comprison 3 (j=4)	Comparison 4 (j=5)	
sorted					
A[1]	5	5	-2	-2	-2
unsorted					
A[2]	13	13	13	13	13
A[3]	-2	-2	5	5	5
A[4]	10	10	10	10	10
A[5]	2	2	2	2	2

Pass No. 2 : i =2

	Comparison 1 (j=3)	Comparison 2 (j=4)	Comparison 3 (j=5)	
sorted				
A[1]	-2	-2	-2	-2
A[2]	13	5	5	2
unsorted				
A[3]	5	13	13	13
A[4]	10	10	10	10
A[5]	2	2	2	5

Pass No. 3 : i =3

	Comparison 1 (j=4)	Comparison 2 (j=5)	
sorted			
A[1]	-2	-2	-2
A[2]	2	2	2
A[3]	13	10	5
unsorted			
A[4]	10	13	13
A[5]	5	5	10

Pass No. 4 : i=3

	Comparison 1 (j=4)	
sorted		
A[1]	-2	-2
A[2]	2	2
A[3]	5	5
A[4]	13	10
unsorted		
A[5]	10	13

Algoritma diatas dapat dijelaskan sbb. :

1. Buat 4 pass pada array dengan i = 1, 2, 3, 4
2. Untuk i = 1 (pass = 1), buat perbandingan dengan j = 2, 3, 4, 5

   A[1] dengan A[2]; A[1] dengan A[3]; A[1] dengan A[4]; A[1] dengan A[5]
3. Untuk i = 2 (pass = 2), buat perbandingan dengan j = 3, 4, 5

   A[2] dengan A[3]; A[2] dengan A[4]; A[2] dengan A[5]
4. Untuk i = 3 (pass = 3), buat perbandingan dengan j = 4, 5

   A[3] dengan A[4]; A[3] dengan A[5]
5. Untuk i = 4 (pass = 4), buat perbandingan dengan j = 5

   A[4] dengan A[5]

Praktik

```python
def selection_sort(arr):
    n = len(arr)

    # Iterasi untuk setiap elemen dalam array
    for i in range(n):
        # Anggap elemen saat ini adalah yang terkecil
        min_index = i

        # Cari elemen terkecil di sisa array
        for j in range(i + 1, n):
            if arr[j] < arr[min_index]:
                min_index = j

        # Tukar elemen terkecil dengan elemen saat ini
        arr[i], arr[min_index] = arr[min_index], arr[i]

    return arr

# Contoh penggunaan
data = [64, 25, 12, 22, 11]
print("Sebelum diurutkan:", data)
sorted_data = selection_sort(data)
print("Setelah diurutkan:", sorted_data)
```

```
Sebelum diurutkan: [64, 25, 12, 22, 11]
Setelah diurutkan: [11, 12, 22, 25, 64]
```

Bubble Sort

Ide dasar dari bubble sort adalah :

- membandingkan elemen-elemen secara berurutan melalui suatu array.
- Setiap kali perbandingan dibuat, elemen dipindah jika tidak sesuai urutannya.

Untuk ilustrasi dari algoritma ini, akan diberikan contoh sbb. :

Sebuah Array A, memiliki 5 nilai-nilai bertipe data integer.

A[1] : 5 A[2] : 13 A[3] : -2 A[4] : 10 A[5] : 2

Pass No. 1 : i = 1

	Comp. 1 (j=1)	Comp. 2 (j=2)	Comp. 3 (j=3)	Comp. 4 (j=4)	
unsorted					
A[1]	13	5	5	5	5
A[2]	5	13	-2	-2	-2
A[3]	-2	-2	13	10	10
A[4]	10	10	10	13	2
sorted					
A[5]	2	2	2	2	13

Pass No. 2 : i = 2

	Comparison 1 (j=1)	Comparison 2 (j=2)	Comparison 3 (j=3)	
unsorted				
A[1]	5	-2	-2	-2
A[2]	-2	5	5	5
A[3]	10	10	10	2
sorted				
A[4]	2	2	2	10
A[5]	13	13	13	13

Pass No. 3 : i = 3

	Comparison 1 (j=1)	Comparison 2 (j=2)	
unsorted			
A[1]	-2	-2	-2
A[2]	5	5	2
sorted			
A[3]	2	2	5
A[4]	10	10	10
A[5]	13	13	13

Pass No. 4 : i = 4

	Comparison 1 (j=1)	
sorted		
A[1]	-2	-2
A[2]	2	2
A[3]	5	5
A[4]	10	10
A[5]	13	13

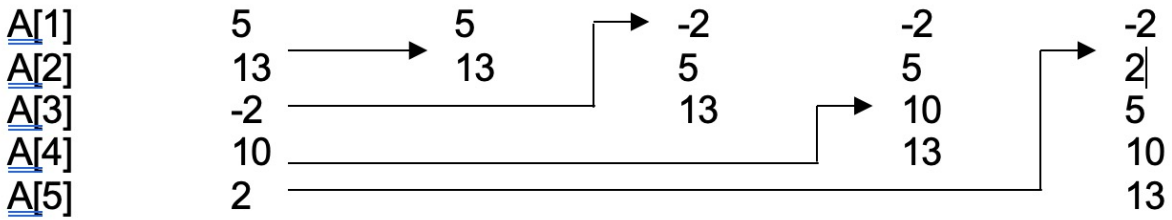
Praktik

```python
def bubble_sort(arr):
    n = len(arr)

    # Iterasi sebanyak panjang array
    for i in range(n):
        # Flag untuk mengecek apakah terjadi pertukaran
        swapped = False

        # Iterasi membandingkan elemen berpasangan
        for j in range(0, n - i - 1):
            # Jika elemen kiri lebih besar dari kanan, tukar
            if arr[j] > arr[j + 1]:
                print(f"Tukar {arr[j]} dengan {arr[j+1]}")
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True

        # Jika tidak ada pertukaran, array sudah terurut
        if not swapped:
            break

    return arr

# Contoh penggunaan
data = [64, 34, 25, 12, 22, 11, 90]
print("Sebelum diurutkan:", data)
sorted_data = bubble_sort(data)
print("Setelah diurutkan:", sorted_data)
```

```
Sebelum diurutkan: [64, 34, 25, 12, 22, 11, 90]
Tukar 64 dengan 34
Tukar 64 dengan 25
Tukar 64 dengan 12
Tukar 64 dengan 22
Tukar 64 dengan 11
Tukar 34 dengan 25
Tukar 34 dengan 12
Tukar 34 dengan 22
Tukar 34 dengan 11
Tukar 25 dengan 12
Tukar 25 dengan 22
Tukar 25 dengan 11
Tukar 22 dengan 11
Tukar 12 dengan 11
Setelah diurutkan: [11, 12, 22, 25, 34, 64, 90]
```

Insertion Sort

Ilustrasi :



Langkah-langkah (Pengurutan secara Ascending) :

- For i=2 , elemen A[2] dipindah ke TEMP.
  TEMP dibandingkan dengan data A[1] :
  - Jika TEMP < A[1] ➔ A[2] ← A[1] ; A[1] ← TEMP
  - Jika TEMP > = A[1] ➔ A[2] ← TEMP

- For i=3 , elemen A[3] dipindah ke TEMP.
  TEMP dibandingkan dengan data A[2] :
  - Jika TEMP < A[2] ➔ A[3] ← A[2] ; TEMP dibandingkan dengan A[1], proses 1a dan 1b diulangi
  - Jika TEMP > = A[2] ➔ A[3] ← TEMP
- Dst. sampai dengan i = 5

Praktik

```python
def insertion_sort(arr):
    # Mulai dari elemen kedua karena elemen pertama dianggap sudah terurut
    for i in range(1, len(arr)):
        key = arr[i]        # Elemen yang akan disisipkan
        j = i - 1

        # Geser elemen yang lebih besar dari key ke kanan
        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]  # Geser ke kanan
            j -= 1

        # Sisipkan key di posisi yang tepat
        arr[j + 1] = key

        # Tampilkan langkah-langkahnya
        print(f"Langkah {i}: {arr}")

    return arr

# Contoh penggunaan
data = [12, 11, 13, 5, 6]
print("Sebelum diurutkan:", data)
sorted_data = insertion_sort(data)
print("Setelah diurutkan:", sorted_data)
```

```
Sebelum diurutkan: [12, 11, 13, 5, 6]
Langkah 1: [11, 12, 13, 5, 6]
Langkah 2: [11, 12, 13, 5, 6]
Langkah 3: [5, 11, 12, 13, 6]
Langkah 4: [5, 6, 11, 12, 13]
Setelah diurutkan: [5, 6, 11, 12, 13]
```

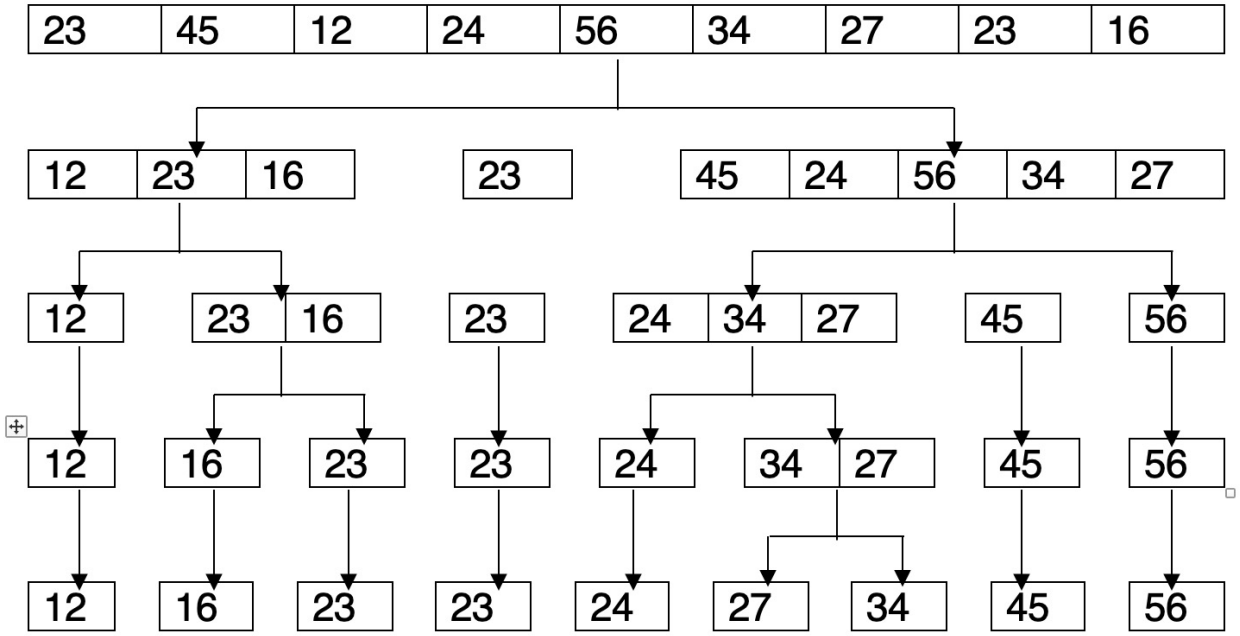
Quick Sort

Disebut juga dengan metode *Partition Exchange Sort*, diperkenalkan oleh C.A.R. Hoare pada tahun 1962. Pada metode quick sort, jarak dari kedua elemen yang ditukarkan dibuat cukup besar dengan tujuan untuk mempertinggi efektifitasnya. Hal ini mengingat metode buble sort yang menggunakan jarak cukup dekat ternyata kurang efektif.

Ide dasar :

- Pilih sembarang elemen (biasanya elemen pertama), misalnya X, kemudian semua elemen disusun dengan menempatkan X pada posisi J sedemikian rupa sehingga elemen ke 1 s/d J-1 mempunyai nilai lebih kecil dari X dan elemen ke J+1 s/d N mempunyai nilai lebih besar dari X.
- Langkah berikutnya, proses diatas diulang untuk kedua subvektor, dst.

Ilustrasi :


23 45 12 24 56 34 27 23 16
12 23 16
23
45 24 56 34 27
12
23 16
23
24 34 27
45
56
12
16 23
23
24
34 27
45
56
12
16
23
23
24
27
34
45
56

Praktik

```python
def quick_sort(arr):
    # Basis: jika array berisi 0 atau 1 elemen, sudah terurut
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]  # pilih elemen pertama sebagai pivot
        left = [x for x in arr[1:] if x < pivot]     # elemen yang lebih kecil dari pivot
        right = [x for x in arr[1:] if x >= pivot]   # elemen yang lebih besar atau sama dengan pivot

        print(f"Pivot: {pivot}")
        print(f"Kiri: {left}")
        print(f"Kanan: {right}\n")

        # Rekursi untuk bagian kiri dan kanan, lalu gabungkan hasilnya
        return quick_sort(left) + [pivot] + quick_sort(right)

# Contoh penggunaan
data = [10, 7, 8, 9, 1, 5]
print("Sebelum diurutkan:", data)
sorted_data = quick_sort(data)
print("Setelah diurutkan:", sorted_data)
```

```
Sebelum diurutkan: [10, 7, 8, 9, 1, 5]
Pivot: 10
Kiri: [7, 8, 9, 1, 5]
Kanan: []

Pivot: 7
Kiri: [1, 5]
Kanan: [8, 9]

Pivot: 1
Kiri: []
Kanan: [5]

Pivot: 8
Kiri: []
Kanan: [9]

Setelah diurutkan: [1, 5, 7, 8, 9, 10]
```

TERIMA KASIH

Next ---- PERTEMUAN ke-10

TEKNIK ELEKTRO
FTI UJ

ENDANG SRI RAHAYU