TUGAS
ALGORITME PENGURUTAN
KELAS PENGEMBANGAN PERANGKAT LUNAK

LAILATUL QADRI

YBM PLN di desa Plosogeneng,
Jombang, jawa timur

YBM PLN
Yayasan Baitul Maal

Pesantren Teknologi Informasi dan Komunikasi
PeTIK
Creates Future Skilled Professionals

PENGERTIAN ALGORITMA SORTING

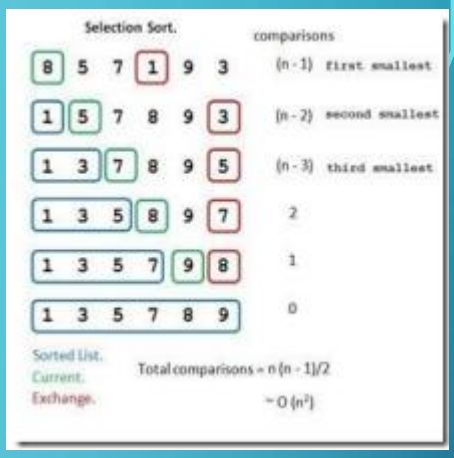
Secara singkatnya sorting adalah metode untuk pengurutan data.sedangkan menurut garis besarnya adalah Sorting (Pengurutan) adalah suatu proses penyusunan kembali kumpulan objek menggunakan tata aturan tertentu. Sorting disebut juga sebagai suatu algoritma untuk meletakkan kumpulan elemen data ke dalam urutan tertentu berdasarkan satu atau beberapa kunci dalam tiap-tiap elemen. Pengurutan atau sorting merupakan proses dasar yang ada dalam sebuah algoritma dan struktur data. Penggunaan algoritma sorting dapat pula diaplikasikan pada algoritma Python. Tujuan utama dari proses pengurutan atau sorting adalah untuk mengurutkan data berdasarkan keinginan baik itu dari yang terendah maupun yang tertinggi, sehingga data yang dihasilkan akan lebih terstruktur, teratur dan sesuai dengan kebutuhan. Pengertian Algoritma Sorting

Macam-macam algoritma pengurutan :

- Selection Sort
- Bubble Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Shell Sort
- Counting Sort
- Radix Sort
- Heap Sort

SELECTION SORT

Algoritma selection sort merupakan pengurutan dengan konsep memilih elemen dengan nilai paling rendah dan menukar elemen tersebut dengan elemen ke i. Nilai dari 1 ke n, yang dimana n merupakan jumlah total elemen di kurang satu.



Analogi algoritma selection sort :

- Memulai pengecekan data dari data 1 hingga ke data n.
- Menentukan bilangan dengan index terkecil dari data pada bilangan tersebut.
- Menukar bilangan index terkecil dengan bilangan pertama.
- Begitu Seterusnya hingga data berhasil di urutkan semuanya.

E: > bulan juli > pengurutan.py > ...

```python
#pengurutan - ssorting

# #membuat fuction yang menerima parameter array
def selection_sort(array):
    # membuat for perulangan sebanyak isi/value dari array
    for i in range(len(array)-1):
        #membuat variabel index_awal dan mengisinya dengan i
        index_awal = i
        #membuat for perulangan dimana i + 1,dan loob sebanyak isi dara array
        for j in range(i+1, len(array)):
            #jika array[j] lebih kecil dari array[index_awal]
            if array[j] < array[index_awal]:
                #maka index_awal = j
                index_awal = j
        #mengubah value
        array[i], array[index_awal] = array[index_awal], array[i]


angka = [1, 4, 3, 9, 6, 5]
selection_sort(angka)

print(angka)

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\bulan juli>  e:; cd 'e:\bulan juli'; & '
ncher' '54199' '--' 'e:\bulan juli\pengurutan.
[1, 3, 4, 5, 6, 9]
PS E:\bulan juli>
```

BUBBLE SORT

Algoritma bubble sort cukup populer dan sederhana. Proses pada bubble sort dilakukan dengan pertukaran data di sebelahnya secara terus menerus hingga dalam suatu iterasi tertentu tidak ada lagi perubahan atau pertukaran. Algoritma bubble sort termasuk ke dalam kategori algoritma comparison sort, karena menggunakan perbandingan pada operasi antar elemen nya.

Analogi algoritma bubble sort :

- Bandingkan nilai pada data ke satu dengan data ke dua.
- Apabila nilai data ke satu lebih besar dari data ke dua maka tukar posisinya.
- Kemudian data yang lebih besar tersebut dibandingkan lagi dengan data ketiga.
- Apabila data ke tiga lebih kecil dari data ke dua maka tukar posisinya.
- Dan begitu seterusnya hingga semua data yang ada menjadi terurut

```
angka = [9, 5, 8, 6, 7, 4, 3, 1, 2]
def bubble_sort(array):
    for i in range(len(array)-1):
        for j in range(len(array)-1-i):
            if array[j] > array[j+1]:
                array[j], array[j+1] = array[j+1], array[j]

bubble_sort(angka)
print(angka)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\bulan juli>  e:; cd 'e:\bulan juli'; & '
ncher' '54281' '--' 'e:\bulan juli\pengurutan.
[1, 2, 3, 4, 5, 6, 7, 8, 9]
PS E:\bulan juli>
```

INSERTION SORT

Algoritma insertion sort merupakan suatu metode pengurutan data dengan melakukan penempatan setiap elemen data pada posisinya dengan membandingkan dengan data-data yang telah ada. Prinsip dari insertion sort adalah dengan membagi data yang akan diurutkan menjadi dua kelompok, satu kelompok yang belum diurutkan dan yang satunya lagi sudah diurutkan, Elemen yang pertama diambil dari kelompok list yang belum diurutkan dan kemudian ditempatkan sesuai posisinya pada bagian lain yang belum diurutkan.

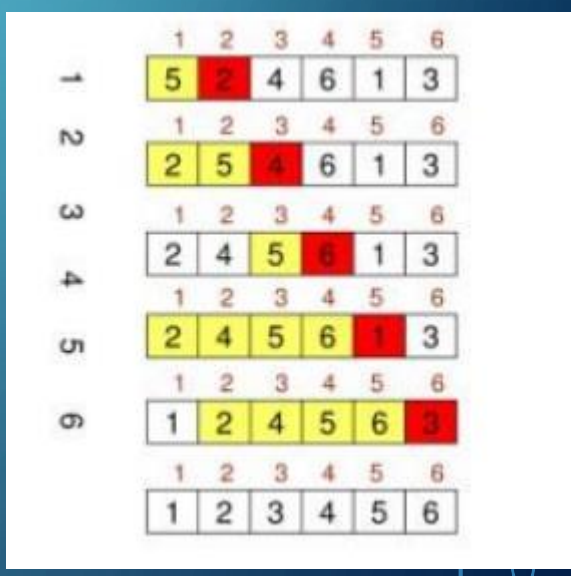
Analogi algoritma insertion sort :

- Membandingkan data kedua dengan data kesatu.
- Apabila data ke dua lebih kecil maka tukar posisinya.
- Data ketiga dibandingkan dengan data kesatu dan kedua.
- Apabila data ketiga lebih kecil tukar lagi posisinya.
- Data keempat dibandingkan dengan data ketiga hingga kesatu.
- Apabila data keempat lebih kecil dari ketiga maka letakkan data keempat ke posisi paling depan.
- Begitu seterusnya hingga tidak ada lagi data yang dapat dipindahkan

```
angka = [9, 5, 8, 6, 7, 4, 3, 1, 2]
def insertion_sort(array):
    for i in range(1, len(array)):
        value = array [i]
        j = i -1
        while j >= 0 and array[j] > value:
            array[j+1] = array[j]
            j = j - 1
        array[j+1] = value
insertion_sort(angka)
print(angka)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\bulan juli>  e:; cd 'e:\bulan juli'; & 'C
ncher' '54318' '--' 'e:\bulan juli\pengurutan.p
[1, 2, 3, 4, 5, 6, 7, 8, 9]
PS E:\bulan juli>
```



MARGE SORT

Algoritma merge sort merupakan salah satu pengurutan dengan metode memecah data kemudian mengolah untuk diselesaikan pada setiap bagian dan menggabungkan kembali sehingga data tersebut berhasil tersusun. Merge sort dalam menyelesaikan pengurutan membutuhkan fungsi rekursif. Analogi algoritma marge sort :

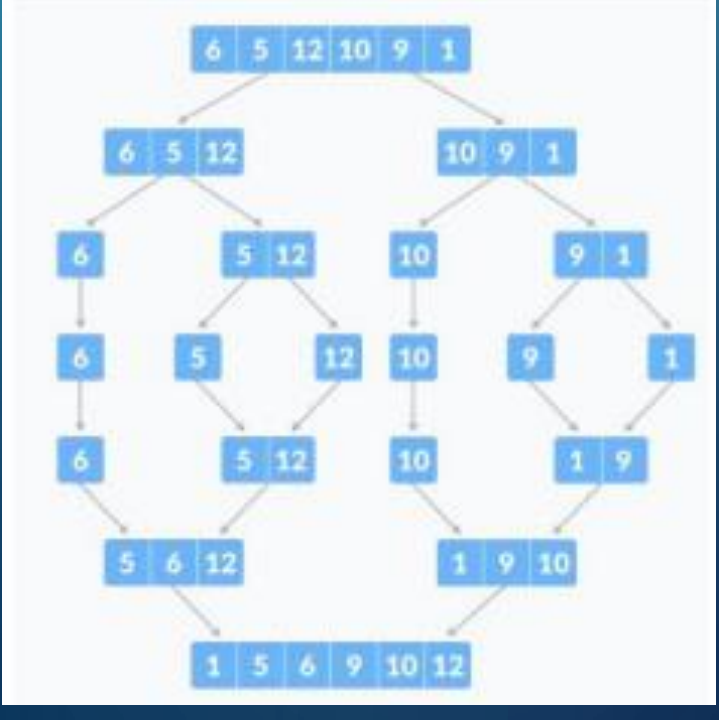
•Data dipecah menjadi dua kelompok dimana kelompok pertama adalah setengah apabila data genap atau setengah kurang satu apabila data ganjil dari seluruh data.

•Kemudian dilakukan pemecahan kembali pada masing-masing kelompok hingga hanya terdapat satu data pada satu kelompok.

•Setelah digabungkan kembali dengan membandingkan pada blok yang sama apakah data pertama lebih besar dari pada data ketengah ditambah satu, jika iya maka data ketengah ditambah satu dipindah menjadi data pertama.

•Kemudian data pertama tadi hngga data ketengah dipindah menjadi data kedua sampai data ketengah ditambah satu.

•Begitu seterusnya sehingga membentuk sebuah data yang tersusun dalam satu kelompok yang utuh.



```python
angka = [9, 5, 8, 6, 7, 4, 3, 1, 2]
def merge_sort(array):
    if len(array) > 1:
        mid = len(array) // 2
        leftarr = array[:mid]
        rightarr = array[mid:]
        merge_sort(leftarr)
        merge_sort(rightarr)
        merge(leftarr, rightarr, array)

def merge(left, right, array):
    i, j, k = 0, 0, 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            array[k] = left[i]
            i = i + 1
        else:
            array[k] = right[j]
            j = j + 1
        k = k + 1

    while i < len(left):
        array[k] = left[i]
        k = k + 1
        i = i + 1

    while j < len(right):
        array[k] = right[j]
        k = k + 1
        j = j + 1

merge_sort(angka)
print(angka)
```

```
75          j = j + 1

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\bulan juli>  e:; cd 'e:\bulan juli'; & 'C:
ncher' '54361' '--' 'e:\bulan juli\pengurutan.py
[1, 2, 3, 4, 5, 6, 7, 8, 9]
PS E:\bulan juli>
```

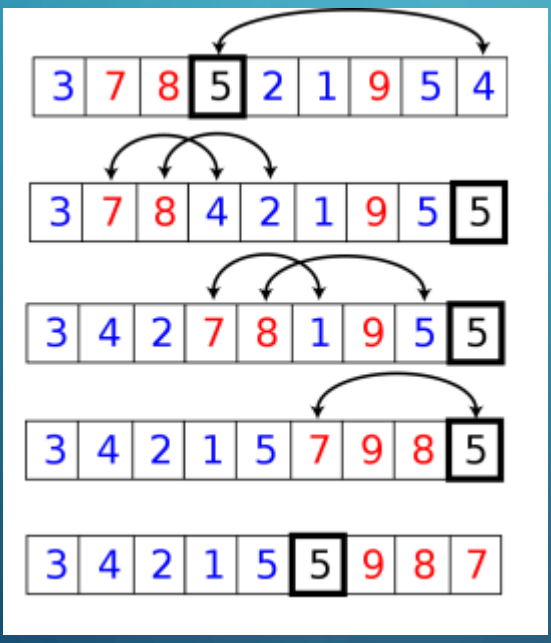
QUICK SORT

Algoritma quick sort ini cara kerjanya berprinsip pada penekatan divide and conquer yakni dengan memilih satu elemen sebagai elemen pivot dan mempartisi array sehingga sisi kiri pada pivot mempunyai semua elemen dengan nilai yang lebih kecil dibandingkan dengan elmen pivot dan pada sisi kanan mempunyai semua elemen dengan nilai yang lebih besar dibandingkan dengan nilai elemen pivot. Analogi algoritma marge sort :

•Mempunyai data A yang memiliki N elemen, pilih sembarang elemen dari data tersebut biasanya elemen pertama misalkan elemen x

•Kemudian semua elemen tersebut disusun dengan menempatkan x pada posisi j sedemikian rupa sehingga elemen ke satu sampai pada j-1 dan memiliki nilai yang lebih besar dari x

•Begitu seterusnya setiap sub data



```python
angka = [9, 5, 8, 6, 7, 4, 3, 1, 2]
def quick_sort(array):
    quick_sort_rec(array, 0, len(array)-1)

def quick_sort_rec(array, start, end):
    if start < end:
        pivot_idx = partition(array, start, end)
        quick_sort_rec(array, start, pivot_idx-1)
        quick_sort_rec(array, pivot_idx+1, end)

def partition(array, start, end):
    pivot = array[end]
    i = start
    for j in range(start, end):
        if array[j] <= pivot:
            array[i], array[j] = array[j], array[i]
            i = i + 1
    array[i], array[end] = array[end], array[i]
    return i


quick_sort(angka)
print(angka)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\bulan juli>  e:; cd 'e:\bulan juli'; & 'C:
ncher' '54382' '--' 'e:\bulan juli\pengurutan.py
[1, 2, 3, 4, 5, 6, 7, 8, 9]
PS E:\bulan juli>
```

REFERENSI

https://ekarisky.com/analisis-perbandingan-algoritmaselection-sort-dengan-merge-sort-algoritma/

https://dqlab.id/mengenal-algoritma-sorting-padaalgoritma-python

https://bekti.net/blog/implementasi-bubble-sort-cpp/

https://slideplayer.info/slide/14733804/

https://blogbugabagi.blogspot.com/2019/12/algoritmasorting-qoutput-quicj-sortt.html

https://www.programiz.com/dsa/merge-sort

https://id.wikipedia.org/wiki/Quicksort