

Tanggal 6 Juli 2020

Nama : Maulana Alifq Ibtisan  
NIM : 120018020  
Kelas : C

Mata Kuliah : Algoritma dan Struktur Data

JL

### 1.) a) Insertion Sort

prinsip kerja :

Insertion sort mengurutkan data dengan cara membandingkan dan mengurutkan dua data pertama pada array, kemudian membandingkan data pada array berikutnya apakah sudah berada di tempat semestinya.

Kinerja Big-O Notation : O(n)

dimana n adalah jumlah item.

Algoritma :

1. Membagi sebuah list menjadi 2 bagian yaitu bagian sorted dan unsorted, dengan mengasumsikan elemen pertama dalam list tersebut adalah bagian sorted dan elemen kedua sampai terakhir adalah elemen unsorted.
2. Membandingkan elemen pertama bagian unsorted dengan tiap elemen bagian sorted yang menjadi pembanding, maka sisipkan elemen unsorted disebelah kiri elemen sorted, lalu menggeser elemen sorted pembanding ke kanan.
3. Untuk pencarian dari terkecil ke terbesar :
  - Jika elemen pertama bagian unsorted lebih sedikit kecil dari elemen bagian sorted yang menjadi pembanding, maka sisipkan elemen unsorted disebelah kiri elemen sorted, lalu menggeser elemen sorted pembanding ke kanan.
  - Jika elemen pertama bagian unsorted lebih besar dari bagian sorted yang menjadi pembanding, maka sisipkan elemen unsorted disebelah kanan elemen sorted.
- Untuk pencarian dari terbesar ke terkecil :
  - Jika elemen pertama bagian unsorted lebih besar dari elemen bagian sorted yang menjadi pembanding, maka sisipkan elemen unsorted disebelah kiri elemen sorted, lalu menggeser elemen sorted pembanding ke kanan.
  - Jika elemen pertama bagian unsorted lebih kecil dari elemen bagian sorted yang menjadi pembanding, maka sisipkan elemen unsorted disebelah kanan elemen sorted

4. Melakukan pengecekan sampai elemen terakhir bagian unsorted sehingga data menjadi list.

### b) Selection sort

Prinsip kerja :

Selection sort mengurutkan data dengan cara memilih data terkecil untuk konduksi dibandingkan dan difiturkan dengan elemen pada data awal, dan seterusnya sampai pengecekan seluruh elemen selesai sehingga menghasilkan data yang list.

## Kinerja Big-O Notation

Algoritma :

1. Mengambil elemen pertama pada list sebagai elemen minimum jika sort ascending, atau mengambil elemen pertama dalam list sebagai elemen maximum jika sort descending.
2. Mengambil data terkecil pada elemen sisanya. Jika data terkecil yang diambil lebih kecil dari elemen minimum awal, maka posisinya ditukarkan lalu mengulangi data terkecil.
3. Bergantik tetapi, mengambil elemen selanjutnya menjadi elemen minimum.
4. Mengulangi langkah 1 - 3 hingga semuanya sedang dicuci sehingga data menjadi urut.

### a) Bubble Sort

Prinsip kerja :

Bubble sort mengurutkan data dengan cara menukarkan data dengan data yang ada disebelahnya secara turun menurun sampai dalam satu iterasi tersebut yang tidak ada perubahan lagi.

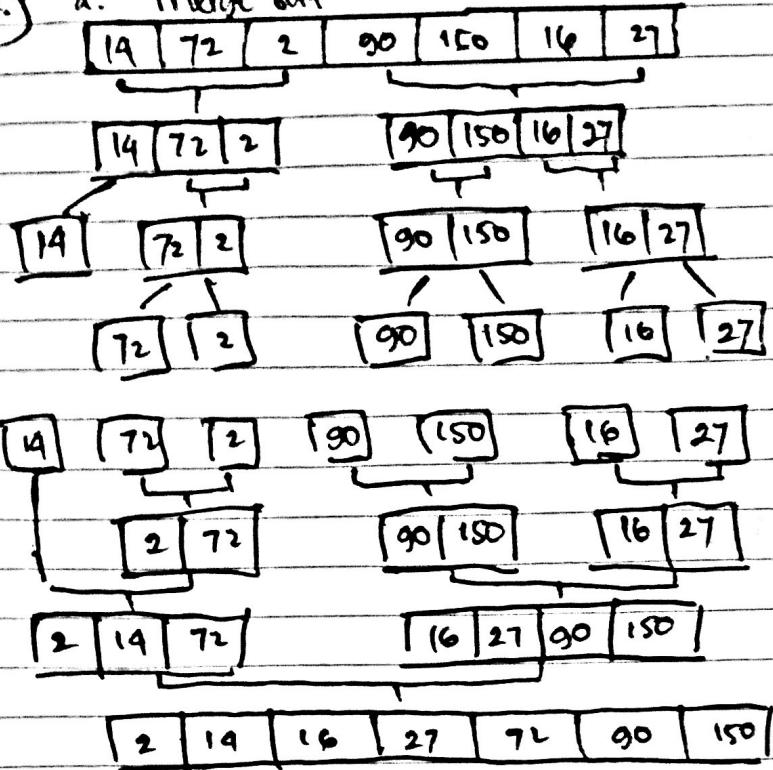
. Kinerja Big-O Notation :

- Best case :  $O(n)$
- Average case :  $O(n^2)$
- Worst case :  $O(n^2)$

• Algoritma :

1. Memeriksa apakah memiliki nilai dalam array
2. Mbandingkan data 1 dengan 2.
3. Jika data 1 lebih besar dari data 2 maka tukar posisinya, kemudian data yang besar selanjutnya dibandingkan dengan data 3
4. Lakukan proses hingga array diurutkan.

## 2.) a. Merge Sort

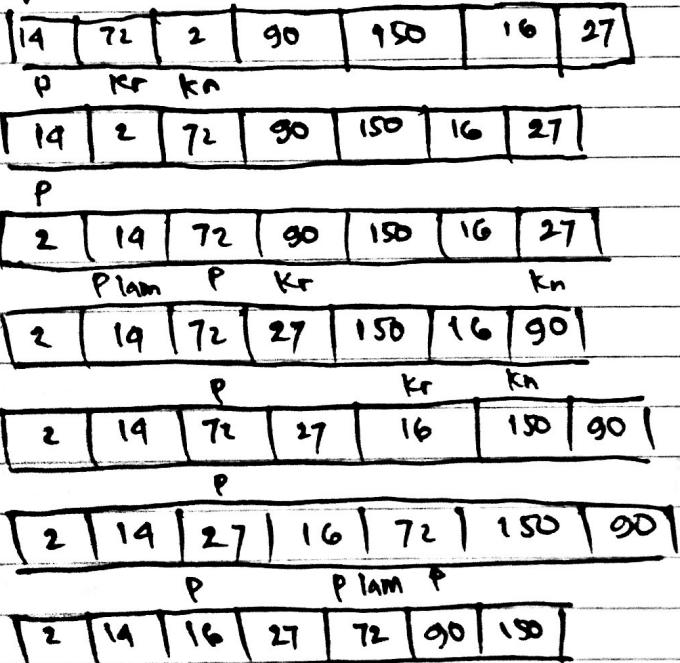


1. List dibagi menjadi beberapa bagian

bagian

2. Bagian-bagian tersebut diurutkan

## b. Quick Sort



1. data pertama menjadi pivot penanda  
kanan harus  $> 19$ , penanda kiri harus  $< 19$

2. berhenti dan bertukar karena  $72 > 19$   
dan  $2 < 19$

3. 19 sebagai pivot lama digantikan oleh  
72 sebagai pivot baru

4.  $90 > 72$  dan  $27 < 72$  maka berhenti  
lalu bertukar

5.  $150 > 72$  dan  $16 < 72$  berhenti dan bertukar

Dose

Kode Python:

## a. Merge sort

```
def mergeSort (numbers):
    if len (numbers) > 1:
        middle = len (numbers) / 2
        kiri = numbers [:middle]
        kanan = numbers [middle:]

        mergeSort (kiri)
        mergeSort (kanan)

        i = j = k = 0

        while i < len (kiri) and j < len (kanan):
            if kiri [i] < kanan [j]:
                numbers [k] = kiri [i]
                i += 1
            else:
                numbers [k] = kanan [j]
                j += 1
            k += 1

        while i < len (kiri):
            numbers [k] = kiri [i]
            i += 1
            k += 1

        while j < len (kanan):
            numbers [k] = kanan [j]
            j += 1
            k += 1
```

numbers = [49, 72, 2, 90, 150, 16, 27]

mergeSort (numbers)

print (numbers)

Date

## b. Quick Sort

Kode python:

```
def quickSort(A):
```

```
    quickSorting(A, len(A) - 1)
```

```
def quickSorting(A, Awal, Akhir):
```

```
    if Awal < Akhir:
```

```
        titik_belah = Partisi(A, Awal, Akhir)
```

```
        quickSorting(A, Awal, titik_belah - 1)
```

```
        quickSorting(A, titik_belah + 1, Akhir)
```

```
def partisi(A, Awal, Akhir):
```

```
    nilai_pivot = A[Awal]
```

```
    sub_kiri = Awal
```

```
    sub_kanan = Akhir
```

```
    Selesai = False
```

```
    while not Selesai:
```

```
        while sub_kiri <= sub_kanan and A[sub_kiri] <= nilai_pivot:
```

```
            sub_kiri += 1
```

```
        while A[sub_kanan] >= nilai_pivot and sub_kanan >= nilai_pivot:
```

```
            sub_kanan -= 1
```

```
        if sub_kanan < sub_kiri:
```

```
            Selesai = True
```

```
        else:
```

```
            temp = A[sub_kiri]
```

```
A[sub_kiri] = A[sub_kanan]
```

```
A[sub_kanan] = temp
```

```
temp = A[Awal]
```

```
A[Awal] = A[sub_kanan]
```

```
A[sub_kanan] = temp
```

```
return sub_kanan
```

```
numbers = [14, 72, 2, 90, 150, 16, 27]
```

```
quickSort(numbers)
```

```
print(numbers)
```

3.) Membuat sebuah class Stack kemudian membuat list untuk hexa decimal yaitu 0-9 dan A-F. Selama saat kondisi perulangan tidak 0 maka akan terus melakukan perulangan.

Kode python:

```
def hex(desimal):
```

```
    s = Stack()
```

```
    while desimal != 0:
```

```
        sisa = desimal % 16
```

```
        desimal = desimal // 16
```

```
        if sisa == 10:
```

```
            sisa = 'A'
```

```
        elif sisa == 11:
```

```
            sisa = 'B'
```

```
        elif sisa == 12:
```

```
            sisa = 'C'
```

```
        elif sisa == 13:
```

```
            sisa = 'D'
```

```
        elif sisa == 14:
```

```
            sisa = 'E'
```

```
        elif sisa == 15:
```

```
            sisa = 'F'
```

```
        s.push(sisa)
```

```
    hasil = ""
```

```
    for i in range(len(s)):
```

```
        hasil = hasil + str(s.pop())
```

```
    return hasil
```

```
print(hex(2550))
```

## 1.) a) Struktur

- a. Deque merupakan sebuah antiran yang memiliki dua ujung sekaligus. Artinya item bisa disisipkan melalui kedua ujungnya yaitu ujung head dan tail.
- b. Queue Priority adalah queue dengan aturan Highest Priority In First Out (HPIFO). Konsep ini memiliki cara kerja ketika item akan disisipkan ke antiran maka item itu akan disertai dengan suatu penanda prioritas.
- \* Perbedaannya adalah queue hanya dapat menyisipkan elemen melalui ujung belakang (tail) dan keluar melalui ujung depan (head) yang disertai dengan penanda prioritas.

## 5.) a. Preorder dimulai dari akar lalu ke subtree bagian kiri dilanjut subtree kanan hingga tidak ada subtree lagi

1, 2, 4, 5, 3, 6

- b. Inorder dimulai dan traverse subtree dengan bagian kiri, lalu mengunjungi simpulnya, dilanjutkan traverse subtree bagian kanan.

4, 2, 5, 1, 6, 3

- c. Postorder dilakukan dengan subtree kiri dan kanan ditulis simpul dikunjungi dulu sebelum simpulnya dikunjungi.

1, 5, 2, 6, 3, 1