LAPORAN PRAKTIKUM

WORKSHEET 3

Oleh:

Muhamad Ilham Habib (140810180018)

Kelas B

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

JATINANGOR

2019

I. Tujuan

1. Mahasiswa paham mengenai perhitungan worst case.
2. Mahasiswa paham cara menghitung Big-O Notation.
3. Mahasiswa paham menghitung Big-Ω dan Big- Θ.
4. Mahasiswa paham menghitung aturan kompleksitas waktu asimtotik.

II. Landasan Teori

II.1 Dalam analisis algoritma kita selalu mengutamakan perhitungan worst case dengan alasan sebagai berikut:

a. Worst-case running time merupakan *upper bound* (batas atas) dari running time untuk input apapun. Hal ini memberikan jaminan bahwa algoritma yang kita jalankan tidak akan lebih lama lagi dari *worst-case*.
b. Untuk beberapa algoritma, *worst-case* cukup sering terjadi. Dalam beberapa aplikasi pencarian, pencarian info yang tidak ada mungkin sering dilakukan.
c. Pada kasus average-case umumnya lebih sering seperti worst-case. Contoh: misalkan kita secara random memilih angka dan mengimplementasikan insertion sort, *average-case* = *worst-case* yaitu fungsi kuadratik dari *n*.

II.2 Big-O Notation adalah cara untuk mengkonversi keseluruhan langkah-langkah suatu algoritma kedalam bentuk Aljabar, yaitu denganmenghiraukan konstanta yang lebih kecil dan koefisien yang tidakberdampak besar terhadap keseluruhan kompleksitas permasalahan yang diselesaikan oleh algoritma tersebut.

a. *Worst-case* dihitung dengan *Big-O Notation*.
b. $T(n) = O(f(n))$ artinya $T(n)$ berorde paling besar $f(n)$ bila terdapat konstanta C dan n_0 sehingga $T(n) \leq C.f(n)$, untuk $n \geq n_0$.
c. Dalam pembuktian *Big-O Notation,* perlu dicari nilai n_0 dan C sehingga terpenuhi kondisi $T(n) \leq C.f(n)$.

II.3 Big-O Notation polinomial berderajat *n* digunakan untuk memperkirakan kompleksitas dengan mengabaikan suku berorde rendah.

a. Teorema 1

$T(n) = a_m n^m + a_{m-1} n^{m-1} + a_1 n + a_0$ adalah polinom berderajat *m* maka $T(n) = O(n^m)$

b. Teorema 2

Misalkan $T_1(n) = O(f(n))$ dan $T_2(n) = O(g(n))$, maka

1. $T_1(n) + T_2(n) = O(max(f(n), g(n))$ atau $T_1(n) + T_2(n) = O(f(n) + g(n))$
2. $T_1(n).T_2(n) = O(f(n)).O(g(n)) = O(f(n).g(n))$
3. $O(c.f(n)) = O(f(n))$, *c* adalah konstanta
4. $f(n) = O(f(n))$

II.4 Aturan Menentukan Kompleksitas Waktu Asimptptik

a. Jika kompleksitas waktu T(n) dari algoritma sudah dihitung, maka kompleksitas waktu asimptotiknya dapat langsung ditentukan dengan mengambil suku yang mendominasi fungsi T dan menghilangkan koefisiennya (sesuai TEOREMA 1)
b. Kita bisa langsung menggunakan notasi Big-O, dengan cara: Pengisian nilai (assignment), perbandingan, operasi aritmatika (+, -, / ,* , div, mod), read, write, pengaksesan elemen larik, memilih field tertentu dari sebuah record, dan pemanggilan function/void membutuhkan waktu O(1)

II.5 Notasi Big-O hanya menyediakan batas atas (upper bound) untuk perhitungan kompleksitas waktu asimptotik, tetapi tidak menyediakan batas bawah (lower bound). Untuk itu, lower bound dapat ditentukan dengan Big-Ω Notation dan Big-θ Notation.

a. $T(n) = \Omega(f(n))$, artinya $T(n)$ berorde paling kecil $f(n)$ bila terdapat konstanta *C* dan *n* sehingga $T(n) \geq C.(f(n))$, dengan syarat nilai *c* dan *n* positif.

b. *T(n)=Θ(h(n))*, artinya *T(n)* berorde sama dengan *h(n)* Jika *T(n)=O(h(n))* dan *T(n)=Ω(g(n))*.

$C_1 f(n) \leq T(n) \leq C_2 f(n)$, dengan syarat nilai *c* dan *n* positif

III. Worksheet 3

1. Untuk $T(n) = 2 + 4 + 6 + 8 + 10 + \cdots + n^2$, tentukan nilai C, f(n), n_o, dan notasi Big-O sedemikian sehingga $T(n) = O(f(n))$ jika $T(n) \leq C\ untuk\ suatu\ n \geq n_0$

2. Buktikan bahwa untuk konstanta-konstanta positif p, q, dan r:
   $T(n) = pn^2 + qn + r$ adalah $O(n^2), \Omega(n^2), dan\ \Theta(n^2)$

3. Tentukan waktu kompleksitas asimptotik (Big-O, Big-Ω, dan Big-Θ) dari kode program berikut:

```
for k ← 1 to n do
    for i ← 1 to n do
        for j ← to n do
            w_ij ← w_ij or w_ik and w_kj
        endfor
    endfor
endfor
```

4. Tulislah algoritma untuk menjumlahkan dua buah matriks yang masing-masing berukuran n x n. Berapa kompleksitas waktunya T(n)? dan berapa kompleksitas waktu asimptotiknya yang dinyatakan dalam Big-O, Big-Ω, dan Big-Θ?

5. Tulislah algoritma untuk menyalin (copy) isi sebuah larik ke larik lain. Ukuran elemen larik adalah n elemen. Berapa kompleksitas waktunya T(n)? dan berapa kompleksitas waktu asimptotiknya yang dinyatakan dalam Big-O, Big-Ω, dan Big-Θ?

6. Diberikan algoritma Bubble Sort sebagai berikut:

```
procedure BubbleSort(input/output a1, a2, ..., an : integer)
{ Mengurut tabel integer TabInt[1..n] dengan metode pengurutan bubble-
sort
  Masukan: a1, a2, ..., an
  Keluaran: a1, a2, ..., an (terurut menaik)
}
Deklarasi
   k : integer     { indeks untuk traversal tabel }
   pass : integer { tahapan pengurutan }
   temp : integer { peubah bantu untuk pertukaran elemen tabel }
Algoritma
   for pass ← 1 to n - 1 do
     for k ← n downto pass + 1 do
       if ak < ak-1 then
           { pertukarkan ak dengan ak-1 }
           temp ← ak
           ak ← ak-1
           ak-1←temp
       endif
     endfor
   endfor
```

(a) Hitung berapa jumlah operasi perbandingan elemen-elemen tabel!
(b) Berapa kali maksimum pertukaran elemen-elemen tabel dilakukan?
(c) Hitung kompleksitas waktu asimptotik (Big-O, Big-Ω, dan Big-Θ) dari algoritma Bubble Sort tersebut!

7. Untuk menyelesaikan problem X dengan ukuran N tersedia 3 macam algoritma:
   (a) Algoritma A mempunyai kompleksitas waktu O(log N)
   (b) Algoritma B mempunyai kompleksitas waktu O(N log N)
   (c) Algoritma C mempunyai kompleksitas waktu O(N^2)

   Untuk problem X dengan ukuran N=8, algoritma manakah yang paling cepat? Secara asimptotik, algoritma manakah yang paling cepat?

8. Algoritma mengevaluasi polinom yang lebih baik dapat dibuat dengan metode Horner berikut:

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \ldots + x(a_{n-1} + a_n x)))\ldots))$$

```
function p2(input x : real) → real
{ Mengembalikan nilai p(x) dengan metode Horner}

Deklarasi
  k : integer
  b1, b2, ..., bn : real

Algoritma
  bn ← an
  for k ← n - 1 downto 0 do
     bk ← ak + bk+1 * x
  endfor
  return b0
```

Hitunglah berapa operasi perkalian dan penjumlahan yang dilakukan oleh algoritma diatas, Jumlahkan kedua hitungan tersebut, lalu tentukan kompleksitas waktu asimptotik (Big-O)nya. Manakah yang terbaik, algoritma p atau p2?

Nama : M. Ilham Habib.
NPM : 1908101B0018
Kelas : B.

1. $T(n)$: Deretgeometri

$S_n = \frac{a(r^n-1)}{r-1} = \frac{2(2^n-1)}{2-1} = 2 \cdot 2^n - 2$

$T(n) = O(2^n)$ dan $f(n) = 2^n$

$T(n) \leq C \cdot f(n)$

$2^{n+1} - 2 \leq C \cdot 2^n$ | $n_0 = 1$ kk $C \geq 1$

$: 2^n$

$\frac{2^{n+1}}{2^n} - \frac{2}{2^n} \leq C$

$2 - \frac{2}{2^n} \leq C$; $n_0 = 1$

$2 - 1 \leq C$

$1 \leq C$.

2. $T(n) = pn^2 + qn + r$

- $O(n^2) \rightarrow$ Big O

$T(n) \leq C \cdot f(n)$

$pn^2 + qn + r \leq C \cdot n^2$ | $n_0 = 1$ kk $p+q+r \leq C$

$: n^2$

$p + \frac{q}{n} + \frac{r}{n^2} \leq C$; $n_0 = 1$

$p + q + r \leq C$.

- $\Omega(n^2) \rightarrow$ Big Ω

$Tn \geq C \cdot f(n)$

$pn^2 + qn + r \geq C \cdot n^2$ | $n_0 = 1$ kk $p+q+r \geq C$

$: n^2$

$p + \frac{q}{n} + \frac{r}{n} \geq C$; $n_0 = 1$

$p + q + r \geq C$.

- Karena Big O = Big Ω = n^2
  Maka Big Θ = n^2

3. $T(n) = O(n) + O(n) + O(n) + O(1)$
$= O(n^3) \rightarrow f(n)$

- Big-O = $O(f(n)) = O(n^3)$
- Big-Ω = $\Omega(f(n)) = \Omega(n^3)$

Karena Big-O = Big-Ω
Maka Big-Θ = $\Theta(f(n)) = \Theta(n^3)$

4.
```
for i ← 1 to n do      O(n)
  for j ← 1 to n do    O(n)
    hasil [i,j] ← a[i,j] + b[i,j]
  endfor
endfor
```

$T(n) = O(n) + O(n)$
$= O(n^2) \rightarrow f(n)$

- Big-O = $O(f(n)) = O(n^2)$
- Big-Ω = $\Omega(f(n)) = \Omega(n^2)$
- Big-O = Big-Ω maka Big-Θ = $\Theta(n^2)$

5.
```
for i ← 1 to n do      O(n)
    b[i-1] ← a[i-1]
endfor
```

$T(n) = O(n) \rightarrow f(n)$

- Big-O = $O(n)$
- Big-Ω = $\Omega(n)$
- Big-Θ = $\Theta(n)$

6. a).

Pass	Jumlah operasi
1	n-1
2	n-2
3	n-3
⋮	⋮
n	1

$T(n) = (n-1) + (n-2) + \dots + 1$
$= \frac{n(n-1)}{2}$
$= \frac{(n^2-n)}{2}$

b). Maks pertukaran elemen = $\frac{n(n-1)}{2}$

c). Kompleksitas waktu asimptotik.

Big-O → $T(n) = \frac{n^2-n}{2} = O(n^2)$

Big-Ω → $T(n) = n^2 + n = \Omega(n^2)$

Big-Θ → $T(n) = n^2 + n = \Theta(n^2)$

Nama : M. Ilham Habib.
NPM : 1408101800 18
Kelas : B.

[7] a. Algoritma A : $O(\log 8)$
b. " B : $O(8 \log 8)$
c. " C : $O(64)$

yang paling cepat adalah Algoritma A karena semakin kecil angka didalam kurung /f(n) maka semakin sedikit operasi yang dikerjakan.

[8] Algoritma P → Jumlah = n kali
kali = n kali

$T(n) = n+n = 2n = n$

Algoritma P_2 → $T_2(n) = 1+n$
$= O(n)$

maka P dan P_2 sama-sama baik karena Big-Onya sama // $O(n)$.