Muhammad Hanif

140810170033

SOAL DAN JAWABAN UTS

1. **Soal 1. a.**

```
For j = 1 to n-1
    k = j                  \\ n-1
    For i = j + 1 to n
       if a[i] < a[k] then
            k=i            \\ 2 ∑ [ i=2 sampai n](n/2 (2+n)) = 2n + n2
       endif
     endfor
  tm=a[j]      \\ operasi assignment
  a[j] = a[k]  \\ operasi assignment
  a[k] = tm    \\ operasi assignment
endfor
```

T(n) = (n-1) + 2n+n2 + (n-1) + (n-1) + (n-1)
= 6n + n2 - 4 = O (n2)

Kompleksitas O

T(n) <= c . f(n)
n2 + 6n – 4 <= c . n2
1 + 6/n – 4/n2 <= c
Misal $\mathbf{n_0 = 1}$
1 + 6 – 4 <= c
3 <= c
c >= 3

Big O terbukti karena n_0 bernilai positif dan c >= 1

1.b.

```
for i=0 to n-1
    for j=0 to n-1
        c[i,j] = 0                    \\ n-1
        for k=0 to n-1
            cij= d[i,k] and b[k,j]  \\ operasi assignment
            c[i,j] = c[i,j] or cij      \\ operasi assignment
         endfor
    endfor
endfor
```

T(n) = (n-1)(n-1)(n-1)
= (n2 -2n + 1)(n-1)
= (n3 – 3n2 + 3n – 1)
T(n) = O(n3)

Kompleksitas O
T(n) <= c.f(n)
(n3 – 3n2 + 3n – 1) <= c. n3
1 – 3/n + 3/n2 – 1) <= c
misal n0 = 1
1 – 3 + 3 – 1 <= c
c >= 0

Big O terbukti karena n_0 bernilai positif dan c >= 0

1. Algoritma berikut menggunakan data a[1..n]
   a)
```
Ada = 0;
Kx=1;
Input br;
For (i=1; i<n; i++){
    If(a[i] == br && (!ada)){
            Ada = 1;
            kx = 1;
            I = n+1;
    }

    }
```

T(n) = n/2(n+1)
T(n) = O (n)

Kompleksitas O
T(n) <= c. f(n)
n/2(n+1) <= c. n
½(n+1) <= c
misal n_0 = 1
½ (1+1) <= c
c >= 1

Big O terbukti positif dengan n_0 = 1 , c >= 1 maka Big O = O(n)

   b)
```
L = 1;
R=n;
Ada = 0;
```

```
Input br;
While((L<= R) && (!ada)){
    M = (L+R) div 2;
        If (a[m] == br)
            Ada = 1;
        Else if (br<a[m])
            R = m+1;
        Else
            L=m+1;
}
```

Dari m = (L+R) div 2 didapat
loop 1 n/2
loop 2 n/2
loop k $n/2^k$

$n/2^k = 1$
$n = 2^k$
k = $\log_2 n$

oleh karena itu
M = (L+R) div 2; \\ $\log_2 n$
If (a[m] == br) \\$\log_2 n$
Ada = 1; \\ 1
Else if (br<a[m]) \\ $\log_2 n$
R = m+1 atau L=m+1 \\ $\log_2 n$

T(n) = 1 + 1 + 1 + $\log_2 n$ + $\log_2 n$ + 1 + $\log_2 n$ + $\log_2 n$
= 4 + 4/ $\log_2 n$
T(n) = O($\log_2 n$)

Kompleksitas O
T(n) <= c. f(n)
4 + 4($\log_2 n$) <= c. $\log_2 n$
4 + 4 / $\log_2 n$ <= c
misal n_0 = 2
4 + 4 <= c
8 <= c
c >= 8

Big O terbukti positif dengan n_0 = 2, c >= 8 maka Big O = O($\log_2 n$)

a) .

b) Komputer A mengeksekusi 10^9 instruksi/detik, computer B 10^7 instruksi/detik. Komputer A akan menggunkana algoritma (a.a) dan computer B menggunakan algoritma (2,b). Datanya sebanyak 10^8. Hitung *running time* masing-masing algoritma menggunakan kompleksitas O, dan algoritma yang lebih baik

Menghitung running time :
Running time = jumlah instruksi / kecepatan eksekusi
Running time algoritma a :
$10^8/10^9 = 10^{-1}$ detik

Running time algoritma b:
$(\log_2 10^8)/10^7 = 26.57542 \times 10^{-7} = 2{,}657542 \times 10^{-8}$ detik$^-$

Algoritma yang lebih baik adalah algoritma yang memiliki efisiensi waktu lebih baik, yaitu algoritma b