1. Cari Kompleksitas O, T(n), c dan n0
2. for j=1 to n-1

k=j

for i=j+1 to n

if a[i]<a[k] then

k=i

endif

endfor

tm = a[j]

a[j] = a[k]

a[k] = tm

endfor

endfor

**Solusi**

* loop for j =1 🡪 n-1 kali
* loop for i=j+1

|  |  |  |
| --- | --- | --- |
| j | i=j+1 | Operasi perbandingan if |
| 1 | 2 | n-1 |
| 2 | 3 | n-2 |
| 3 | 4 | n-3 |
| 4 | 5 | n-4 |
| .  .  . | .  .  . | .  .  . |
| n-1 | n | n-(n-1) = 1 |

* operasi perbandingan pada loop for i=j+1

(n-1) + (n-2) + (n-3) + … + 1 =

T(n) =

Kompleksitas – O

Sehingga, T(n) = O(n2) dengan c > 0 dan n0 =1.

b) for i=0 to n-1

for j=0 to n-1

c[i,j]=0

for k=0 to n-1

cij=d[i,k]and b[k,j]

c[i,j]=c[i,j] or cij

endfor

endfor

endfor

**Solusi**

* loop for k=0 🡪 2n3
* loop for j=0 🡪 n2

T(n) = 2n3 + n2 = O(n3)

Kompleksitas O

T(n) < c . f(n)

2n3 + n2 < c . n3

2 +

Sehingga T(n) = O(n3) dengan dan

1. Cari masing masing T(n), kompleksitas O, c dan n0.
2. ada = 0; 🡪 1 kali

kx=1; 🡪 1 kali

input br; 🡪 1 kali

for (i=1;i<n+1;i++){

if(a[i]==br&&(!ada)){ 🡪 n kali

ada=1; 🡪 1 kali

kx=i; 🡪 1 kali

i=n+1; 🡪 1 kali

}

}

**Solusi**

loop for i=1 🡪 n kali

T(n) = 6 + n = O(n)

Kompleksitas – O

7 < c

Sehingga T(n) = O(n) dengan 7 < c dan

1. L=1; 🡪 1 kali

R=n; 🡪1 kali

ada=0; 🡪1 kali

input br; 🡪1 kali

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;

}

**Solusi**

Loop while

Loop -1 🡪

Loop -2 🡪

Loop -3 🡪

Loop -k 🡪

Sehingga, pada kasus terburuk:

k = 2log n

T(n) = 4 ( 2log n ) + 4 = O( 2log n )

Kompleksitas – O

4 ( 2log n ) + 4 < c . 2log n

4 +

8

Sehingga, T(n) = O(2log n) dengan 8 dan

2.b. Komputer A

Running time algoritma (2.a) =

Komputer B

Running time algoritma (2.b) =

Karena running time algoritma (2.b) lebih cepat maka algoritma tersebut lebih baik walaupun komputer B mengeksekusi instruksi lebih lambat daripada komputer A.