Jeremy Edbert Widjaja
24060119130071

UTS Analisis dan Strategi Algoritma

1 a) Divide and Conquer

```
procedure SortMerge (input/output T: Array of character, input i,j : integer)

Deklarasi
    x : integer

Algoritma
    if i < j then
        x ← i+j div 2
        SortMerge (T, i, x)
        SortMerge (T, x+1, j)    {divide}
        Merge (T, i, x, j)
    endif

procedure Merge (input/output T: Array of character, input left, mid, right : integer)

Deklarasi
    S: Array of character
    i, a, b : integer

Algoritma
    a ← left
    b ← mid +1
    i ← left
    while ( a ≤ mid ) and (b ≤ right) do
        if T_a < T_b then
            S_i ← T_a
            a ← a + 1
        else
            S_i ← T_b
            b ← b + 1
        endif
        i ← i+1
    endwhile
```

```
while (a ≤ mid) do
    S_i ← T_a
    a ← a+1
    i ← i+1
endwhile

while (b ≤ right) do
    S_i ← T_b
    b ← b+1
    i ← i+1
endwhile

{conquer}
i traversal left..right
    T_i ← S_i
endtraversal
```

Kompleksitas

$$T(n) = \begin{cases} a & , n=1 \\ 2T(n/2)+cn & , n>1 \end{cases}$$

$$T(n) = 2T(n/2)+cn$$
$$= 2(2T(n/4)+cn/2)+cn = 4T(n/4)+2cn$$
$$= 4(2T(n/8)+cn/4)+2cn = 8T(n/8)+3cn$$

asumsi $n=2^k \longrightarrow \frac{n}{2^k}=1 \Leftrightarrow k=\log_2(n)$

$$= 2^k(T(n/2^k))+kcn$$

$$T(n) = nT(1)+cn^2\log n$$
$$= an + cn^2\log n$$
$$= O(n^2\log n)$$ ⟶ ambil derajat tertinggi

b) Decrease and Conquer

```
procedure SelectionSort(input/output arr: Array of Character, input i, j: integer)
    if i < j then
        SearchReplace(arr, i, j)
        SelectionSort(arr, i+1, j)
    endif

procedure SearchReplace(input/output arr: Array of Character, input i, j : integer)

Deklarasi
    idxmin, k, tmp : integer

Algoritma
    idxmin ← i
    k traversal [i+1 ... j-1]
        if arr_k < arr_idxmin then
            idxmin ← k
        endif
    endtraversal
```

{swap}

$tmp \leftarrow arr_i$

$arr_i \leftarrow arr_{idxmin}$

$arr_{idxmin} \leftarrow tmp$

Kompleksitas

1) Proses rekursif memakan waktu 1 elemen kurang dari sebelumnya
2) Pencarian nilai indeks minimum pada array memakan waktu sebanyak n kali.

Sehingga

$$T(n) = \begin{cases} a & , n = 1 \\ T(n-1) + cn & , n > 1 \end{cases}$$

$$\begin{aligned} T(n) &= T(n-1) + cn \\ &= cn + (c(n-1) + T(n-2)) \\ &= cn + (c(n-1) + (c(n-2) + T(n-3))) \\ &= cn + c(n-1) + c(n-2) + \dots + 2c + T(1) \end{aligned}$$

$$T(1) = a$$

$$\begin{aligned} &= c(n + (n-1) + (n-2) + (n-3) + \dots + 2) + a \\ &= c\left(\frac{n \times (n-1)}{2}\right) + a \\ &= c\left(\frac{n^2 - n}{2}\right) + a \quad = O(n^2) \end{aligned}$$

2 a) Using exhaustive search

1) Enumerasikan himpunan bagian dengan menggunakan total yang diassign pada orang ke-i dan job ke-j
2) Evaluasi tiap kemungkinan yang ada

Himpunan	Total biaya
(1,1) (2,2) (3,3)	9 + 4 + 1 = 14
(1,1) (2,3) (3,2)	9 + 3 + 8 = 20
(1,2) (2,1) (3,3)	2 + 6 + 1 = 9
(1,2) (2,3) (3,1)	2 + 3 + 5 = 10
(1,3) (2,2) (3,1)	7 + 4 + 5 = 16
(1,3) (2,1) (3,2)	7 + 6 + 8 = 21

3) Pilihan solusi terbaik adalah (1,2) (2,1) (3,3) dengan biaya = 9

b) Jika dilihat patternnya, matriks 3 X 3 menghasilkan kemungkinan sebanyak $3! = 3 \cdot 2 \cdot 1 = 6$
maka jika terdapat n job dan n orang, akan ada n! kemungkinan.

Dalam penghitungan biaya tiap elemen matriks, akan membutuhkan kompleksitas waktu $O(n)$
sehingga algoritma ini memerlukan kompleksitas waktu $O(n \cdot n!)$