# Kompleksitas Waktu dan Efisiensi Algoritma 2

wijanarto

# Pengantar

- Pada pertemuan sebelumnya sudah di bahas dasar teoritis, alasan menentukan kompleksitas waktu suatu algoritma dalam bentuk fungsi g(n) , klasifikasinya serta contoh-contohnya
- Dua bagian selanjutnya akan membicarakan standar perhitungan kompleksitas waktu dan efisiensi algoritma yang di dasarkan atas struktur dasar algoritma, procedure atau function call baik secara rekursi atau bukan
- Bebas Bahasa : untuk contoh dapat dalam bentuk bahasa apapun atau notasi tertentu, yang di pandang adalah struktur algoritma, procedure call, rekursif

# STRUKTUR PROGRAM (SP)

- Aturan dasar analisa terhadap SP akan menyangkut banyak langkah, yang di pengaruhi oleh :
  - Banyak operator yang di gunakan, asumsi 1 operator adalah 1 langkah
  - Procedure
    - Built-In atau User Define
  - Kontrol langkah
    - Sekuensial (sequential)
    - struktur kondisi (conditional)
    - Perulangan (loop)

# Operator dan Assignment

- Assignment  boleh di hitung langkahnya, tergantung kesepakatan
- Aritmatika
  - Div, *,-,+,  di hitung 1 langkah
  - Mod, Pemangkatan,  di hitung 2 langkah
- Relasi
  - <,>,≤,≥,≠,=,  dihitung satu langkah
- Logika
  - And, Or, Not,  di hitung 1 langkah

# PROCEDURE/FUNCTION CALL

- Pada bagian ini analisa lebih cenderung di pandang bagaimana suatu operasi di lakukan pada level bawah (low level), yaitu pada level pemroses melakukan operasi secara mikro di prosesor dan hal ini juga tergantung pada compiler yang di pergunakan, apakah optimasi dapat dilakukan/diberikan atau tidak.
- int x=5*3, dianggap 1 langkah, karena di dalam ekspresi ada operator dan jumlahnya hanya 1 (*)
- int x=5*3+4, dianggap 2 langkah karena di dalam ekspresi tersebut ada 2 operator (*,+), kalau di detailkan akan menjadi seperti int x=5*3;x=x+4.

# PROCEDURE/FUNCTION CALL

- Penggunaan **built-in** procedure/function adalah di anggap 1 langkah, misal ada pemanggilan seperti berikut *sin(x)*, maka di anggap 1 langkah atau sin(x*2) diangap 2 langkah
- Jumlah parameter pada built-in function jika lebih dari satu akan di hitung, misal, *power(2,10)* ada 2 langkah
- Untuk **User Define procedure** atau **function** akan di bahas pada bagian lain
- Karena melibatkan bentuk dari struktur dasar algoritma yang di pakai maka penentuan kompleksitasnya di rumuskan sendiri.

# Contoh (pascal)

```
FUNCTION Sinus(x) : real;
BEGIN
  FOR i : = 0 TO 1000
    IF i mod 2 = 0 THEN d:= 1
    ELSE
     d:= - 1;
  jum:=jum + d * exp ((2 * i + 1) * ln
  (x)) div fakt (2 * i + 1);
  sinus := jum ;
END
```

- **Waktu tempuh = space + banyak langkah ????**

# Contoh (c)

```
float sinus (x)
{
  for (i=0;i<=1000;i++)
  if (i%2==0) d=1;
  else
     d= -1;
  jum=jum + d * exp((2 * i + 1) * ln
  (x)) /fakt(2 * i + 1)
  return jum;
}
```

- **Waktu tempuh = space + banyak langkah ????**

# Contoh (notasi)

```
FUNCTION Sinus(x)→Real
  FOR i←0 TO 1000 Do
    IF (i mod 2 = 0) THEN d ← 1
    ELSE d ← -1
  jum←jum + d * exp ((2 * i + 1) *
  ln (x)) div fakt (2 * i + 1)
  →jum
```

- **Waktu tempuh = space + banyak langkah ????**

# SEKUENSIAL

- Misal ada suatu statemen sebagai berikut,

Statemen s1 dengan t1 langkah

Statemen s2 dengan t2 langkah

Maka banyak langkah statemen gabungannya adalah t1+t2 , atau

$S_1$ banyak langkah $P_1$

$S_2$ banyak langkah $P_2$

$S_3$ banyak langkah $P_3$

. .

. .

$S_n$ banyak langkah $P_n$

Total banyak langkah blok-blok statement tersebut $\sum_{i=1}^{n} P_i$

$S_i$ bisa berupa : assigment, procedure call, percentage, loop, dsb.

# contoh

| | | |
|---|---|---|
| $x \leftarrow x * y$ | operasi 1 | = 1 |
| $y \leftarrow a * \sin(x)$ | operasi 1, proc 1 | = 2 |
| read (b) | proc 1 | = 1 |
| write (x + y + b) | proc 1, operasi 2 | = 3 |
| | Banyak Langkah | = 7 |

# PENCABANGAN/BRANCHING

```
if (k) then s1 else s2
```

k = expresi kondisi dengan banyak langkah

$S_1$,

$S_2$ = blok statement dengan banyak langkah $P_1$ dan $P_2$

$S_1$ dan $S_2$ dapat berupa nested maupun cascade if conditional atau berisi loop

# PENCABANGAN/BRANCHING

Misal :

Expressi kondisi mempunyai *k* langkah

$S_1$ mempunyai $P_1$ langkah

$S_2$ mempunyai $P_2$ langkah

**maka**

Langkah **terburuk** adalah **k+max($P_1$,$P_2$)**, dan

Langkah **terbaik** adalah **k+min($P_1$,$P_2$)**

*Yang digunakan dalam menentukan banyak langkah dalam suatu pencabangan adalah kasus terburuk yaitu* ***k+max($P_1$,$P_2$)***

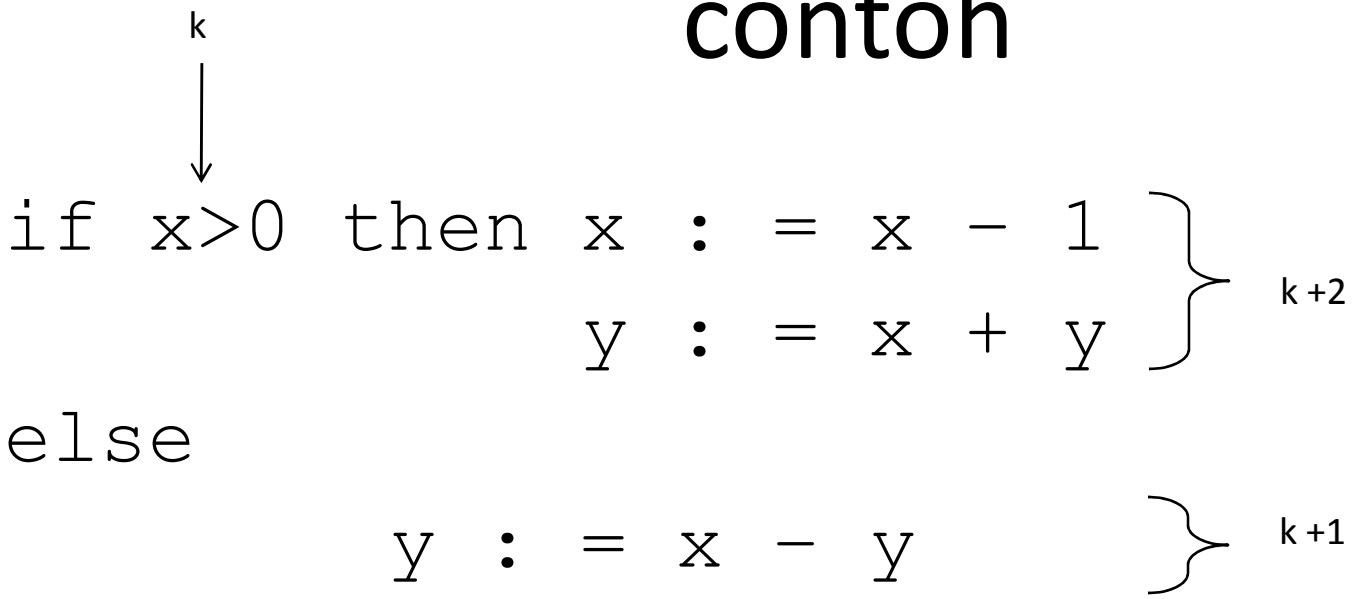
# contoh

- Not ( P AND Q)  mempunyai langkah sebanyak 2
- Not (x > 0 AND y > 0) mempunyai langkah sebanyak 4
- C $n^k \in \theta (n^k)$ C = kombinasi

$$\lim_{n\to\infty} \frac{Cn^k}{n^k} = C$$

- $Cn^k \in O(n^k) \cap \Omega(n^k)$

# contoh




- Banyak langkah kondisi I adalah 2
- Banyak langkah kondisi II adalah 1
- Kasus terjelek adalah k + max ($P_1$, $P_2$) = 1 + 2 = 3
- Dengan demikian banyak langkah untuk pencabangan diatas dihitung 3.

# contoh

```
scanf(“%d”,&x);                    //1 langkah
y=x+5;                             //1 langkah, s0=2
if (x>0) {                         //kondisi = 1
   y=y-5;
   x=x-y; //s1 karena terletak dalam blok
          //(ada 2 statemen) = 2 langkah
}else
  x=abs(x);//s2 1 langkah
```

- Hasil analisisnya, banyak langkah dari kode diatas adalah s0=2,s1=2,s2=1,k=1, s0+k+max(s1,s2)=2+1+2=5 langkah

# LOOPING

- Loop yang dapat di hitung hanya for loop, sedang while dan do while atau repeat until mungkin tidak dapat di hitung karena mungkin dalam loop bentuk tersebut terdapat unpredictable input yang menyebabkan kebocoran loop.
- Dengan demikian loop bentuk tersebut harus dapat di translasikan ke dalam bentuk for loop

X tidak dapat di ketahui akan di baca berapa kali, sedang untuk for loop akan mudah di ketahui

```
For (i=awal i=akhir;i++){
Input(i);
i=i*5;
}
```

Nilai i hanya bisa di gunakan dalam statemen di bawah for (bersifat local pada kalang for)

Jadi for loop lebih mudah di analisis.

# LOOPING

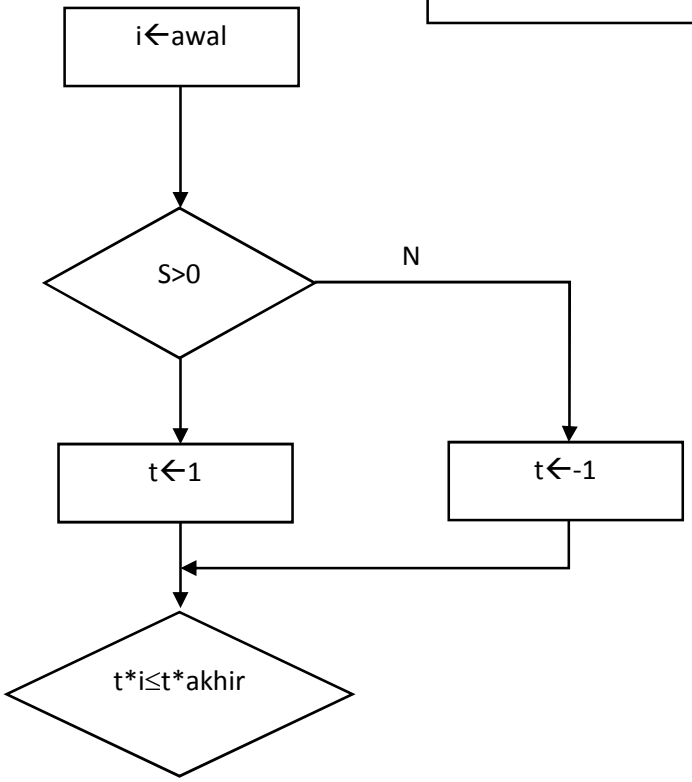
- Unpredicatble value dalam loop

```
x=5
while(x>0){
   .
   .
   Input(x)
}
```

```
do{
   .
   .
   Input(x)
}while(x>0)
```

# BENTUK FOR LOOP

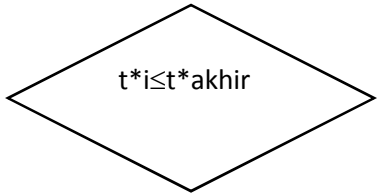
```
for(i=awal;i=akhir;i++){
    Step S
    Statemen(i)
}
```



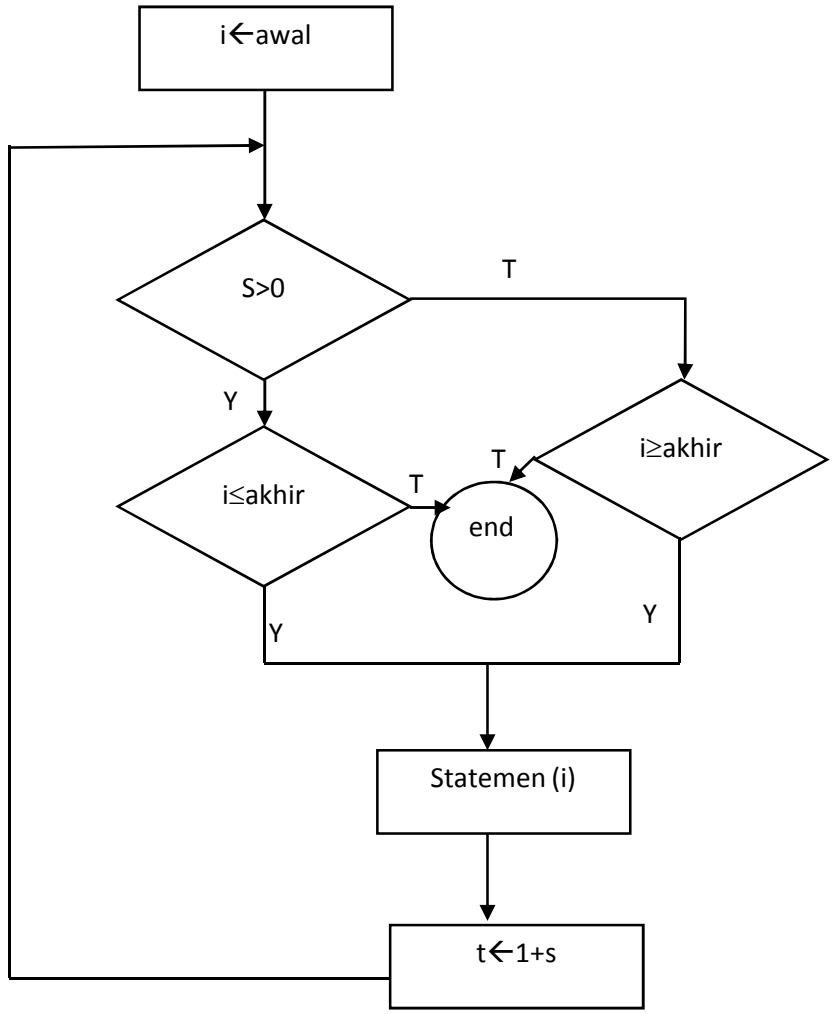

Di MATLAB

I← awal

t ← sign(s)



# atau



# Kasus 1

## Banyak Langkah untuk Statement FOR

Counter : integer

Step : **1 = default**

Statement S mempunyai banyak langkah yang tidak bergantung nilai counter

FOR *counter : awal* TO *akhir atau for (awal;akhir;counter)*

S

S dieksekusi sebanyak *akhir – awal +1* kali

Note :

Counter ≤ Akhir , S dieksekusi sebanyak *akhir – awal + 2* kali

Counter = counter + 1 , S dieksekusi sebanyak *akhir – awal + 1* kali

# rumus

- Banyak Langkah :

$$(akhir - awal + 2) + (akhir - awal + 1)(p + 1)$$

p = banyak langkah dalam statement loop.

# contoh

Berapa banyak langkah dari

FOR i = 1 TO n

x : = x + 5

y : = y + x

Penyelesaian :

Langkah = (akhir – awal + 2) + (akhir – awal + 1) (p + 1)

= (n – 1 + 2) + (n – 1 + 1) (2 + 1)

= (n + 1) + (n)(3)

= n + 1 + 3n

= 4n + 1

# Kasus 2

## Dengan STEP = s

s dieksekusi sebanyak

$$\left\lfloor \frac{akhir - awal}{s} + 1 \right\rfloor$$

atau ((akhir – awal) **div** s + 1)

Rumus

$$\left\lfloor \frac{\boldsymbol{akhir - awal}}{\boldsymbol{s}} + \mathbf{2} \right\rfloor + \left\lfloor \frac{\boldsymbol{akhir - awal}}{\boldsymbol{s}} + \mathbf{1} \right\rfloor + (\boldsymbol{p} + \mathbf{1})$$

# contoh

- Berapa banyak langkah dari
- FOR i:= j TO n STEP 3
- x := x + i
- y := y + j

$$\left\lfloor \frac{akhir - awal}{s} + 2 \right\rfloor + \left\lfloor \frac{akhir - awal}{s} + 1 \right\rfloor \quad (p + 1)$$

$$\left(\frac{n-j}{3} + 2\right) + \left(\frac{n-j}{3} + 1\right)(2+1)$$

$$\left(\frac{n-j}{3} + 2\right) + \left(\frac{n-j}{3} + 1\right)(3)$$

$$\left(\frac{n-j}{3} + 2\right) + \left(3.\frac{n-j}{3} + 3\right)$$

$$\frac{n-j}{3} + 2 + 3.\frac{n-j}{3} + 3$$

- $4\frac{n-j}{3} + 5$ Jadi, $4\frac{n-j}{3} + 5 \in O(n)$ $P_d\,(n) \in O\,(n^d)$ P = Polinomial, d = Derajat

# contoh

- Berapa banyak langkah dari
- FOR i = 0,5 TO 7,1 STEP 0,3
- x := x + i
- y := y + j

$$\left\lfloor \frac{akhir - awal}{s} + 2 \right\rfloor + \left\lfloor \frac{akhir - awal}{s} + 1 \right\rfloor \quad (p + 1)$$

$$\left( \frac{7{,}1 - 0{,}5}{0{,}3} + 2 \right) + \left( \frac{7{,}1 - 0{,}5}{0{,}3} + 1 \right)(2 + 1)$$

$$\left( \frac{6{,}6}{0{,}3} + 2 \right) + \left( \frac{6{,}6}{0{,}3} + 1 \right)3$$

- (22 + 2) + (22 + 1) . 3
- 24 + 23. 3
- 24 + 69
- 93

# **Kasus 3** :
# S bergantung nilai Counter

```
FOR i = 1 TO n
    x : = x + y
    FOR j : = i TO n
        y : = i + j
```

Outer Loop

S = statement dalam outer loop

Inner Loop

Inner Loop

Banyak langkah = (akhir – awal + 2) + (akhir – awal + 1) (p + 1)

= ((n – i) + 2) + ((n – i) + 1) (1 + 1)

= ((n – i) + 2) + ((n – i) + 1) . 2

= ((n – i) + 2) + 2(n – i) + 2

= 3 (n – i) + 4

= 3n – 3i + 4

(P(i)) = Banyak Langkah dalam S + banyak langkah inner loop

= 1 + 3n – 3i + 4

= 3n – 3i + 5

# Rumus Outer Loop (Lanjutan)

Banyak langkah= (akhir – awal + 2) + (akhir – awal + 1) .s + $\sum_{i=1}^{n} P(i)$

$((n-1)+2)+((n-1)+1).1+\sum_{i=1}^{n}(3n-3i+5)$

$2n+1+\sum_{i=1}^{n}3n-\sum_{i=1}^{n}3i+\sum_{i=1}^{n}5$

$2n+1+3n.n-3.\left(\frac{1}{2}n(n+1)\right)+5.n$

$2n+1+3n^2-\frac{3}{2}n^2-\frac{3}{2}n+5n$

$4n+2+6n^2-3n^2-3n+10n$

$3n^2+11n+2$

$3n^2+11n+1 \in O(n^2)$

$$\sum_{i=1}^{n} i = \left(\frac{1}{2}n(n+1)\right)$$

# lanjutan

- FOR i : = awal TO akhir STEP s
- S(i)
- 
- Misal P(i) banyak langkah S(i) maka banyak langkah loop tersebut

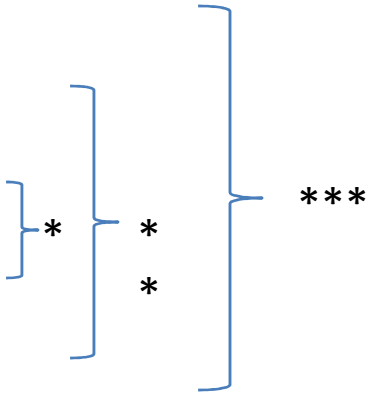
$$2\left\lfloor \frac{akhir - awal}{s} \right\rfloor + 3 + \sum_{i=awal,s}^{akhir} P(i)$$

- Dimana

$\sum_{i=awal,s}^{akhir} P(i)$ =P.awal + (P.awal+s) + (p.awal+2.s) +…+(p.akhir)

# Tugas

```
Diketahui  :
Read (x)
y: = 0
FOR i:= 1 TO n
  x:= x + y
  FOR j:= i + 1 TO n^2
     y:= y + i + j
  write (x,y)
write (x,y)
```




Tentukan T(n) = …. ∈ O (…)

# Solusi Alternatif 1

*Penyelesaian :*

Banyak langkah (*) = (akhir – awal + 2) + (akhir – awal + 1) (p + 1)

$= (n^2 – (i + 1) + 2) + (n^2 – (i + 1) + 1) (2 + 1)$

$= ( n^2 – i – 1 + 2) + (n^2 – i – 1 + 1) 3$

$= n^2 – i + 1 + 3n^2 – 3i$

$= 4n^2 – 4i + 1$

(akhir-awal + 2) + (akhir – awal + 1) (p + 1) = 2(akhir – awal) + 3 + p (akhir – awal + 1)

Banyak langkah (*) = 2(akhir – awal) + 3 + p (akhir – awal + 1)

= 2 ($n^2$ – (i + 1)) + 3 + 2 ($n^2$ – (i + 1) + 1)

= 2 ($n^2$ – i – 1) + 3 + 2 ($n^2$ – i )

= 2$n^2$ – 2i – 2 + 3 + 2$n^2$ – 2i

= 4$n^2$ – 4i + 1

Banyak langkah (**) = 2 + Banyak langkah (*)

= 2 + 4$n^2$ – 4i + 1

= 4$n^2$ – 4i + 3

Banyak langkah (***) = 2 (akhir – awal) + 3 + $\sum_{i=awal,s}^{akhir} P(i)$ +(akhir-awal +1)

= 2 (n – 1) + 3 + $\sum_{i=1}^{n} 4n^2 - 4i + 3$ +(akhir-awal +1)

= 2n – 2 + 3 + $\sum_{i=1}^{n} 4n^2$ – $\sum_{i=1}^{n} 4i$ + $\sum_{i=1}^{n} 3$ +(akhir-awal +1)

= 2n + 1 + 4$n^2$ . n – 4$\left(\frac{1}{2}n(n+1)\right)$ + 3. n +(akhir-awal +1)

= 4$n^3$ + 5n + 1 – 2$n^2$ – 2n +(akhir-awal +1)

= 4$n^3$ – 2$n^2$ + 3n +1 +(akhir-awal +1)

Banyak Langkah Program = T(n) = 3 + Banyak langkah (***)

= 3 + 4$n^3$ – 2$n^2$ + 3n + 1

= 4$n^3$ – 2$n^2$ + 3n + 4

4$n^3$ – 2$n^2$ + 3n + 4 $\in$ O ($n^3$)