Analisis Algoritma

Pertemuan 3

Selasa, 10 Maret 2020

Rabu, 11 Maret 2020

REVIEW DULU YUK!

3 macam kompleksitas waktu :

1. *Best Case*
2. *Average Case*
3. *Worst Case*

MENGAPA *WORST CASE* DIUTAMAKAN UNTUK DIHITUNG?

1. Merupakan *upper bound* dari *running time* untuk input apapun
2. Untuk beberapa algoritma, *worst case* sering terjadi. contohnya pada kasus pencarian
3. Pada kasus *average case*, umumnya lebih sering seperti *worst case*, Karena :

   Average case = worst case (fungsi kuadratik dari n)

WORST CASE DALAM KOMPLEKSITAS WAKTU ASIMTOTIK

*Dengan menggunakan **Big-O Notation***

Contoh :

$$T(n) = 2n^2 + 6n + 1$$

- *Untuk n yang besar, pertumbuhan T(n) sebanding dengan n^2*
- *Suku 6n + 1 tidak berarti jika dibandingkan dengan $2n^2$, dan boleh diabaikan sehingga T(n) = $2n^2$ + suku-suku lainnya*
- *Koefisien 2 pada $2n^2$ boleh diabaikan, sehingga T(n) = $O(n^2)$ -> **Kompleksitas Waktu Asimtotik***

Big-O Notation

√123

STUDY HARD!

+ × ÷

DEFINISI BIG-O NOTATION

T(n) = O(f(n)), *artinya T(n) berorde paling besar f(n) bila terdapat konstanta C dan* n_0

Untuk $n \geq n_0$,

$$T(n) \leq C \, . \, f(n)$$

Jika n dibuat semakin besar, waktu yg dibutuhkan tidak akan melebihi C dikalikan f(n). Maka f(n) adalah upper bound.

Dalam pembuktian Big-O, diperlukan nilai n_0 *dan nilai C agar terpenuhi kondisi*

$$T(n) \leq C \, . \, f(n)$$

BIG-O NOTATION POLINOMIAL BERDERAJAT N

Digunakan untuk memperkirakan kompleksitas dengan mengabaikan suku berorde rendah.

Contoh $T(n) = 3n^3 + 6n^2 + n + 8 = O(n^3)$ dinyatakan pada :

TEOREMA 1

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

Artinya:

Mengambil suku paling tinggi derajatnya yang diartikan laju pertumbuhan lebih cepat dibandingkan yang lainnya.

BIG-O NOTATION POLINOMIAL BERDERAJAT N

TEOREMA 2

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

(a) (i) $T_1(n)+T_2(n) = O(\max(f(n), g(n)))$

(ii) $T_1(n)+T_2(n) = O(f(n)+g(n))$

(b) $T_1(n) \cdot T_2(n) = O(f(n)) \, O(g(n)) = O(f(n) \cdot g(n))$

(c) $O(cf(n)) = O(f(n))$, c adalah konstanta

(d) $f(n) = O(f(n))$

CONTOH SOAL

1. *Misalkan* $T_1(n) = O(n)$ *dan* $T_2(n) = O(n)$, *dan* $T_3(n) = O(mn)$, *Dengan m sebagai peubah, maka:*

   (a) $T_1(n) + T_2(n) = O(max(f(n,n^2) = O(n^2)$ *Teorema 2(a)(i)*

   (b) $T_2(n) + T_3(n) = O(n^2+mn)$ *Teorema 2(a)(ii)*

   (c) $T_1(n) . T_2(n) = O(n . n^2) = O(n^3)$ *Teorema 2(b)*

2. (a) $O(5n^2) = O(n^2)$ *Teorema 2(c)*

   (b) $n^2 = O(n^2)$ *Teorema 2(d)*

Aturan Kompleksitas Waktu Asimptotik

STUDY HARD!

CARA 1

Jika kompleksitas waktunya T(n) sudah dihitung, maka kompleksitas waktu asimptotik dapat ditentukan dengan mengambil suku yang mendominasi fungsi T & menghilangkan koefisiennya (Teorema 1)

Contoh :

Pada algoritma cariMax, T(n) = n-1 = O(n)

CARA 2

Big-O Notation:

Pengisian nilai, perbandingan, operasi aritmatika, read, write, pengaksesan elemen larik, memilih field dari record, dan pemanggilan fungsi membutuhkan waktu O(1). Cara penghitungan :

$$T(n) \leq C \,.\, (g(n)), \text{ dengan syarat nilai c dan n positif}$$

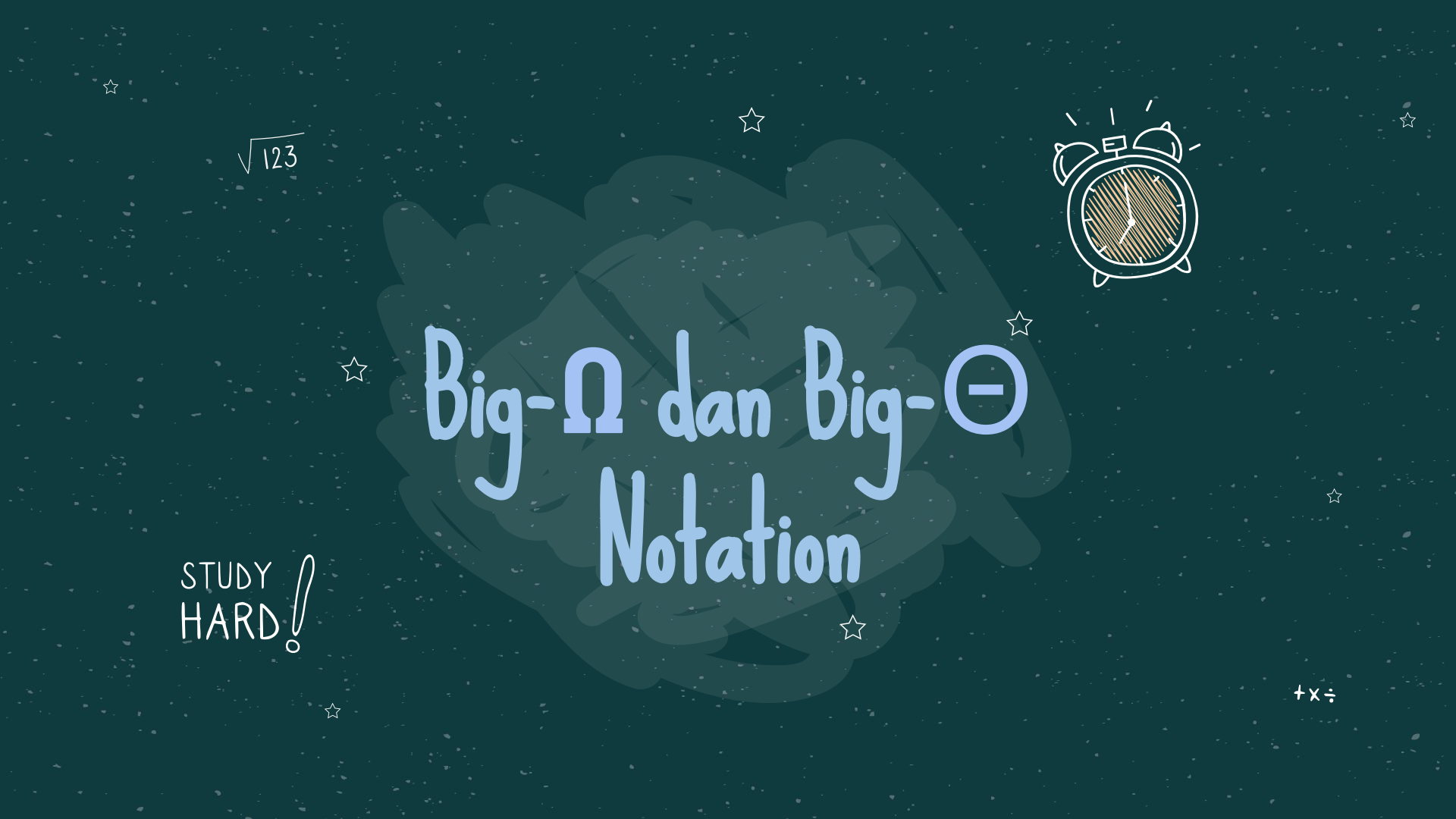
CONTOH SOAL

read (x)	*O(1)*
x <- x+1	*O(1) + O(1) = O(1)*
write (x)	*O(1)*

Kompleksitas waktu asimtotik algoritmanya O(1) + O(1) + O(1) = O(1)

Penjelasan :

O(1) + O(1) + O(1) = O(max(1,1)) + O(1)
= O(1) + O(1)
= O(max(1,1))
= O(1)


√123
Big-Ω dan Big-Θ
Notation
STUDY
HARD!
+x÷

Big-O hanya menyediakan upper bound.

Sedangkan untuk lower bound, dapat diperoleh dengan big-Ω dan big-Θ

DEFINISI

Big-Ω Notation:

T(n) = Ω(f(n)), artinya T(n) berorde paling kecil f(n) bila terdapat konstanta C dan n_0 sehingga :

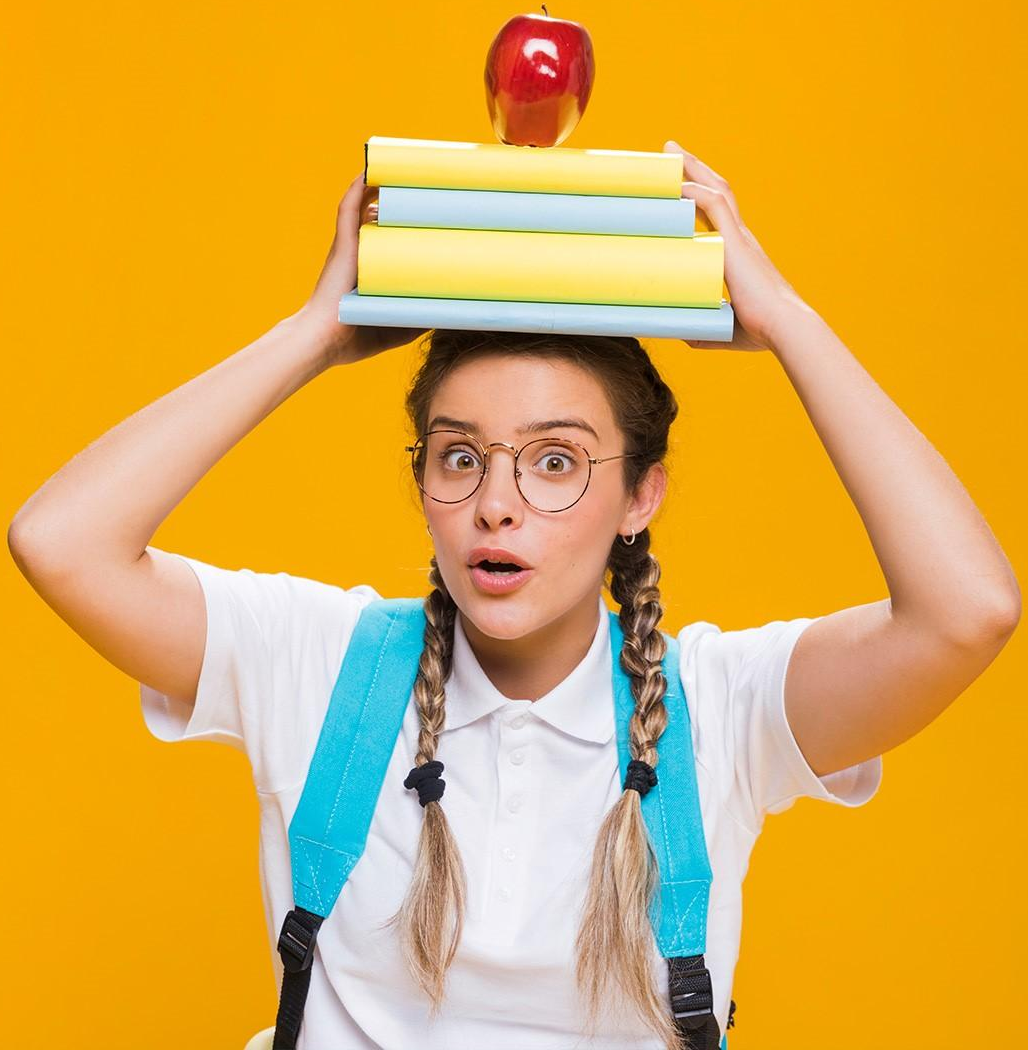
$$T(n) \geq C \,.\, (f(n))$$, *dengan syarat nilai c dan n positif*

Big-Θ Notation:

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*

LATIHAN KUY!!!



LATIHAN

Kerjakan di lab, worksheet pada Modul Praktikum 3 di Google Classroom

TUGAS

Buat Laporan Praktikum :

- *Selesaikan worksheet, program c++, dan bagian analisis di modul praktikum.*
- *Push laporan ke github masing-masing.*

  Nama repository : AnalgoKu
- *Kirim link repository kalian di Google Classroom*

Deadline : Hari sebelum praktikum, jam 22.00

THANKS!

See you next week.