# ANALISIS ALGORITMA

Ir. RISMAYANI, S.Kom., M.T., CBPA., CNET., C.GL., CIISA

# ASYMPTOTIC NOTATION

- **Notasi Asimtotik (*Asymptotic Notation*)**

- Notasi asimtotik merupakan himpunan fungsi yang dibatasi oleh suatu fungsi n Î N yang cukup besar.
- Fungsi : N → R (sering $R^+$)
- Notasi Asimtotik digunakan untuk menentukan kompleksitas suatu algoritma dengan melihat waktu tempuh algoritma. Waktu tempuh algoritma merupakan fungsi : N → $R^+$
- Menggambarkan karakteristik/perilaku suatu algoritma pada batasan tertentu (berupa suatu fungsi matematis)
- Dituliskan dengan notasi matematis yg dikenal dgn notasi *asymptotic*
- Notasi *asymptotic* dapat dituliskan dengan beberpa simbul berikut **Θ , *O*, *o*, Ω, ω**
- Mendefinisikan himpunan fungsi ;
- Pada prakteknya untuk membandingan 2 ukuran fungsi.
- Notasi menggambarkan perbedaan *rate-of-growth* hubungan antara definisi fungsi dan definisi himpunan fungsi.

# LANJUTAN

- **Terdapat tiga macam yaitu :**

- Keadaan terbaik (best case) : Dilambangkan dengan notasi **q(…)** *dibaca Theta*
- Keadaan rata-rata (average case) : Dilambangkan dengan notasi **W(…)** *dibaca Omega*
- Keadaan terburuk (worst case) : Dilambangkan dengan notasi **O(…)** *dibaca Big-O*

engan menggunakan patokan keadaan terburuk (worst case) yang dinyatakan dengan Big-O

# LANJUTAN

**Θ- Notation (Notasi Big – Theta)**

Merupakan notasi asymptotic untuk batas atas dan bawah.

untuk fungsi g *(n)*, Θ(g *(n)*) adalah himpunan fungsi f *(n)* didefinisikan sebagai berikut:

Θ (g *(n)*) : {f *(n)*: adalah nilai konstanta positif $c_1$, $c_2$ dan $n_0$ sehingga **$0 \leq c_1 g$ *(n)* f *(n)* $\leq c_2 g$ *(n)*** untuk semua $n \geq n_0$}

Untuk menunjukkan bahwa f *(n)* adalah anggota dari Θ(g *(n)*)

***g (n)* adalah *asymptotically tight bound* untuk *f (n).*** Atau dapat dikatakan bahwa Θ (big-theta) adalah *tight-bound* dari suatu fungsi

**Contoh :**

1. $\frac{1}{2} n^2 - 3n = \Theta(n^2)$

Jawab : $0 \leq c_1 g\ (n)\ f\ (n) \leq c_2 g\ (n)$

$c_1 n^2 \leq \frac{1}{2} n^2 - 3n \leq c_2 n^2$

$c_1 \leq \quad - \quad \leq c_2$

$c_1 =$

$c_2 =$

$n = 7$

# LANJUTAN

**O-Notation (Big-Oh Notation)**

Asimtotik notasi Big-Theta (Θ) merupakan batas fungsi dari atas dan bawah. Ketika kita hanya memiliki asimtotik batas atas, kita menggunakan notasi O (Big-Oh)

Untuk fungsi *g (n)*,kita definisikan O(*g (n)*) sbg *big-Oh* dari *n*, sbg himpunan:

O (g *(n)*) = {f *(n)*: konstanta c dan $n_0$ positif sedemikian sehingga $0 \leq f(n) \leq c\, g(n)$ untuk semua n> = n0}

f *(n)* = Θ(g *(n)*) mengisyaratkan bahwa f *(n)* = O (g *(n)*) selama Θ-notation adalah lebih kuat daripada O-notation

Contoh :

$n^2 - 3n = O(n^2)$

$0 \leq f(n) \leq c\, g(n)$

$0 \leq n^2 - 3n \leq c\, n^2$

$0 \leq \; - \; \leq c$

c =

n = 7

# LANJUTAN

**Ω – Notation**

Ω – Notation memberikan batas bawah asimtotik

Ω (g *(n)*) = {f *(n)*: terdapat konstanta c dan $n_0$ positif sedemikian sehingga $0 \leq c\, g\,(n) \leq f\,(n)$ untuk semua $n \geq n0$}

Contoh

1. $7n^2 = \Omega\,(n)$

Jawab

$0 \leq c\, g\,(n) \leq f\,(n)$

$0 \leq c\, n \leq 7n^2$

$0 \leq c \leq 7n$

$c = 7$

$n = 1$

Theorema :

Untuk setiap fungsi f *(n)* dan g *(n)*, kita memiliki

f *(n)* = Θ (g *(n)*) jika dan hanya jika

f *(n)* = O (g *(n)*) dan

f *(n)* = Ω (g *(n)*)

Kami biasanya menggunakan teorema ini untuk membuktikan batas asimtotik ketat daribatas asimtotik atas dan bawah

# LANJUTAN

**o-notation (little-oh notation)**

Asymptotic upper bound disediakan oleh O-notation mungkin bukan merupakan *tight-asymptotic*
$2n^2 = O(n^2)$ adalah tight asymptotic, namun

$2n = O(n^2)$ *tidak !*

o-notation untuk menunjukkan sebuah batas atas yang tidak secara ketat terikat (***not asymptotically tight***)

$o(g(n)) = \{f(n)$: untuk setiap konstanta positif $c \geq 0$, terdapat $n_0$ konstan $> 0$ sedemikian sehingga $0 \leq f(n) < cg(n)$ untuk semua $n \geq n0\}$

**ω -notation**

Kita menggunakan **ω -notation** untuk menunjukkan batas bawah yang *not asymptotically tight*
$f(n) = \omega(g(n))$ jika dan hanya jika $g(n) = O(f(n))$

$\omega(g(n)) = \{f(n)$: untuk setiap konstanta positif $c > 0$, terdapat $n_0$ konstan$> 0$ sedemikian sehingga $0 \leq c\,g(n) < f(n)$ untuk semua $n \geq n_0\}$

Contoh : n2/2= w *(n)*, but n2/21 w(n2)

# LANJUTAN

MEMBACA BIG-OH

- O(1) artinya algoritma konstan
- O(n) artinya algoritma linear
- $O(n^2)$ artinya algorritma quadratic
- $O(n^3)$ artinya algoritma qubic
- O(log n) contohnya pada full balanced Binary Search Tree
- $O(n^m)$ artinya algoritma eksponensial
- Notasi Big-O bisa berisi kombinasi dari contoh di atas
- Penyederhanaan Big-O dilakukan pada komponen yang “less important”

**Contoh 3.** Algoritma *sequential search*.

```
procedure PencarianBeruntun(input a1, a2, ..., an : integer, x : integer,
                            output idx : integer)
Deklarasi
  k : integer
  ketemu : boolean    { bernilai true jika x ditemukan atau false jika x
tidak ditemukan }

Algoritma:
  k←1
  ketemu ← false
  while (k ≤ n) and (not ketemu) do
    if ak = x then
      ketemu←true
    else
      k ← k + 1
    endif
  endwhile
  { k > n or ketemu }

  if ketemu then    { x ditemukan }
     idx←k
  else
     idx← 0        { x tidak ditemukan }
  endif
```

# SIFAT-SIFAT ALGORITMA

- 1. Banyaknya langkah instruksi harus berhingga

  Pelaksanaan sebuah algoritma yang terprogram haruslah dapat diakhiri atau diselesaikan melalui sejumlah langkah operasional yang berhingga.

- Jika tidak demikian, kita tidak akan dapat mengharapkan bahwa pelaksaan algoritma tersebut dapat menghasilkan suatu solusi yang baik.

# LANJUTAN

- 2. Langkah atau instruksi harus jelas

  Artinya bahwa penulisa setiap langkah yang terdapat didalam sebuah algoritma harus memiliki arti yang khusus atau spesifik sehingga dapat dibedakan antara penulisan langkah untuk komputer(program/pemrograman) dengan penulisan langkah bagi manusia(pesudocode).

- Manusia akan lebih mudah memahami algoritma yang terdiri atas simbol-simbol(Contoh: pembuatan algoritma dengan diagram alur/flowchart) sedangkan komputer hanya membutuhkan sebuah penulisan algoritma dengan kode-kode yang dituangkan dalam bahasa yang dimengerti oleh komputer itu sendiri(bahasa pemrograman).

# LANJUTAN

- 3. Proses harus jelas dan mempunyai batasan

  Rangkaian suatu proses yang berisi langkah-langkah instruksi dari suatu algoritma yang akan dilaksanakn harus ditetapkan dengna jelas, baik dan pasti sebab sebuah algoritma harus memiliki instruksi dasar tertentu dimana setiap instruksi harus memiliki unsur pelaksana yang berfungsi sebagai pemroses data yang akan dimasukkan dalam sebuah komputer.
- Dengan demikian, sebuah algoritma harus ditulis dengan jelas tentang batasa-batasan proses yang akan dilaksanakan oleh komputer.

# LANJUTAN

- 4. Input dan Output harus mempunyai batasan

  Input merupakan data yang dimasukkan ke dalam algoritma yang untuk kemudian akan dilaksanakan oleh komputer.
- Dengan begitu, input yang diberikan harus sesuai dengan jenis dari bahasa pemrograman yang digunakan, sedangkan ouput merupakan hasil yang diperoleh dari pekerjaan yang dilaksanakan komputer untuk kepentingan user yang merupakan pihak diluar komputer.
- Algoritma harus menghasilkan output karena merupaka solusi yang diharapkan dari suatu masalah yang timbul.

# LANJUTAN

- 5. Efektifitas

  Instruksi yang diberikan pada komputer agar hanya menjalankan atau melaksanakan proses yang mampu dilaksanakannya.

- Yang dimaksud mampu adalah bahwa suatu algoritma atau instruksi-instruksi dalam sebuah program hanya akan dapat dilaksanakan jika informasi yang diberikan oleh instruksi-instruksi tersebut lengkap, benar dan jelas.

# LANJUTAN

- 6. Adanya batasan ruang lingkup

  Sebuah algoritma yang baik adalah hanya ditujukan bagi suatu masalah tertentu saja.
- Susunana input harus ditentukan lebih dulu sebab susunan tersebut enentukan sifat umum dari algoritma yang bersangkutan.

# TERIMA KASIH