Running Time dan Notasi Asimtotik

Wijayanti Nurul Khotimah, M.Sc.

Tujuan Perkuliahan

- ✓ Mahasiswa mampu menghitung running time dari suatu algoritma
- ✓ Mahasiswa mampu menampilkan running time dalam notasi asimtotik
- ✓ Mahasiswa mampu membandingkan keefektifan dua buah algoritma

Agenda Perkuliahan

- ✓ Perhitungan Running Time
- ✓ Notasi Asimtotic

Latar Belakang

- ✓ **Running Time**: **jumlah waktu** yang digunakan untuk mengeksekusi seluruh operasi di dalam suatu algoritma.
- ✓ Running Time biasanya dinyatakan dengan ***T(n)***, dimana ***n*** adalah ukuran input

Contoh Menghitung *Running Time*

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] = key$	c_8	$n-1$

Cara Menghitung *Running Time (2)*

- ✓ Selanjutnya running time diperoleh dengan menjumlahkan seluruh waktu yang dibutuhkan untuk eksekusi algoritma

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1)$$
$$+ c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) \; .$$

- ✓ Dari contoh di atas tampak bahwa t_j adalah suatu variabel yang nilainya tergantung pada isi dari input.
- ✓ Jika input sudah terurut, maka t_j pada baris ke-5 akan bernilai 1 untuk *j=2,3,..,n*.

Cara Menghitung Running Time (3)

- ✓ Kondisi inilah yang disebut ***best case*** dengan running time:

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)\,. \end{aligned}$$

- ✓ Running time tersebut dapat diekspresikan dengan *an+b* (**fungsi linear**)

Cara Menghitung Running Time (4)

- ✓ Jika isi input dalam kondisi urutan yang terbalik, maka pada baris no 5 kita harus membandingkan seluruh isi dari subarray A[1,...,j-1] sehingga $t_j=j$ untuk j=2,3,4,...,n
- ✓ Sehingga running timenya adalah:

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\
&\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\
&= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n \\
&\quad - (c_2 + c_4 + c_5 + c_8)\,.
\end{aligned}
$$

Cara Menghitung Running Time (5)

- ✓ Kondisi ini disebut kondisi ***worst case*** dan running timenya dapat diekspresikan dengan an^2+bn+c (***quadratic function***)

Worst case VS Best Case

- Umumnya perhitungan running time difokuskan pada kondisi *worst case* karena beberapa alasan berikut:

1. Kondisi *worst case* merupakan batas atas dari running time, artinya algoritma tidak akan berjalan lebih lambat dari ini.
2. Untuk beberapa kasus, kondisi *worst case* sering terjadi. Contoh: pencarian informasi dalam suatu database, kondisi informasi tidak ada lebih sering terjadi dari pada kondisi informasi yang dicari berada pada data pertama.

Pertumbuhan Fungsi

- ✓ Pada perhitungan sebelumnya, *cost* dari masing-masing operasi (c_i) diabaikan sehingga selanjutnya running time dinyatakan dengan an^2+bn+c (dimana a,b, dan c adalah *abstract cost*).
- ✓ Selanjutnya running time dinyatakan dengan abstraksi yang lebih sederhana yang disebut dengan ***rate of growth*** atau ***order of growth*** atau **pertumbuhan fungsi**.
- ✓ Dalam kasus di atas, *order of growth*-nya adalah an^2 (diambil dari pangkat terbesar) dan selanjutnya bisa dinyatakan dengan notasi asimtotik $\Theta(n^2)$

Latar Belakang

- ✓ Waktu yang dibutuhkan sebuah algoritma meningkat seiring dengan meningkatnya ukuran input.
  - Ukuran input → tanpa batas
  - Waktu → terbatas
- ✓ Waktu tersebutlah yang menjadi karakteristik dari efisiensi suatu algoritma dan yang selanjutnya digunakan untuk membandingkan performa relatif suatu algoritma.
- ✓ Selanjutnya, efisiensi algoritma dinyatakan dengan fungsi *n* yang disebut dengan **notasi asimtotik**

Notasi Asimtotik dan Kelas Efisiensi

- ✓ Merupakan suatu cara untuk membandingkan fungsi-fungsi dengan mengabaikan faktor konstanta dan ukuran input.
- ✓ Ada 3 macam notasi asimtotik untuk kelas efisiensi yaitu:

1. O($g(n)$) → read: big oh: class of functions $f(n)$ that grow no faster than $g(n)$
2. Ω($g(n)$) → read: big omega: class of functions $f(n)$ that grow at least as fast as $g(n)$
3. Θ($g(n)$) →read: big theta : class of functions $f(n)$ that grow at same rate as $g(n)$

O-notation

- ✓ Definisi: $f(n)$ berada dalam $O(g(n))$, dilambangkan dengan $f(n) \in O(g(n))$, jika *order of growth* dari $f(n) \leq$ order of growth dari $g(n)$, dimana terkadang f(n) dibatasi dengan konstanta c dan non-negative integer n_0 sehingga

  $f(n) \leq c\, g(n)$ untuk semua $n \geq n_0$

- ✓ Contoh
  - $100n+5$ berada dalam $O(n^2)$
- ✓ Bukti:

$$100n + 5 \leq 100n + n \text{ (for all } n \geq 5) = 101n \leq 101n^2.$$

Jadi didapatkan nilai c=101, dan n_0=5

Grafik notasi big-oh

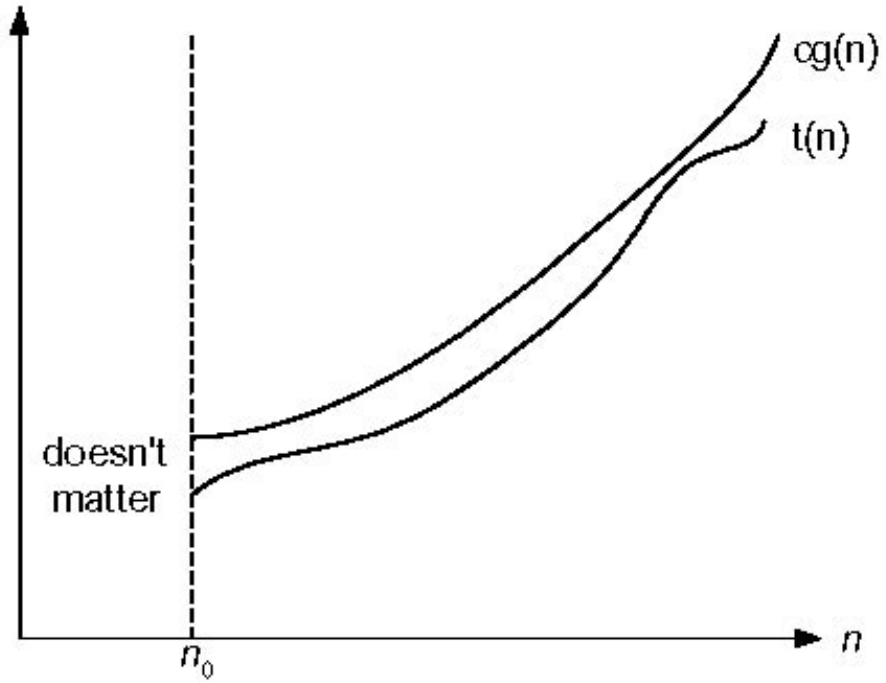

Figure 2.1 Big-oh notation: $t(n) \in O(g(n))$

Ω-notation

- Definisi
  - Sebuah fungsi *t(n)* berada dalam *Ω(g(n))*, dilambangkan dengan *t(n)* $\in$ *Ω(g(n))*, jika order of growth dari *t*(*n*) ≥ order of growth dari *g*(*n*) dimana *t(n)* dibatasi oleh beberapa konstanta dari *g(n)* seperti c dan n_0 sehingga

    $t(n) \geq cg(n)$ for all $n \geq n_0$

- Contoh:
  - $n^3 \in \Omega(n^2)$

$$n^3 \geq n^2 \quad \text{for all } n \geq 0,$$

Di dapat c=1, dan n_0=0

Grafik notasi big-omega

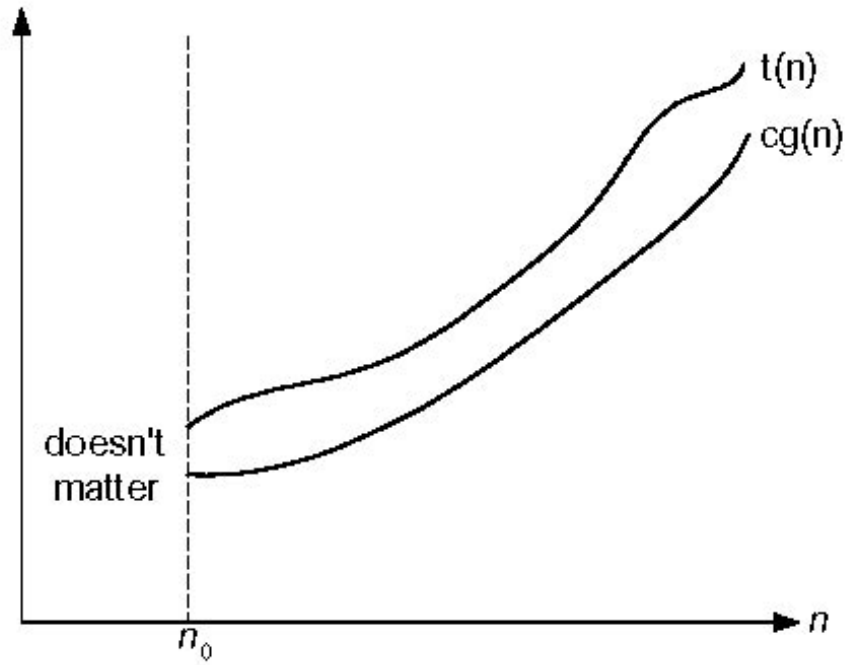

Fig. 2.2 Big-omega notation: $t(n) \in \Omega(g(n))$

Θ-notation

- Definisi
  - Sebuah fungsi *t(n)* berada dalam $\Theta(g(n))$, dilambangkan dengan $t(n) \in \Theta(g(n))$, jika order of growth *t(n)* berada diantara order of growth dari *g(n)* dengan beberapa batasan c_1 ,c_2 dan n_0 sehingga

    $c_2\, g(n) \leq t(n) \leq c_1\, g(n)$ untuk semua $n \geq n_0$

- Contoh
  - $(1/2)n(n-1) \in \Theta(n^2)$

    Batas atas: $\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2 \quad \text{for all } n \geq 0.$

    Batas bawah $\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \geq \frac{1}{2}n^2 - \frac{1}{2}n\frac{1}{2}n \text{ (for all } n \geq 2) = \frac{1}{4}n^2.$

    Sehingga: C2=1/4, C1=1/2, n_0=0

Grafik notasi big-THETA

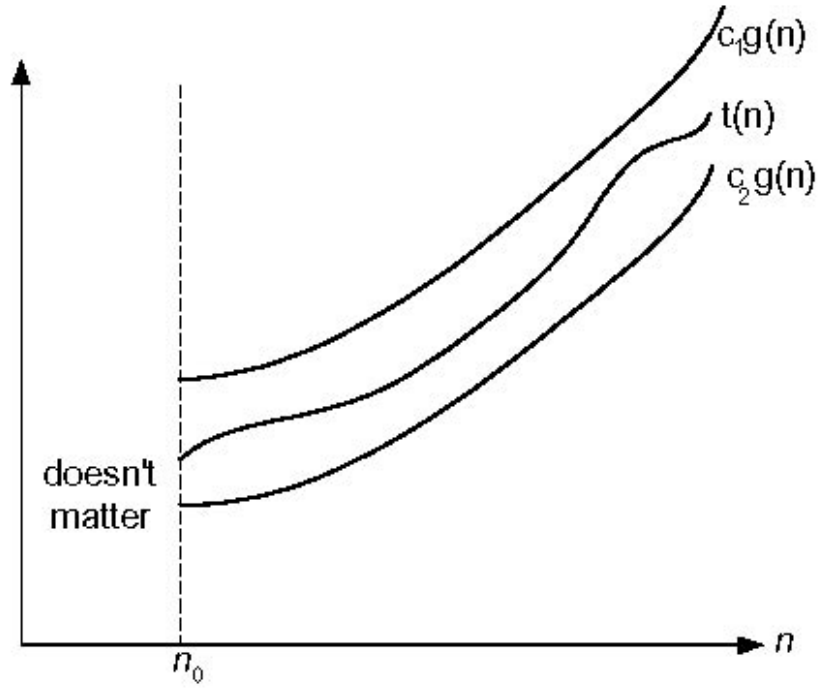

Figure 2.3 Big-theta notation: $t(n) \in \Theta(g(n))$

Teorema

- Jika $t_1(n) \in O(g_1(n))$ dan $t_2(n) \in O(g_2(n))$, maka $t_1(n) + t_2(n) \in O(max\{g_1(n), g_2(n)\})$.
  - Hal ini juga berlaku untuk big OMEGA dan big THETA
- Bukti: Buku Levitin hal 56
- Contoh:
  $(½)n(n-1) \in O(n^2)$
  $n-1 \in O(n)$
  Maka : $(½)n(n-1) + n-1 \in O(max\{n^2, n\}) = O(n^2)$

- $f(n) \in O(f(n))$
- $f(n) \in O(g(n))$ iff $g(n) \in \Omega(f(n))$
- If $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$, then $f(n) \in O(h(n))$
  Note similarity with $a \leq b$

- If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then
  $$f_1(n) + f_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

Also, $\Sigma_{1 \leq i \leq n} \Theta(f(i)) = \Theta(\Sigma_{1 \leq i \leq n} f(i))$

Menentukan Order of Growth Menggunakan Limit

$$\lim_{n \to \infty} T(n)/g(n) = \begin{cases} 0 & \text{order of growth of } T(n) < \text{order of growth of } g(n) \\ c > 0 & \text{order of growth of } T(n) = \text{order of growth of } g(n) \\ \infty & \text{order of growth of } T(n) > \text{order of growth of } g(n) \end{cases}$$

Contoh

- **$10n$ vs. n^2**
- **$n(n+1)/2$ vs. n^2**

L'Hôpital's rule dan Stirling's formula

L'Hôpital's rule: If $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$ and the derivatives f', g' exist, then

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

Latihan: log *n* vs. *n*

$$\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n\to\infty} \frac{(\log_2 n)'}{(\sqrt{n})'} = \lim_{n\to\infty} \frac{(\log_2 e)\,\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = 2\log_2 e \lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0.$$

Bagaimana hubungan notasi asimtotiknya?

Stirling's formula: $n! \approx (2\pi n)^{1/2} (n/e)^n$

$$\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n} = \lim_{n\to\infty} \sqrt{2\pi n}\frac{n^n}{2^n e^n} = \lim_{n\to\infty} \sqrt{2\pi n}\left(\frac{n}{2e}\right)^n = \infty.$$

Orders of growth dari beberapa fungsi penting

- Semua fungsi algoritmic $\log_a n$ berasal dari class yang sama $\Theta(\log n)$
- Semua polinomial dengan derajat yang sama k, berasal dari class yang sama

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_0 \in \Theta(n^k)$$

- Fungsi Exponential a^n mempunyai orders of growth yang berbeda untuk nilai a yang berbeda
- order $\log n$ < order n^α (α>0) < order a^n < order $n!$ < order n^n

Monotonicity

- ✓ Suatu fungsi f(n) dikatakan monotonically increasing jika $m \leq n$ menunjukkan $f(m) \leq f(n)$.
- ✓ Suatu fungsi f(n) dikatakan monotonically decreasing jika $m \leq n$ menunjukkan $f(m) \geq f(n)$.
- ✓ Suatu fungsi f(n) dikatakan strictly increasing jika $m<n$ menunjukkan $f(m) < f(n)$.
- ✓ Suatu fungsi f(n) dikatakan strictly decreasing jika $m<n$ menunjukkan $f(m) > f(n)$.

Floors and ceilings

- ✓ Untuk semua bilang real x, kita menyatakan bilangan integer terbesar yang kurang dari atau sama dengan x dengan ⌊x⌋ (dibaca: the floor of x).
- ✓ Bilangan integer terkecil yang lebih dari atau sama dengan x dinyatakan dengan ⌈x⌉ (the ceiling of x)

Logarithms

$$
\begin{aligned}
\lg n &= \log_2 n && \text{(binary logarithm)} , \\
\ln n &= \log_e n && \text{(natural logarithm)} , \\
\lg^k n &= (\lg n)^k && \text{(exponentiation)} , \\
\lg \lg n &= \lg(\lg n) && \text{(composition)} .
\end{aligned}
$$

Fungsi Umum Lainnya

1	**constant**
log *n*	**logarithmic**
n	**linear**
n* log *n	***n*-log-*n***
n^2	**quadratic**
n^3	**cubic**
2^n	**exponential**
***n*!**	**factorial**

LATIHAN

2. Use the informal definitions of O, Θ, and Ω to determine whether the following assertions are true or false.

a. $n(n+1)/2 \in O(n^3)$ **b.** $n(n+1)/2 \in O(n^2)$

c. $n(n+1)/2 \in \Theta(n^3)$ **d.** $n(n+1)/2 \in \Omega(n)$

5. List the following functions according to their order of growth from the lowest to the highest:

$$(n-2)!,\ \ 5\lg(n+100)^{10},\ \ 2^{2n},\ \ 0.001n^4+3n^3+1,\ \ \ln^2 n,\ \ \sqrt[3]{n},\ \ 3^n.$$