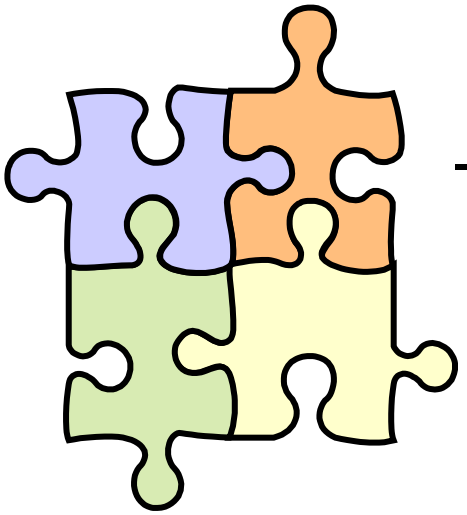
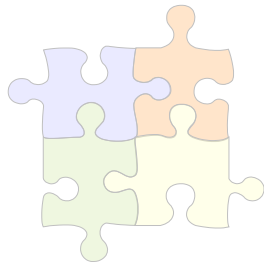
Notasi Asimtot untuk Analisis Efisiensi Waktu



Analisis Algoritma

14/09/2006

Isi



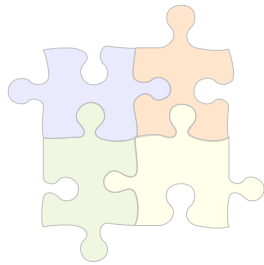
- Notasi Asimtot:
  - O (big oh)
  - Ω (big omega)
  - Θ (big theta)
- Kelas-kelas Efisiensi Dasar

Pada bahasan berikut…

- t(n) & g(n): setiap fungsi non negatif didefinisikan pada sekumpulan angka alami

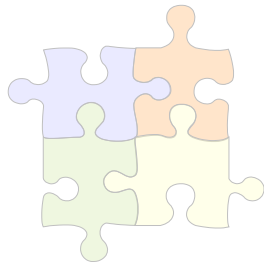
- t(n) → algoritma running time
  - Biasanya didindikasikan oleh perhitungan operasi dasarnya C(n)
- g(n) → beberapa fungsi sederhana untuk membandingkan perhitungan dengan

$O(g(n))$: Informally



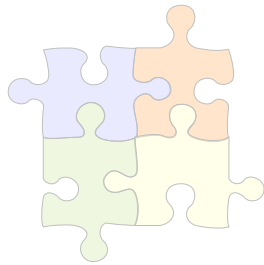
- **$O(g(n))$** adalah kumpulan semua fungsi dengan tingkat pertumbuhan **lebih** kecil atau **sama** dengan g(n)
- Contoh:
  - $n \in O(n^2)$; $100n + 5 \in O(n^2)$
  - $\frac{1}{2}\, n\,(n-1) \in O(n^2)$
  - $n^3 \notin O(n^2)$; $0.0001\, n^3 \notin O(n^2)$; $n^4+n+1 \notin O(n^2)$

$\Omega(g(n))$: Informally



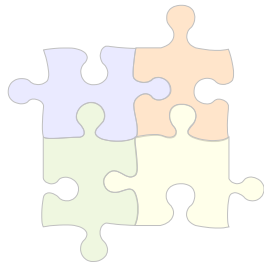
- **$\Omega(g(n))$** kumpulan semua fungsi dengan tingkat pertumbuhan **lebih besar** atau **sama** dengan g(n)
- Examples:
  - $n^3 \in \Omega(n^2)$
  - $\frac{1}{2}\, n\, (n-1) \in \Omega(n^2)$
  - $100n + 5 \notin \Omega(n^2)$

$\Theta(g(n))$: Informally



- **$\Theta(g(n))$** adalah kumpulan semua fungsi dengan tingkat pertumbuhan **sama** dengan g(n)
- Examples:
  - an^2+bn+c; $a>0 \in \Theta(n^2)$; $n^2+\sin n \in \Theta(n^2)$
  - $\frac{1}{2}\, n\,(n-1) \in \Theta(n^2)$; $n^2+\log n \in \Theta(n^2)$
  - $100n + 5 \notin \Theta(n^2)$; $n^3 \notin \Theta(n^2)$

Notasi-*O*: Formally

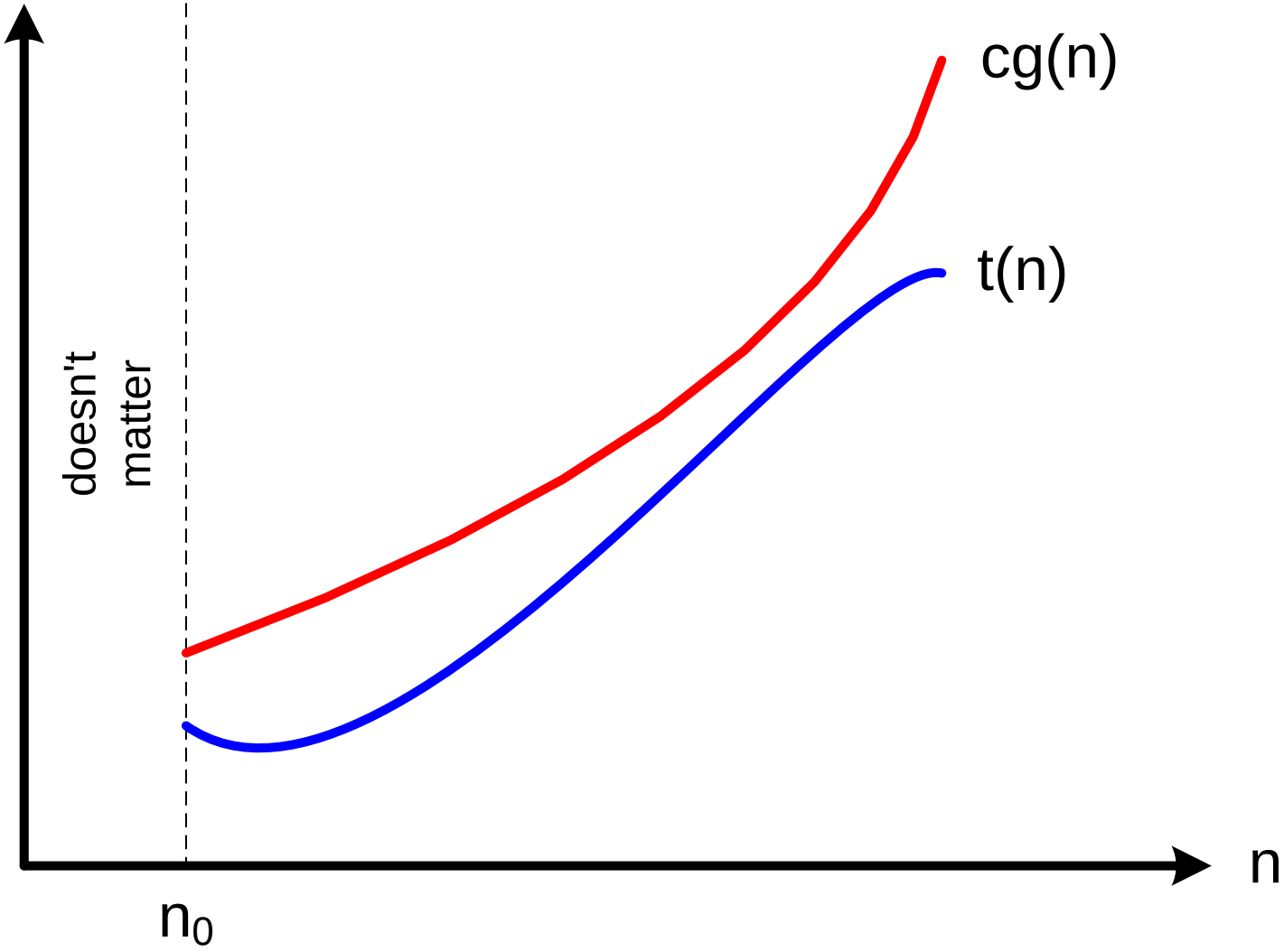


- **DEF1**: Sebuah fungsi t(n) dikatakan himpunan bagian dari *O*(g(n)),ditunjukkan oleh t(n) $\in$ *O*(g(n)), jika t(n) di batas atas beberapa pengali kontstan dari g(n) untuk semua ukuran n

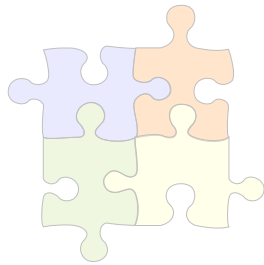
- Contoh: terdapat beberapa konstanta positif c dan beberapa integer nonnegatif n_0, yaitu:

$$t(n) \leq cg(n) \text{ untuk semua } n \geq n_0$$

Ilustrasi :t(n) ∈ *O*(g(n))



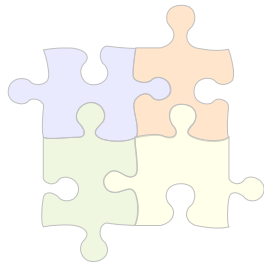
Contoh: $100n + 5 \in O(n^2)$



- DEF1: cari c and n_0, sehingga $t(n) \leq cg(n)$ untuk semua $n \geq n_0$

- $100n + 5 \leq 100n + n$ (for all $n \geq 5$) $= 101n \leq 101n^2 \rightarrow c=101, n_0=5$
- $100n + 5 \leq 100n + 5n$ (for all $n \geq 1$) $= 105n \leq 105n^2 \rightarrow c=105, n_0=1$
- …

Ω-notation: Formally

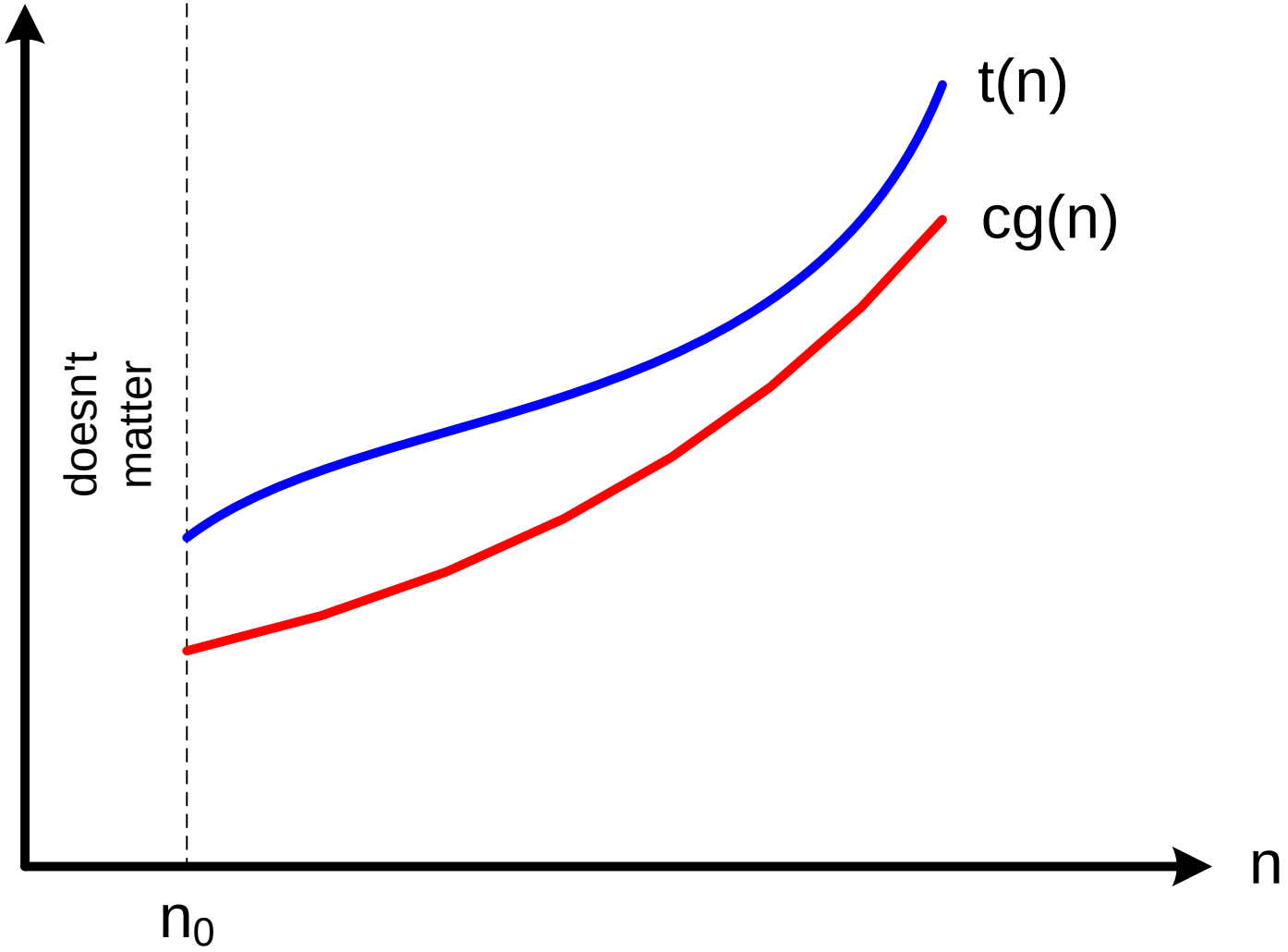


- **DEF2**: Suatu fungsi t(n) dikatakan himpunan bagian dari $\Omega(g(n))$, ditunjukkan dengan $t(n) \in \Omega(g(n))$, jika t(n) dibatasi dibawah beberapa pengali konstan dari g(n) for semua ukuran n

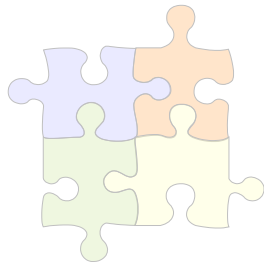
- Contoh: terdapat beberapa konstanta positif c dan beberapa integer nonnegatif n_0, yaitu:

$$t(n) \geq cg(n) \text{ for all } n \geq n_0$$

Ilustrasi : $t(n) \in \Omega(g(n))$



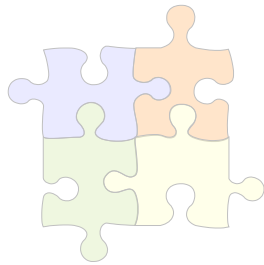
Contoh: $n^3 \in \Omega(n^2)$



- DEF2: temukan c dan n_0, sehingga t(n) **≥** cg(n) untuk semua $n \geq n_0$

- $n^3 \geq n^2$ (for all $n \geq 0$) → c=1, n_0=0
- …

Θ-notation: Formally

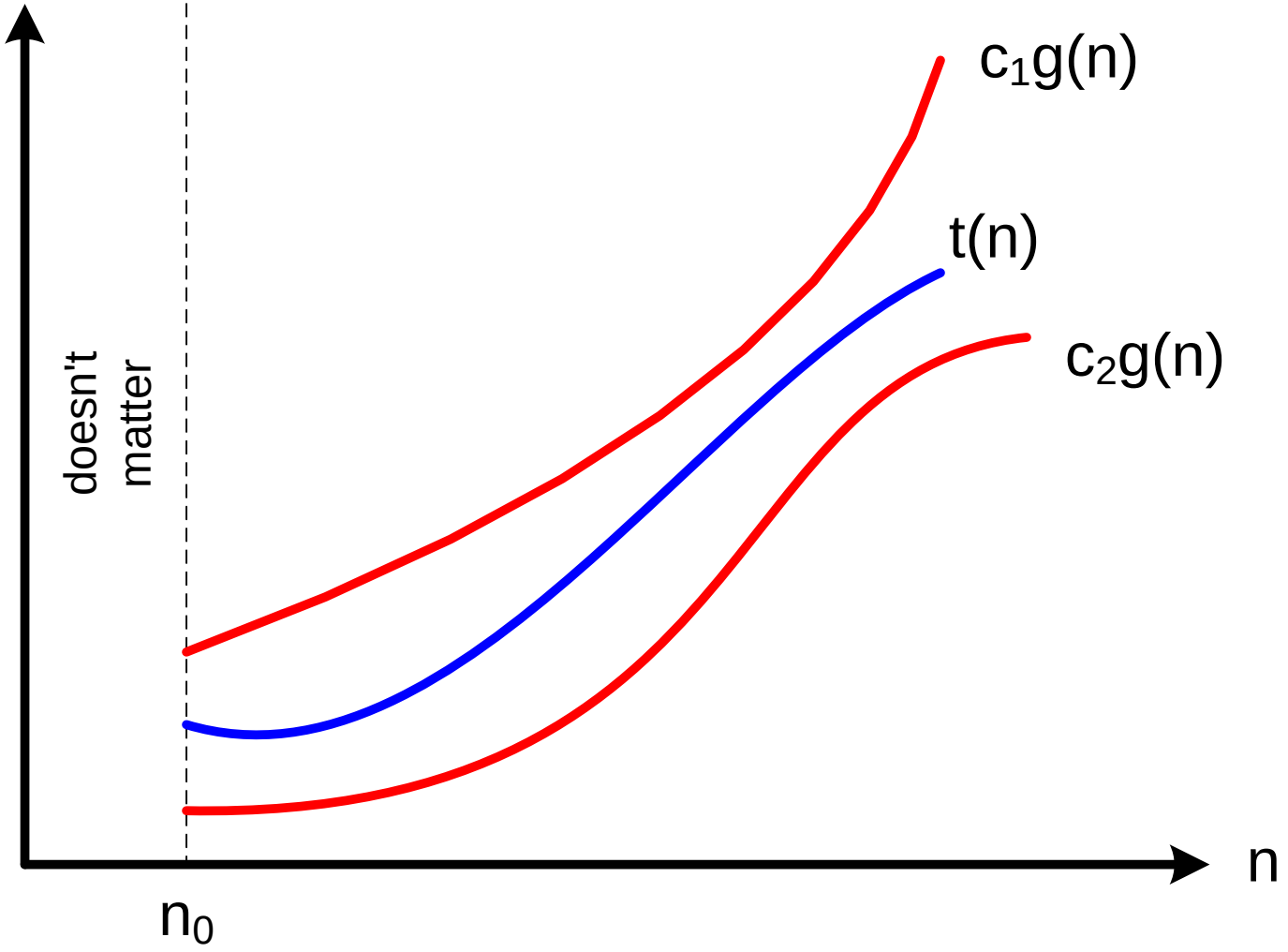


- **DEF3**: Fungsi t(n) dikatakan himpunan bagaian Θ(g(n)), ditunjukkan oleh $t(n) \in \Theta(g(n))$, jika t(n) dibatasi **diatas** dan **dibawah** beberapa pengali tetap dari g(n) for untuk semua ukuran n

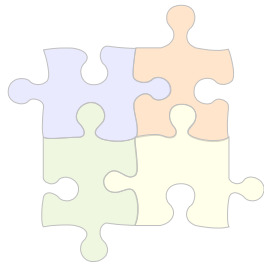
- Contoh: terdapat beberapa konstanta positif c dan beberapa integer nonnegatif n_0, yaitu:

$$c_2 g(n) \leq t(n) \leq c_1 g(n) \text{ for all } n \geq n_0$$

$t(n) \in \Theta(g(n))$: Illustration



Contoh: ½n(n-1) ∈ Θ(n^2)



- DEF3: temukan c_1 dan c_2 dan beberapa integer nonnegatifn_0, sehingga

  $c_2 g(n) \leq t(n) \leq c_1 g(n)$ for all $n \geq n_0$

- Batas atas ½ n(n-1) =

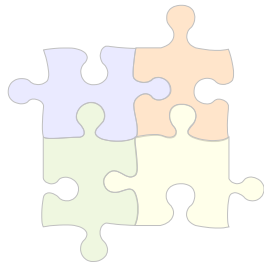
  ½ n^2 – ½ n ≤ ½ n^2 (for all n ≥ 0)

- Batas bawah : ½ n(n-1) =

  ½ n^2 – ½ n ≥ ½ n^2 - ½ n ½ n (for all n ≥ 2) = ¼ n^2

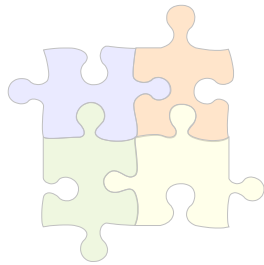
- c_1 = ½, c_2 = ¼, n_0 = 2

Kelas-kelas Efisiensi Dasar



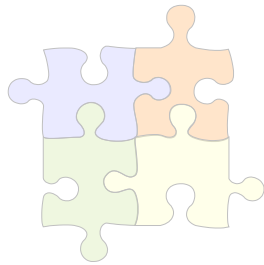
- Efisiensi waktu sejumlah besar algoritma terbagi ke dalam beberapa kelas
  - 1, log n, n, n log n, n^2, n^3, 2^n, n!
- Konstanta pengali diabaikan →algoritma pada efisiensi yang lebih buruk **mungkin saja** memproses lebih cepat dibanding algoritma pada efisensi yang lebih baik
  - Contoh Alg A: n^3, alg B: 10^6n^2; kecuali $n > 10^6$,alg B memproses lebih cepat daripada alg A

Class: 1



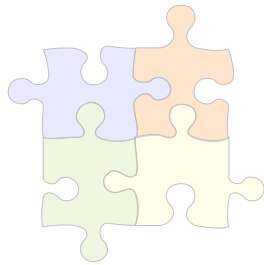
- Nama: *constant*
- Komentar:
  - Efisiensi pada kasus terbaik
  - Hanya sedikit contoh yang ada → running time algoritma biasanya akan menjadi tak terbatas ketika ukuran inputnya meningkat tak terbatas

Class: log n



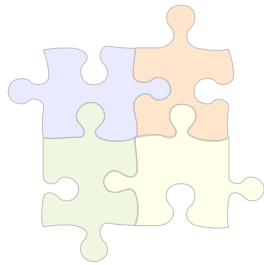
- Nama: *logarithmic*
- Komentar:
  - Biasanya merupakan hasil pemotongan ukuran problem dengan faktor konstan pada tiap iterasi algoritma
  - Algoritma logaritmik tidak dapat menerima semua input atau bahkan pembagian tetap dari input tersebut: sembarang algoritma yang melakukannya setidaknya akan memiliki linear running time

Class: n



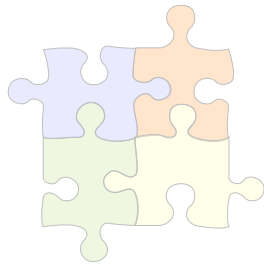
- Nama: *linear*
- Komentar:
  - Algoritma yang memeriksa sebuah daftar dengan ukuran n (contoh: pencarian sekuensial) termasuk dalam class ini

Class: n log n



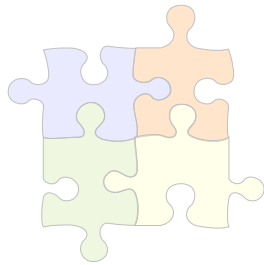
- Nama: *n-log-n*
- Komentar:
  - Banyak algoritmadivide-and-conquer, termasuk merge sort and quick sort pada kasus average, masuk dalam class ini

Class: n^2



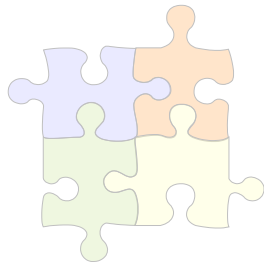
- Nama: *quadratic*
- Komentar:
  - Khususnya, algoritma yang mengkarakterisasi efisiensi menggunakan dua looping yang di-embed
  - Contoh standar: algoritma sorting dasar dan operasi tertentu pada matrik n-by-n

Class: n^3



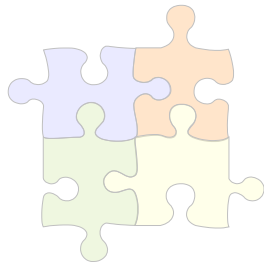
- Nama: *cubic*
- Komentar:
  - Biasanya, algoritma yang mengkarakterisasi efisiensi menggunakan dua looping yang di-embed
  - Beberapt algoritma nontrivial dari aljabar linear masuk ke dalam class ini

Class: 2^n



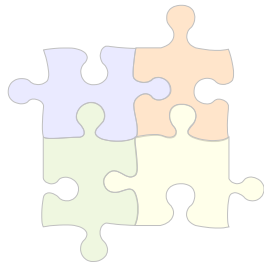
- Nama: *exponential*
- Komentar:
  - Digunakan khusus untuk algoritma yang men-generate seluruh subset suatu set dengan n-element
  - Istilah ‘exponential’ seting digunakan untuk hal ini dan bahkan untuk algoritma dengna tingkat pertumbuhan yang lebih cepat

Class: n!



- Nama: *factorial*
- Komentar:
  - Khusus untuk algoritma yang dipakai untuk men-generate semua permutasi suatu set dengan n-element

Properti yang berguna

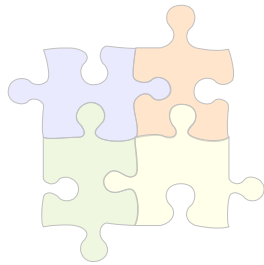


- **Teorema**:

  Jika $t_1(n) \in O(g_1(n))$ dan $t_2(n) \in O(g_2(n))$,
  maka $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

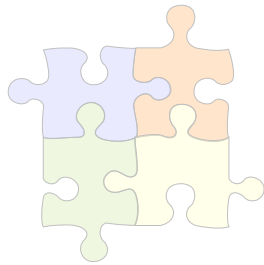
- Teorima tersebut berlaku juga pada notasi Ω and Θ

Contoh



- Alg yang dipakai untuk memeriksa apakah dalam array terdapat elemen yang sama:
  1. Sort the array
  2. Scan array yang telah di sort untuk memeriksa kesamaan elemen berikutnya
- (1) = ≤ ½n(n-1) comparison → $O(n^2)$
- (2) = ≤ n-1 comparison → $O(n)$
- The efficiency of (1)+(2) = $O(\max\{n^2,n\})$ = $O(n^2)$

Using Limits for Comparing OoG

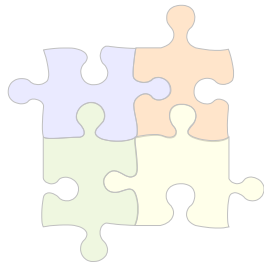


- A 'convenient' method for comparing order of growth of two specific functions
- Three principal cases:

$$\lim_{n\to\infty}\frac{t(n)}{g(n)}\begin{cases}0 & \text{implies that t(n) has a smaller OoG than g(n)}\\ c & \text{implies that t(n) has the same OoG as g(n)}\\ \infty & \text{implies that t(n) has a larger OoG than g(n)}\end{cases}$$

- The first two cases → t(n) $\in O$(g(n)); the last two cases → t(n) $\in \Omega$(g(n)); the second case alone → t(n) $\in \Theta$(g(n))

Limit-based: why convenient?



- It can take advantage of the powerful calculus techniques developed for computing limits, such as
  - L'Hopital's rule

$$\lim_{n\to\infty}\frac{t(n)}{g(n)}=\lim_{n\to\infty}\frac{t'(n)}{g'(n)}$$

  - Stirling's formula

$$n!\approx\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \text{ for large value of n}$$

Example (1)

- Compare OoG of ½n(n-1) and n^2.

$$\lim_{n\to\infty}\frac{\frac{1}{2}n(n-1)}{n^2}=\frac{1}{2}\lim_{n\to\infty}\frac{n^2-n}{n^2}=\frac{1}{2}\lim_{n\to\infty}\left(1-\frac{1}{n}\right)=\frac{1}{2}$$

  - The limit = c → ½n(n-1) $\in \Theta(n^2)$

- Compare OoG of $\log_2 n$ and $\sqrt{n}$

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

  - The limit = 0 → $\log_2 n$ has smaller order of $\sqrt{n}$

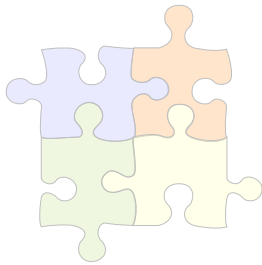
Example (2)

- Compare OoG of n! and 2^n.

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

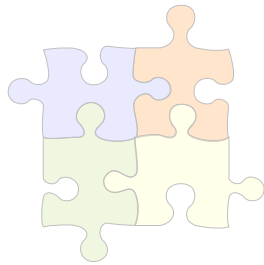
  - The limit = ∞ → n! $\in \Omega(2^n)$

Exercises (1)



1. True or false:
   - $n(n+1)/2 \in O(n^3)$
   - $n(n+1)/2 \in O(n^2)$
   - $n(n+1)/2 \in \Theta(n^3)$
   - $n(n+1)/2 \in \Omega(n)$
2. Indicate the class $\Theta(g(n))$:
   - $(n^2+1)^{10}$
   - $(10n^2+7n+3)^{\frac{1}{2}}$
   - $2n \log (n+2)^2+(n+2)^2 \log (n/2)$

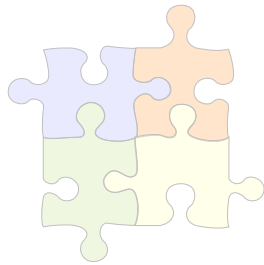
Exercises (2)



3. Prove that every polynomial
   $p(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_0$ with $a_k > 0$ belongs to $\Theta(n^k)$

4. Prove that exponential functions a^n have different orders of growth for different values of base $a > 0$

Exercises (3)



5. You are facing a wall that stretches infinitely in both directions. There is a door in the wall, but you know neither how far away nor in which direction. You can see the door only when you are right next to it. Design an algorithm that enables you to reach the door by walking at most $O(n)$ steps where n is the (unknown to you) number of steps between your initial position and the door.