

Kompleksitas Waktu dan Ruang

Sukmawati Nur Endah
Departemen Informatika UNDIP

Kemangkusan Algoritma (Efisiensi Algoritma)

- ▶ Algoritma yang bagus adalah algoritma yang mangkus
- ▶ Kemangkusan algoritma diukur dari :
 - ▶ Berapa jumlah waktu dan ruang (space) memori yang dibutuhkan untuk menjalankannya
- ▶ Algoritma yang mangkus :
 - ▶ Algoritma yang meminimumkan kebutuhan waktu dan ruang
- ▶ Kebutuhan waktu dan ruang tergantung pada :
 - ▶ Ukuran masukan (jumlah data) yang diproses
 - ▶ Biasa disimbolkan dengan n
 - ▶ Misal : Mengurutkan 1000 buah elemen $\rightarrow n = 1000$
Menghitung $6!$ $\rightarrow n = 6$

Mengapa kita memerlukan
Algoritma yang mangkus



Ilustrasi

- ▶ Dipunyai :
 - ▶ Algoritma yang waktu eksekusinya dalam orde eksponensial (2^n), dengan n adalah jumlah masukan yang diproses
 - ▶ Sebuah komputer yang mampu menjalankan program dengan masukan berukuran n dalam waktu $10^{-4} \times 2^n$ detik
- ▶ Dengan algoritma dan komputer tersebut, maka:
 - ▶ $n=10$ dibutuhkan waktu eksekusi 0,1024 detik ~ 1/10 detik
 - ▶ $n=20$ dibutuhkan waktu eksekusi 104,8576 detik ~ ...??... menit
 - ▶ Berapa waktu yang dibutuhkan untuk $n = 30$?
 - ▶ Jika dapat menjalankan komputer tanpa gangguan selama 1 tahun, berapa masukan yang dapat diselesaikan dalam 1 tahun tersebut?
 - ▶ Apa yang harus dilakukan agar mampu menyelesaikan jumlah masukan lebih banyak?

Ilustrasi

► Solusi 1

- Membeli komputer baru yang lebih cepat (10^{-6}) → dengan masukan n maka kecepatannya $10^{-6} \times 2^n$ detik
- Berapa masukan yang dapat dilakukan dengan komputer baru tersebut?

► Solusi 2

- Mengubah algoritmanya, misalkan menemukan algoritma baru untuk masalah sama dengan waktu penyelesaian n^3
- Sehingga dengan menggunakan komputer 1 dapat menyelesaikan masalah dalam waktu $10^{-4} \times n^3$ detik
- Satu hari dapat menyelesaikan lebih dari 900 masukan, berapa dalam satu tahun?
- Berapa masukan dapat diselesaikan dengan komputer 2 dalam satu tahun?

Kebutuhan Waktu dan Ruang

- ▶ Kebutuhan waktu dihitung dalam satuan detik
- ▶ Kebutuhan ruang dihitung dalam satuan byte atau kilobyte
- ▶ Bagaimana menghitung kebutuhan waktu?
 - ▶ Biasanya dengan menghitung durasi waktu yang dibutuhkan algoritma dalam menyelesaikan masalah tanpa menghitung kebutuhan waktu untuk menampilkan antar muka program, operasi I/O (baca/tulis)
- ▶ Ilustrasi : Menghitung rata-rata dari n buah data dengan prosedur HitungRerata
 - ▶ Asumsikan data masukan sudah dibaca
 - ▶ Jalankan program yang mengandung prosedur itu
 - ▶ Hitung selisih waktu sebelum pemanggilan prosedur dan sesudah selesai pemanggilan
 - ▶ Selisih tersebut merupakan kebutuhan waktu menghitung rerata n buah data
- ▶ Namun, model perhitungan seperti ini kurang dapat diterima, apa alasannya?

Kebutuhan Waktu dan Ruang

- ▶ Alasan 1:
 - ▶ Arsitektur komputer yang berbeda menghasilkan waktu yang berbeda pula untuk melaksanakan operasi-operasi dasar (penambahan, perkalian, pembagian, pengurangan)
 - ▶ Kebutuhan waktu jika menjalankan pada komputer IBM akan berbeda dengan saat dieksekusi pada komputer Macintosh
 - ▶ Komputer dengan arsitektur yang berbeda maka :
 - ▶ Akan berbeda pula perintah (*instruction*) dalam bahasa mesin yang dimilikinya
 - ▶ Akan berbeda pula kecepatan (*speed*) operasi piranti kerasnya
 - ▶ Perbedaan ini menghasilkan ukuran waktu (dan kebutuhan ruang memori) yang berbeda pula
 - ▶ Misal : komputer super cepat saat ini mampu melakukan operasi dasar dalam waktu 10^{-9} detik, tapi PC biasa memerlukan 10^{-6} detik

Kebutuhan Waktu dan Ruang

- ▶ Alasan 2:
 - ▶ Kebutuhan waktu sebuah algoritma bergantung pada compiler bahasa pemrograman yang digunakan
 - ▶ Compiler yang berbeda akan menerjemahkan program (dalam bahasa tingkat tinggi) ke dalam kode mesin (object code-bahasa tingkat rendah) yang berbeda pula
 - ▶ Kode mesin yang berbeda akan menggunakan ruang memori dan memerlukan waktu pelaksanaan program yang berbeda pula
- ▶ Model pengukuran waktu/ruang harus independen dari pertimbangan mesin dan compiler apapun
- ▶ Besaran yang dipakai → kompleksitas algoritma

Kompleksitas Algoritma

- ▶ Ada dua macam :
 - ▶ Kompleksitas waktu
 - ▶ Kompleksitas ruang
- ▶ Kompleksitas waktu diekspresikan sebagai jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan n
- ▶ Kompleksitas ruang diekspresikan sebagai jumlah memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari ukuran masukan n

Terminologi Kompleksitas Waktu/Ruang

- ▶ Ukuran besar masukan data untuk suatu algoritma : n
- ▶ Kompleksitas waktu : $T(n) \rightarrow$ jumlah operasi yang dilakukan
- ▶ Kompleksitas ruang : $S(n) \rightarrow$ jumlah memori yang dibutuhkan
- ▶ Pembahasan lebih lanjut berfokus pada kompleksitas waktu dengan pertimbangan :
 - ▶ Kompleksitas ruang terkait erat dengan bentuk struktur data
 - ▶ Ukuran memori sudah bukan lagi persoalan kritis

Kompleksitas Waktu

- ▶ Perhatikan Algoritma menghitung rata-rata berikut ini:

Procedure HitungRerata (input a_1, a_2, \dots, a_n : integer, output r : real)

Deklarasi

i : integer

jumlah : real

Algoritma

jumlah $\leftarrow 0$

$i \leftarrow 1$

while $i \leq n$ do

 jumlah \leftarrow jumlah + a_i

$i \leftarrow i+1$

endwhile

$r \leftarrow$ jumlah / n

Kompleksitas waktu

- ▶ Jenis operasi yang ada dalam Algoritma HitungRerata

- ▶ Operasi pengisian nilai

- $\text{jumlah} \leftarrow 0$ 1 kali

- $i \leftarrow 1$ 1 kali

- $\text{jumlah} \leftarrow \text{jumlah} + a1$ n kali

- $i \leftarrow i+1$ n kali

- $r \leftarrow \text{jumlah} / n$ 1 kali

- ▶ Jumlah seluruh operasi pengisian nilai : $3 + 2n$

Lanjutan

- ▶ Operasi penjumlahan

jumlah + a1 n kali

i+1 n kali

- ▶ Jumlah seluruh operasi penjumlahan : $2n$

- ▶ Operasi pembagian

jumlah / n 1 kali

- ▶ Jumlah seluruh operasi pembagian : 1

- ▶ Kompleksitas waktu : Total waktu yang diperlukan

$$3 + 2n + 2n + 1 = 4n + 4$$

Kompleksitas Waktu

- ▶ Idealnya untuk menghitung kompleksitas adalah seperti di atas
- ▶ Namun untuk alasan praktis : cukup dihitung jumlah operasi abstrak yang mendasari suatu algoritma
- ▶ Contoh :
 - ▶ Algoritma pencarian → operasi perbandingan x dengan elemen larik-lariknya
 - ▶ Algoritma pengurutan → operasi perbandingan elemen-elemen larik dan operasi pertukaran elemen-elemen
 - ▶ Algoritma perkalian dua matriks $A \times B$ masing-masing berukuran $n \times n$???

Kompleksitas Waktu

- ▶ Apa operasi dasar untuk Algoritma HitungRerata?
- ▶ Jika berdasarkan operasi dasarnya, berapa $T(n)$ Algoritma HitungRerata?

Latihan

- ▶ Buatlah algoritma untuk mencari elemen terbesar di dalam sebuah larik (array) yang berukuran n elemen
 - ▶ Apa operasi dasarnya?
 - ▶ Berapa kompleksitas waktu berdasarkan operasi dasarnya?

Tiga Jenis Kompleksitas Waktu

- ▶ $T_{\max}(n)$: Kompleksitas waktu untuk kasus terburuk (*worst case*)
 - ▶ Kebutuhan waktu maksimum yang diperlukan sebuah algoritma sebagai fungsi dari n
- ▶ $T_{\min}(n)$: Kompleksitas waktu untuk kasus terbaik (*best case*)
 - ▶ Kebutuhan waktu minimum yang diperlukan sebuah algoritma sebagai fungsi dari n
- ▶ $T_{\text{avg}}(n)$: Kompleksitas waktu untuk kasus rata-rata (*average case*)
 - ▶ Kebutuhan waktu rata-rata yang diperlukan sebuah algoritma sebagai fungsi dari n

PR

- ▶ Buatlah algoritma pencarian beruntun (sequensial search)
- ▶ Hitunglah kompleksitas waktu terbaik, terburuk dan rata-rata dari algoritma tersebut!