PROGRAM STUDI

S1 SISTEM KOMPUTER

UNIVERSITAS DIPONEGORO

Algoritma dan Pemrograman
Kompleksitas Algoritma

Oky Dwi Nurhayati, ST, MT
email: okydn@undip.ac.id

Masalah Analisis Algoritma

Kinerja yang perlu ditelaah pada algoritma:

- beban komputasi
- efisiensi penggunaan memori

Yang perlu diperhatikan:

- Kasus rata-rata : running time untuk tipikal data tertentu.
- Kasus terjelek : running time yang mungkin paling jelek pada konfigurasi masukan data tertentu.
- Program → bahasa yang dipakai
- Program sensitif terhadap input
- Program sulit dimengerti, dan secara matematis hasil tak tersedia/diketahui
- Sering program tak bisa dibandingkan

Kompleksitas Algoritma

- Sebuah algoritma tidak saja harus benar, tetapi juga harus mangkus (*efisien*).
- 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 ialah algoritma yang meminimumkan kebutuhan waktu dan ruang.
- Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan (n), yang menyatakan jumlah data yang diproses.
- Kemangkusan algoritma dapat digunakan untuk menilai algoritma yang terbaik.

Algoritma Rekursif

Bentuk rekursif :

a. suatu subrutin/fungsi/ prosedur yang memanggil dirinya sendiri.

b. Bentuk dimana pemanggilan subrutin terdapat dalam body subrutin

c. Dengan rekursi, program akan lebih mudah dilihat

Bentuk rekursi bertujuan untuk :

b. menyederhanakan penulisan program

c. menggantikan bentuk iterasi

Syarat bentuk rekursif:

- ada kondisi terminal (basis)
- ada *subroutine call yang melibatkan* parameter yang nilainya menuju kondisi terminal (*recurrence)*

Menghitung kompleksitas bentuk rekursif

Untuk bentuk rekursif, digunakan teknik perhitungan kompleksitas dengan relasi rekurens

Menghitung faktorial

Function Faktorial (input n : integer) → integer

{menghasilkan nilai n!, n tidak negatif}

Algoritma

- If n=0 then

  Return 1
- Else

  Return (n*faktorial (n-1))
- Endif

Menghitung faktorial

- Kompleksitas waktu :

b. untuk kasus basis, tidak ada operasi perkalian → (0)

c. untuk kasus rekurens, kompleksitas waktu diukur dari jumlah perkalian (1) ditambah kompleksitas waktu untuk faktorial (n-1)

Menghitung faktorial

Jadi relasi rekurens :

$$T(n) = \begin{cases} 0 & , n = 0 \\ T(n-1)+1 & , n > 0 \end{cases}$$

$$T(n) = 1 + T(n-1)$$

$$= 1 + 1 + T(n-2) = 2 + T(n-2)$$

$$= 2 + 1 + T(n-3) = 3 + T(n-3)$$

= .....

= n + $T(0)$

= n + 0

Jadi T(n) = n → O(n)

Relasi Rekurrens :

$$T(n) = \begin{cases} 1 & , n = 1 \\ 2T(n-1)+1 & , n > 1 \end{cases}$$

$$T(n) = 1 + 2T(n-1)$$

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

$$= 1 + 2 + 2^2(1 + 2T(n-3)) = 1 + 2 + 2^2 + 2^3 T(n-3)$$

$$= \ldots\ldots$$

$$= \left(1 + 2 + 2^2 + \ldots\ldots + 2^{n-2}\right) + 2^{n-1} T(1)$$

$$= 1 + 2 + 2^2 + \ldots.. + 2^{n-1}.1$$

$$= 2^n - 1$$

$$T(n) = 2^n - 1$$

$$T(n) \in O(2^n)$$

Persoalan Minimum & Maksimum

procedure MinMaks2(input A : TabelInt, i, j : integer,
output min, maks : integer)

{ Mencari nilai maksimum dan minimum di dalam tabel A yang berukuran n elemen secara Divide and Conquer.
Masukan: tabel A yang sudah terdefinisi elemenelemennya
Keluaran: nilai maksimum dan nilai minimum tabel}

Deklarasi
min1, min2, maks1, maks2 : integer

Persoalan Minimum &Maksimum

```
if i=j then { 1 elemen }
min←Ai
maks←Ai
else
if (i = j-1) then { 2 elemen }
if Ai < Aj then
maks←Aj
min←Ai
else
maks←Ai
min←Aj
endif
```

Persoalan Minimum &Maksimum

```
else { lebih dari 2 elemen }
k←(i+j) div 2 { bagidua tabel pada posisi k }
MinMaks2(A, i, k, min1, maks1)
MinMaks2(A, k+1, j, min2, maks2)
if min1 < min2 then
min←min1
else
min←min2
endif
if maks1<maks2 then
maks←maks2
else
maks←maks2
endif
```

Persoalan Minimum & Maksimum

Relasi Rekurens

$$T(n) = \begin{cases} 0 & ,n = 1 \\ 1 & ,n = 2 \\ 2T(n/2) + 2 & ,n > 2 \end{cases}$$

Penyelesaian:

Asumsi: $n = 2^k$, *dengan k bilangan bulat positif,* maka

$$T(n) = 2T(n/2) + 2$$

$$= 2(2T(n/4) + 2) + 2 = 4T(n/4) + 4 + 2$$

$$= 4\,(2T(n/8) + 2) + 4 + 2 = 8T(n/8) + 8 + 4 + 2$$

$$= \ldots$$

$$= 2^{k-1}\,T(2) + \sum_{i=1}^{k-1} 2^i$$

$$= 2^{k-1} \cdot 1 + 2^k - 2$$

Persoalan Minimum

$$= 2^{\left(^2\log n\right)-1} + 2^{\left(^2\log n\right)} - 2$$

$= n/2 + n - 2$

$= 3n/2 - 2$

Jadi $T(n) = {}^{3n}\!/_{2} - 2$

$T(n) \in O(n)$

Untuk mengetahui kompleksitas bentuk rekursif, maka
T(n) harus diubah dalam bentuk yang bukan rekursif
Bagaimana mengubah bentuk rekursif ke non rekursif ?
Ada dua macam cara untuk menyelesaikan masalah ini,
yaitu cara coba-coba dan dengan persamaan karakteristik :

1. Cara coba-coba (deret).
2. Metode dengan persamaan karakteristik

Cara coba-coba

Cara ini dilakukan dengan menentukan pola deret yang terbentuk (cara deret). Contoh untuk cara ini telah ditunjukkan dalam mencari kompleksitas waktu untuk beberapa bentuk rekursif sebelumnya. Cara ini agak
sulit dan perlu pengalaman.

$$T(n) = \begin{cases} a & n = 1,2 \\ T(n-1) + T(n-2) + b & n \geq 3 \end{cases}$$

$T(1) = T(2) = a$

$$T(3) = T(1) + T(2) + b = a + a + b = 2a + b$$

$$T(4) = T(2) + T(3) + b = a + 2a + b + b = 3a + 2b$$

$$T(5) = T(3) + T(4) + b = 2a + b + 3a + 2b + b = 5a + 4b$$

$$T(6) = T(4) + T(5) + b = 3a + 2b + 5a + 4b + b$$

$$= 8a + 7b$$

→ Sulit untuk diformulasikan

Metode dengan persamaan karakteristik

Bentuk Persamaan Linier Tak Homogen

Langkah-langkahnya adalah sebagai berikut:

1. Perhatikan bentuk rekursifnya :

$$T(n) = a_1 T(n-1) + a_2 T(n-2) + ... + a_k T(n-k) + f(n)$$

$$f(n) = t^n P_d(n)$$

$$P_d(n) = b_0 n^d + b_1 n^{d-1} + ... + b_k$$

→ polinomial dengan orde / derajat terbesar *d*

→ didapatkan nilai *t dan d*

Metode dengan persamaan karakteristik

Asumsi $f(n)=0$ → bentuk homogen

$$T(n) = a_1 T(n-1) + a_2 T(n-2) + ... + a_k T(n-k)$$

Misal $T(n) = x^n$

$$x^n = a_1 x^{n-1} + a_2 x^{n-2} + ... + a_k x^{n-k}$$

$$x^n - a_1 x^{n-1} - a_2 x^{n-2} - ... - a_k x^{n-k} = 0$$

Persamaan di atas kemudian dibagi dengan x^{n-k} (ini jika x^{n-k} adalah suku dengan orde terkecil), sehingga didapatkan : $x^k - a_1 x^{k-1} - a_2 x^{k-2} - ... - a_k = 0$

3. Diperoleh persamaan karakteristik:

$$\left(x^k - a_1 x^{k-1} - a_2 x^{k-2} - \ldots - a_k\right)(x-t)^{d+1} = 0$$

t dan d didapatkan dari langkah 1

4. Ada 2 macam kasus :

Kasus 1

Semua akar karakteristik berbeda

$$\{x_1, x_2, x_3, \ldots\}$$

Solusi Umum:

$$T(n) = c_1 x_1^{\ n} + c_2 x_2^{\ n} + c_3 x_3^{\ n} + \ldots.$$

c_1, c_2, c_3 adalah konstanta yang harus dicari

Kasus 2

Semua akar karakteristik sama, yaitu $x_1 = x_2 = ... = x_n$

Solusi Umum:

$$T(n) = \left(c_1 + c_2 n + c_3 n^2 + c_4 n^3 + \ldots\right) x^n$$

Masalah faktorial

$$T(n) = \begin{cases} 0 & , n = 0 \\ T(n-1) + 1 & , n > 0 \end{cases}$$

(i) $T(n) = T(n-1) + 1$

$f(n) = 1$

$= 1^n \left(1 . n^0\right)$

$\rightarrow t = 1 \quad d = 0$

(ii) persamaan homogen (anggap f(n)=0)

$$T(n) = T(n-1)$$

$$T(n) - T(n-1) = 0$$

Misal $T(n) = x^n$, maka

$$x^n - x^{n-1} = 0$$

Persamaan terakhir ini dibagi dengan x^{n-1}

(suku dengan orde terkecil), didapatkan :

$\Leftrightarrow x - 1 = 0$

(iii) Persamaan karakteristik $(x - 1)(x - 1) = 0$

Akar – akarnya adalah : $x_1 = 1$ dan $x_2 = 1$

Akar sama, jadi termasuk kasus 2, sehingga solusi

$$T(n) = (c_1 + c_2 n).1^n$$

$$= c_1 + c_2 n$$

Cari c1 dan c2 :

Dari relasi rekurens :

$$T(0) = 0$$

$$T(1) = T(0) + 1 = 1 \quad \} \quad \ldots\ldots\ldots\ldots (*)$$

Dari solusi umum:

$$T(0) = c_1$$

$$T(1) = c_1 + c_2 \quad \} \quad \ldots\ldots\ldots (**)$$

Masalah faktorial

Dari (*) dan (**) didapatkan persamaan :

$$c_1 = 0$$
$$c_1 + c_2 = 1$$

Dari kedua persamaan terakhir ini diperoleh

$c_1 = 0$ dan $c_2 = 1$

Dengan demikian diperoleh :

$$T(n) = c_1 + c_2 n = n$$

Jadi kompleksitas waktunya adalah $T(n) = n$ dan $T(n) \in O(n)$

- Kompleksitas waktu:

  Asumsi: $n = 2^k$

  $T(n)$ = jumlah perbandingan pada pengurutan dua buah upatabel + jumlah perbandingan pada prosedur *Merge*

$$T(n) = \begin{cases} a & , n = 1 \\ 2T(n/2) + cn & , n > 1 \end{cases}$$

Penyelesaian:

$$
\begin{aligned}
T(n) &= 2T(n/2) + cn \\
&= 2(2T(n/4) + cn/2) + cn = 4T(n/4) + 2cn \\
&= 4(2T(n/8) + cn/4) + 2cn = 8T(n/8) + 3cn \\
&= \ ... \\
&= 2^k \, \mathrm{T}(n/2^k) + kcn
\end{aligned}
$$

Berhenti jika ukuran tabel terkecil, $n = 1$:

$$n/2^k = 1 \rightarrow \ \mathrm{k} = {}^2\log n$$

sehingga

$$
\begin{aligned}
T(n) &= nT(1) + cn \ {}^2\log n \\
&= na + cn \ {}^2\log n \\
&= O(n \ {}^2\log n)
\end{aligned}
$$