Analisis Algoritma Rekursif dengan Master Method

Wijayanti N Khotimah

Master Method

- Master method menyediakan sebuah resep untuk menyelesaikan sebuah recurrence dengan bentuk:

$$T(n) = aT(n/b) + f(n) \, ,$$

- Dimana ***a>= 1*** dan ***b>1*** adalah konstan, dan ***f(n)*** adalah fungsi asimtotik positif.

Mengingat Kembali tentang Recurrence

- ✓ recurrence mendeskripsikan running time sebuah algoritma yang membagi problem dengan ukuran ***n*** ke dalam ***a*** buah sub problem dengan ukuran ***n/b****.* Sebanyak ***a*** buah sub problem tersebut diselesaikan secara rekursif dengan masing-masing membutuhkan waktu ***T(n/b)***. Dan ***f(n)*** merupakan cost yang dibutuhkan untuk membagi problem dan menggabungkan hasil dari sub problem tersebut.

[redacted]

- Misal untuk kasus mergesort: $a=2$, $b=2$, $f(n)=\Theta(n)$
- ***Catatan***: terkadang n/b dalam recurrence tidak selalu berupa integer. Tetapi penggunaan term $T(n/b)$ sebagai pengganti $T(\lfloor n/b \rfloor)$ atau $T(\lceil n/b \rceil)$ tidak akan mempengaruhi nilai asimtotik dari recurrence tersebut

Master Theorem

Mater theorem berbunyi sebagai berikut:

- Jika ***a>=1*** dan ***b>a*** adalah konstan, dan ***f(n)*** adalah sebuah fungsi dan ***T(n)*** adalah sebuah recurrence yang dinyatakan dengan:

$$T(n) = aT(n/b) + f(n) ,$$

- Dimana n/b bisa menggantikan T(└n/b┘) atau T(┌n/b┐) maka:

$$1. \, Jika \; f(n) = O\left(n^{\log_b a-\epsilon}\right) untuk \; nilai \; konstanta \; \epsilon > 0, maka \; T(n) = \Theta\left(n^{\log_b a}\right)$$

$$2. \, jika \; f(n) = \Theta\left(n^{\log_b a}\right), maka \; T(n) = \left(n^{\log_b a-\epsilon} lgn\right)$$

$$3. \, Jika \; f(n) = \Omega\left(n^{\log_b a+\epsilon}\right) untuk \; nilai \; konstanta \; \epsilon > 0, dan \; jika \; af\left(\frac{n}{b}\right) \; cf(n) \; untuk \; konstanta$$

$$c < 1 \; dan \; nilai \; n \; yang \; besar, \qquad maka \; T(n) = \Theta(f(n))$$

Penjelasan Master Theorem

1. $Jika\ f(n) = O(n^{\log_b a-\epsilon})$ $untuk\ nilai\ konstanta\ \epsilon > 0, maka\ T(n) = \Theta(n^{\log_b a})$

2. $jika\ f(n) = \Theta(n^{\log_b a}), maka\ T(n) = (n^{\log_b a-\epsilon} lgn)$

3. $Jika\ f(n) = \Omega(n^{\log_b a+\epsilon})$ $untuk\ nilai\ konstanta\ \epsilon > 0, dan\ jika\ af\left(\frac{n}{b}\right)\ cf(n)\ untuk\ konstanta$

$c < 1\ dan\ nilai\ n\ yang\ besar, \quad maka\ T(n) = \Theta(f(n))$

Keterangan:

- Pada ketiga kasus tersebut, kita membandingkan ***f(n)*** dengan ***n^{logba}***. Besarnya kedua fungsi tersebut menentukan solusi dari recurrence.
- Pada kasus 1, persamaan ***$nlog_b a$*** adalah yang lebih besar sehingga solusinya ***T(n)= Θ (n^{logba})***
- Pada kasus 3, fungsi ***f(n)*** lebih besar sehingga solusinya ***T(n)=Θ(f(n))***
- Pada kasus 2, kedua fungsi tersebut berukuran sama sehingga kita mengalikannya dengan faktor logarithmic sehingga solusinya ***T(n)= Θ(n^{logba} lgn)= Θ(f(n)lgn)***

1. $Jika\ f(n) = O\left(n^{\log_b a-\epsilon}\right) untuk\ nilai\ konstanta\ \epsilon > 0, maka\ T(n) = \Theta(n^{\log_b a})$

2. $jika\ f(n) = \Theta\left(n^{\log_b a}\right), maka\ T(n) = (n^{\log_b a-\epsilon} lgn)$

3. $Jika\ f(n) = \Omega\left(n^{\log_b a+\epsilon}\right) untuk\ nilai\ konstanta\ \epsilon > 0, dan\ jika\ af\left(\frac{n}{b}\right)\ cf(n)\ untuk\ konstanta$

$c < 1\ dan\ nilai\ n\ yang\ besar, \quad maka\ T(n) = \Theta(f(n))$

- Catatan:

✓ Pada kasus 1, ***f(n)*** harus lebih kecil dari $n^{\log_b a}$ secara asimtotik dengan faktor n^{ϵ}.

✓ Ketiga kasus tersebut tidak mengcover semua kemungkinan ***f(n)***. Terdapat rentang antara kasus 1 dan kasus 2 ketika ***f(n)*** lebih kecil dari pada $n^{\log_b a}$ tetapi tidak lebih kecil secara polinomial. Begitu juga dengan rentang antara kasus 2 dan kasus 3. Pada kondisi-kondisi seperti ini master theorem tidak **bisa digunakan**

Contoh

$$T(n) = 9T(n/3) + n \; .$$

$$T(n) = 7T(n/2) + \Theta(n^2) \; ,$$

$$T(n) = T(2n/3) + 1 ,$$

$$T(n) = 3T(n/4) + n \lg n \; ,$$

$$T(n) = 2T(n/2) + n \lg n \; ,$$

$$T(n) = 2T(n/2) + \Theta(n) \; ,$$

$$T(n) = 8T(n/2) + \Theta(n^2) \; ,$$

Latihan

- Cormen 4.5