Analisa Algoritma Rekursif dengan Pohon Rekursi

Wijayanti N Khotimah

Pohon Rekursi

- ✓ Meskipun metode substitusi bisa digunakan untuk menyelesaikan recurrence, terkadang kita mengalami **kesulitan dalam menebak pola suatu persamaan**.
- ✓ Pohon rekursi merupakan salah satu cara untuk menyelesaikan persamaan recurrence secara langsung dengan menggambar.
- ✓ Dalam pohon rekursi, satu buah node merepresentasikan cost dari satu buah sub problem.
- ✓ Perhitungan cost pada pohon rekursi adalah dengan menjumlahkan cost pada setiap node dalam 1 level untuk mendapatkan perlevel cost, kemudian perlevelcost tersebut dijumlahkan untuk mendapatkan cost untuk semua level dalam rekursi.

Contoh: Pohon Rekursi pada MergeSort

Recurrence dari mergesort adalah sebagai berikut:

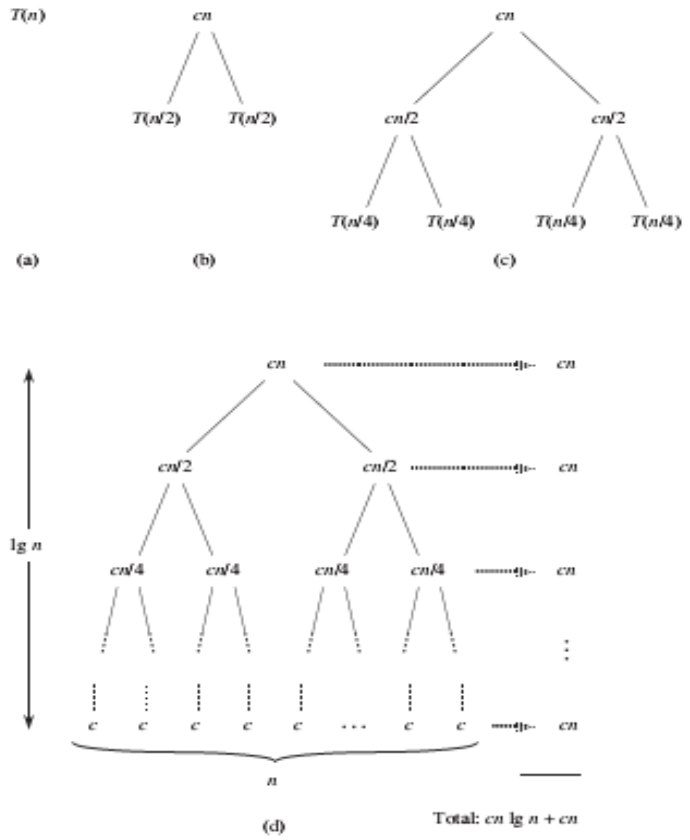
$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 . \end{cases}$$

Misal ***c*** adalah waktu yang dibutuhkan untuk menyelesaikan problem dengan ukuran 1, maka persamaan di atas dapat dinyatakan dengan:

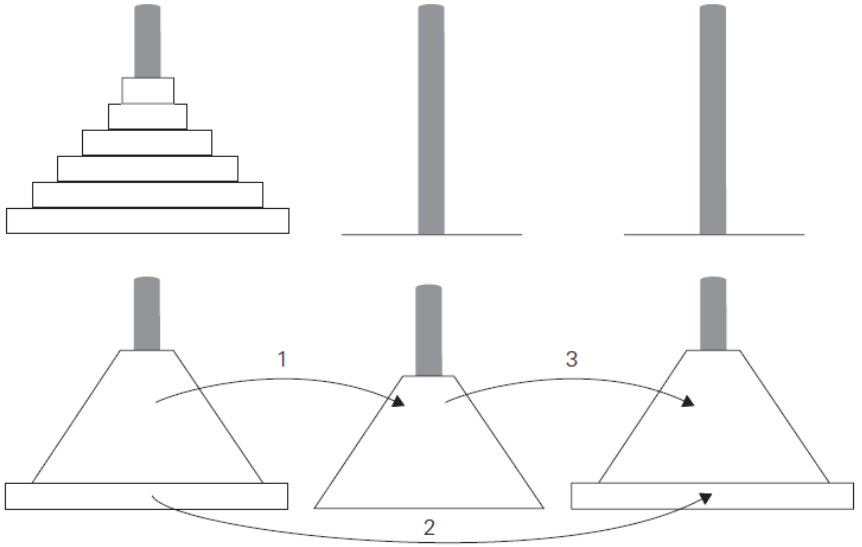
$$T(n) = \begin{cases} c & \text{if } n = 1 , \\ 2T(n/2) + cn & \text{if } n > 1 , \end{cases}$$

Contoh: Pohon Rekursi pada MergeSort

- ✓ Pohon rekursi dari recurrence mergesort bisa digambarkan sbb:
- ✓ Cost untuk masing-masing level adalah *cn*
- ✓ Level ke *i* dari puncak mempunyai jumlah node sebanyak 2^i
- ✓ Pada level paling bawah $2^i=n$, sehingga bisa disimpulkan bahwa jumlah level (i+1) adalah $log_2 n+1$
- ✓ Jadi cost $total=cn(log_2 n+1) = \Theta(nlogn)$



Contoh: algoritma untuk menyelesaikan puzzle Tower of Hanoi



- Solusi rekursif:
  1. pindahkan secara rekursif *n* – 1 keping dari tiang 1 ke tiang 2 dengan memanfaatkan tiang 3 sebagai transisi
  2. pindahkan kepingan terbesar dari tiang 1 ke tiang 3
  3. pindahkan secara rekursif *n* – 1 keping dari tiang 2 ke tiang 3 dengan memanfaatkan tiang 1 sebagai transisi

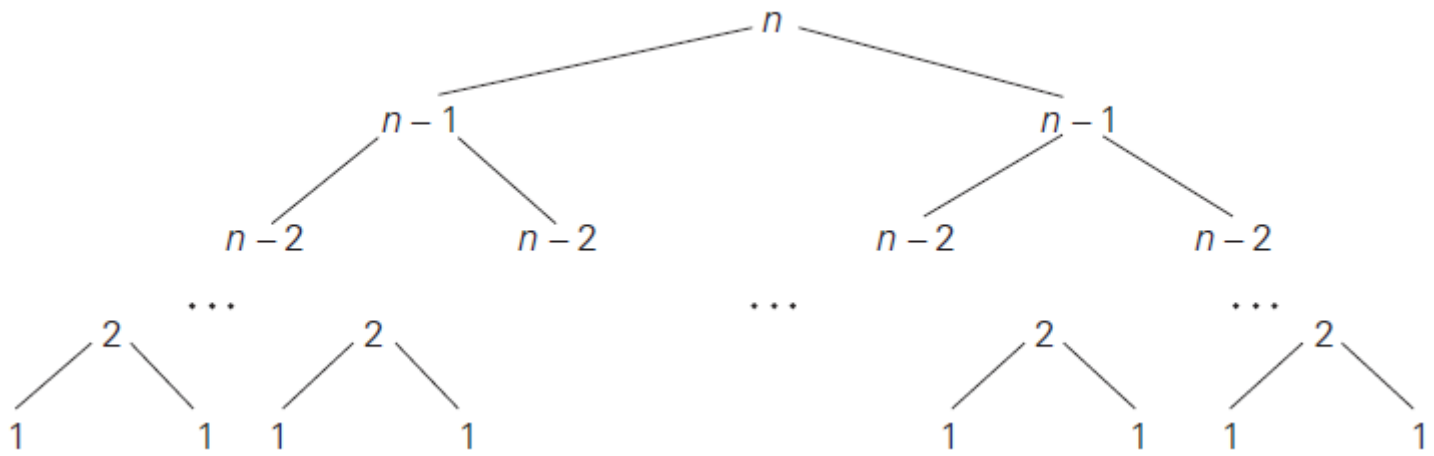
Analisis algoritma rekursif untuk menyelesaikan puzzle Tower of Hanoi

- Parameter apa yang menjadi input size?
- Apa yang menjadi operasi dasar?
- Adakah worst case dan best case pada algoritma tsb?
- Berapa kali operasi dasar dieksekusi dalam bentuk *recurrences*?
  - Apa yang menjadi kondisi awal?
- Apa yang menjadi solusi recurrences tsb?
  - Hint: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

Solving Recurrences (2)

- Pohon Rekursi
  - Susun sebuah *tree* pemanggilan fungsi rekursi
    - ✓Node: pemanggilan fungsi rekursi
    - ✓Label node: parameter pemanggilan fungsi rekursi
  - Hitung jumlah berapa kali pemanggilan fungsi dilakukan deng menghitung jumlah node dalam tree

Contoh pada Tower of Hanoi:



$$C(n) = \sum_{l=0}^{n-1} 2^l \qquad \text{where } l \text{ is the level in the tree} = 2^n - 1$$