REKURENSI

ANALISIS ALGORITMA DAN KOMPLEKSITAS

DIVIDE AND CONQUER

- **Adalah teknik mendisain algoritma dengan cara:**
  - Divide: Memecah instance menjadi beberapa sub-instance kecil terhadap permasalahan yang sama
  - Conquer: Menyelesaikannya secara rekursif dan independen
  - Combine: Menggabungkan solusi-solusi tersebut untuk mendapatkan penyelesaian dari permasalahan awalnya
- **Recursive case: sub-problem yang diselesaikan secara rekursif.**
- **Base case: sub-problem sudah cukup kecil untuk diselesaikan.**
- **Contoh: metode yang dipakai oleh kantor pos dalam mengirimkan surat.**

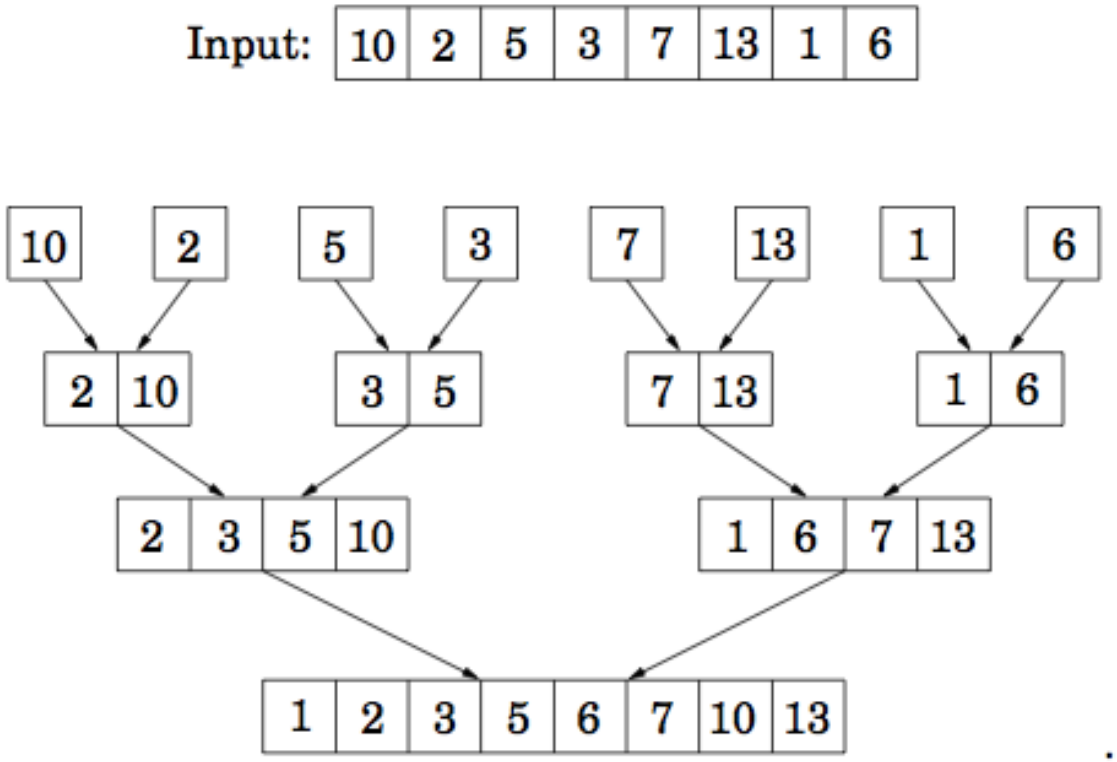
BENTUK UMUM D&C

- **Bentuk umum D&C:**

```
function DC(x)
    if x cukup kecil dan sederhana then return(adhoc)
    else
            Bagi x menjadi instances x1, x2, …, xl.
            for i = 1 to l do yi = DC(xi)
            Gabungkan yi untuk mendapatkan solusi y untuk x.
            return y
```

- **Tiga kondisi :**
  - Harus benar dalam menentukan base case.
  - Harus memungkinkan untuk membagi instance menjadi sub-instance dan menggabungkannya secara efisien.
  - Sub-instances sebaiknya berukuran hampir sama.

CONTOH: MERGESORT



BEBERAPA ASPEK D&C

- **Keuntungan:**
  - Memecahkan masalah yang sulit
  - Efisiensi algoritma
  - Paralelisme
  - Perhitungan yang lebih akurat
- **Isu implementasi:**
  - Rekursi → memilih base case
  - Stack, queue → ukuran stack
- **Bagaimana menghitung running time? → relasi rekurensi**

REKURENSI

- **Jika algoritma memuat panggilan rekursif ke dirinya sendiri, running time-nya dapat digambarkan dengan rekurensi.**
- **Rekurensi: persamaan/pertidaksamaan yang menggambarkan suatu fungsi sebagai nilai-nilainya dengan input yang lebih kecil.**
  - Contoh: worst-case running time dari Merge-Sort:

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

  - Penyelesaiannya adalah $T(n)=\Theta(n \lg n)$.
- **Running time *T*(*n*) hanya terdefinisi untuk bilangan bulat *n*, karena ukuran input selalu bilangan bulat untuk hampir semua algoritma.**
- **Kadang-kadang fungsi floors, ceilings, dan kondisi batas dapat diabaikan pada saat menyelesaikan rekurensi.**

BEBERAPA METODE PENYELESAIAN

- **Substitusi**
  - Menebak batas, kemudian membuktikan dengan induksi matematika.
- **Metode pohon-rekursi**
  - Mengkonversi rekurensi menjadi pohon yang node-nya merepresentasikan biaya.
- **Metode master**
  - Mencari batas rekursi dengan bentuk:

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

    $a \geq 1$, $b > 1$, $f(n)$ adalah fungsi yang diberikan.
  - Mencirikan algoritma D&C dengan *a* sub-problem, masing-masing berukuran 1/b dari ukuran awal, dengan langkah pembagian dan penggabungan memerlukan waktu f(n).

METODE SUBSTITUSI

- **Terdiri dari 2 langkah:**
  - Tebak bentuk penyelesaian.
  - Gunakan induksi matematika untuk menentukan konstanta dan menunjukkan bahwa penyelesaian bisa dipakai.
- **Ketika mengaplikasikan hipotesis induktif ke nilai-nilai yang kecil, tebakan tersebut kita substitusikan → metode substitusi.**
- ***Powerful*, tetapi harus mampu menebak bentuk solusi.**
- **Dapat dipakai untuk menentukan batas bawah maupun batas atas rekurensi.**

CONTOH

- **Jika f(n) adalah jumlah maksimum node daun pada pohon biner dengan tinggi n, maka f(n) = 2 f(n−1), f(0)=1. Buktikan bahwa f(n) = O(2^n).**
- **Bukti :**
  - Base case: f(1)=2 f(0) = 2 x 1 = 2 ≤ c 2^1 dengan c ≥ 1.
  - Langkah induksi:
    - Misalkan f(j) = O(2^j) berlaku untuk j = n-1.
    - Maka (substitusi ke rekurensi awal):

$$f(n) = 2\,f(n-1)$$
$$\leq 2 \times c \times (2^{n-1})$$
$$= c\,2^n$$

berlaku untuk semua n ≥ 1 dan c ≥ 1.

  - Maka terbukti bahwa f(n) = O(2^n).

CONTOH

- **Tentukan batas atas dari rekurensi:**

  $T(n) = 2T(\lfloor n/2 \rfloor) + n$, **dengan** $T(1)=1$

  - Tebakan solusi: $T(n)=O(n \lg n)$
  - Harus dibuktikan bahwa terdapat konstanta positif $c>0$ dan n_0 sedemikian hingga $T(n) \leq cn \lg n$ untuk semua $n \geq n_0$
  - Base case:
    - Jika $n=1$, $T(1) \leq c\ 1 \lg 1=0$ → tidak tepat karena $T(1)=1$
    - Kita dapat mengganti $T(1)$ dengan $T(2)=4$ dan $T(3)=5$ sebagai base case:
      - $T(2) \leq c_1\ 2 \lg 2$ untuk sebarang pilihan $c_1 \geq 2$
      - $T(3) \leq c_1\ 3 \lg 3$ untuk sebarang pilihan $c_1 \geq 2$
    - Maka, kita dapat memilih $c_1 = 2$ and $n_0 = 2$

- Langkah induksi:
  - Misalkan $T(\lfloor n/2 \rfloor) \leq c_2 \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$ untuk $\lfloor n/2 \rfloor$ → mengasumsikan bahwa $T(n)=O(n \lg n)$ berlaku
  - Maka {substitusi ke rekurensi awal $T(n) = 2T(\lfloor n/2 \rfloor)+ n$}:

$$T(n) \leq 2(c_2 \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor))+ n$$
$$\leq c_2 n \lg(n/2)+ n$$
$$= c_2 n \lg n - c_2 n \lg 2 + n$$
$$= c_2 n \lg n - c_2 n + n$$
$$\leq c_2 n \lg n$$

  - Baris terakhir berlaku asalkan $c_2 \geq 1$
- **Terdapat konstanta positif c = max {2, 1}=2 dan $n_0 = 2$ sedemikian hingga $T(n) \leq cn \lg n$ *untuk semua* $n \geq n_0$**
- **Maka terbukti bahwa $T(n)=O(n \lg n)$**

BAGAIMANA CARA MENEBAK SOLUSI?

- **Tidak ada metode tertentu → pengalaman dan kreatifitas.**
- **Dapat dilihat dari kemiripan dengan rekurensi yang sudah ada solusinya.**
  - Misal: $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ → dapat ditebak bahwa solusinya adalah $T(n) = O(n \lg n)$ karena ketika n besar, perbedaan antara $\lfloor n/2 \rfloor$ dan $\lfloor n/2 \rfloor + 17$ tidak signifikan.
- **Dapat juga dengan memilih batas bawah/atas yang *loose*, kemudian sedikit demi sedikit memperbaikinya.**
  - Contoh: $T(n) = 2T(\lfloor n/2 \rfloor) + n$
    - Dapat dimulai dengan batas bawah $T(n) = \Omega(n)$ karena terdapat n dalam rekurens.
    - Batas atas dapat dipilih $T(n) = O(n^2)$.
    - Batas bawah dinaikkan sedikit demi sedikit, batas atas diturunkan sedikit demi sedikit sehingga didapatkan $T(n) = \Theta(n \lg n)$
- **Juga dapat menggunakan pohon rekurensi→ cara berikutnya**

LATIHAN

- **Buktikan bahwa T(n) = O(lg n) untuk T(n) = T(n/2)+1 dengan T(1) = 1.**

METODE POHON REKURSI

Dapat digunakan untuk mendapatkan tebakan yang bagus yang kemudian diverifikasi dengan metode substitusi.

- Pada metode ini, penyederhanaan dilakukan untuk mempermudah penyelesaian.

Dapat juga digunakan sebagai bukti langsung penyelesaian suatu rekurensi.

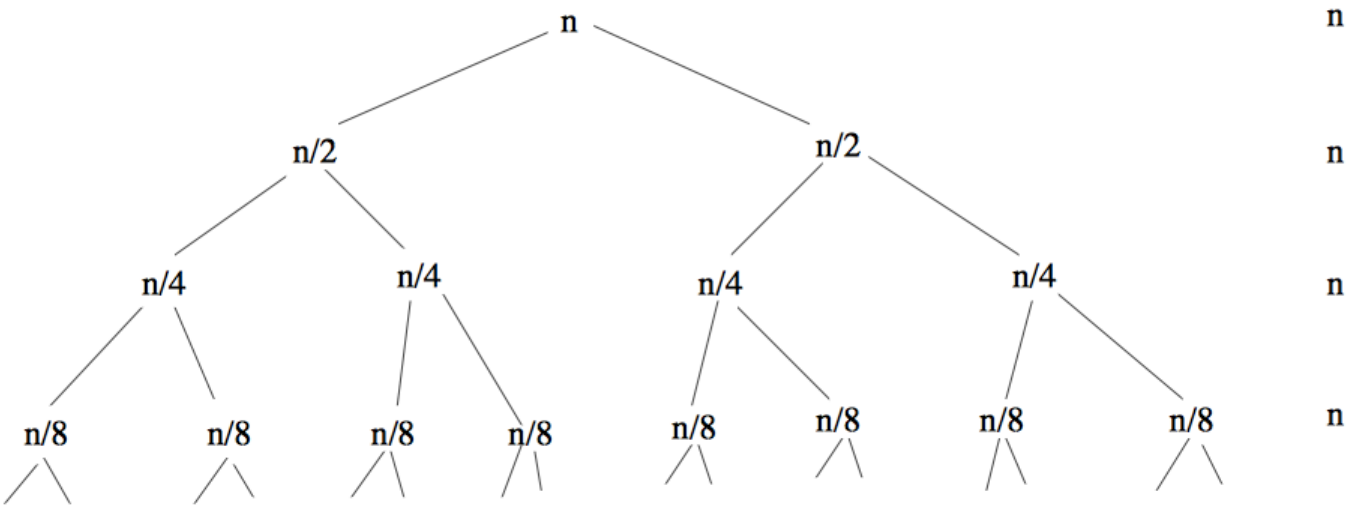
Prinsip:

- Setiap node merepresentasikan cost dari sub-problem tunggal dalam pemanggilan rekursif.
- Pertama, kita jumlahkan cost dari semua node pada setiap level pohon.
- Kemudian, kita jumlahkan cost dari semua level.

CONTOH 1

Bentuk rekurensi dari running time mergesort adalah: T(n) = 2T(n/2) + n, T(1) = O(1).

Cost dari setiap level pohon adalah n.



Depth dari tree:

- Untuk level i : ukuran sub-problem $n/2^i$
- Level terakhir: $n/2^i = 1$ berarti $i = \lg n$.

Jumlah cost setiap level konstan, yaitu n.

Berarti terdapat (lg n + 1) level, yang masing-masing jumlah cost-nya adalah n.

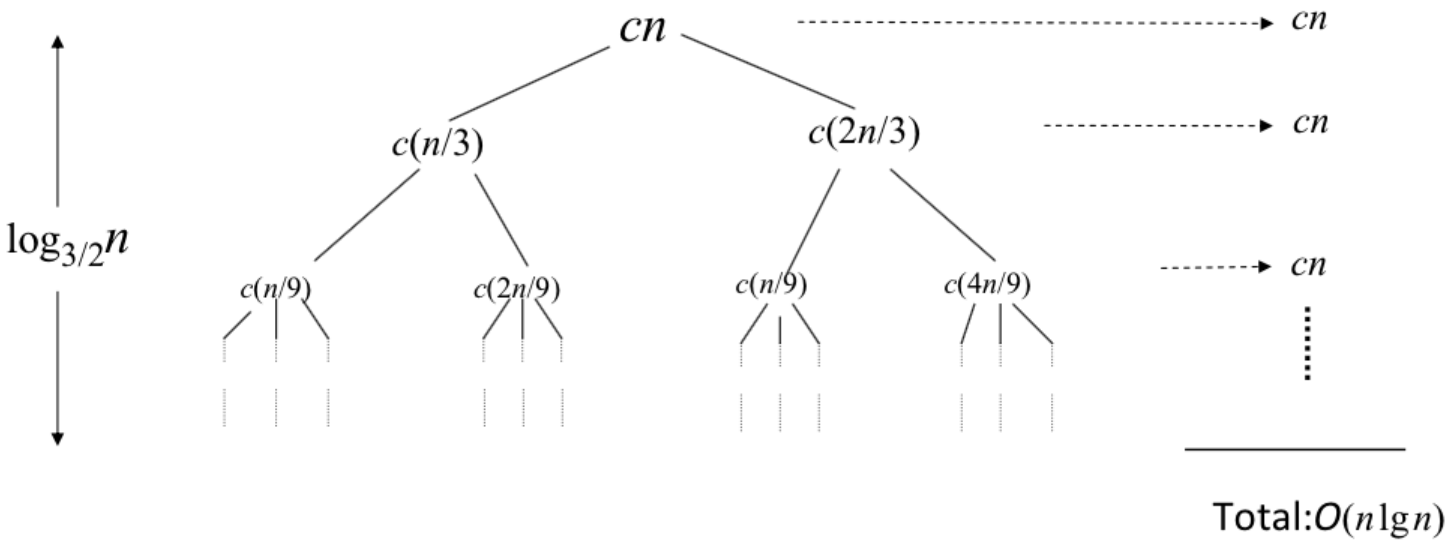
Maka: T(n) = O(n lg n)

Kemudian diperlukan verifikasi dari tebakan dengan metode substitusi.

CONTOH 2

Selesaikan rekurensi: $T(n) = T(n/3)+T(2n/3)+O(n)$.

Pohon rekursi:



Tinggi pohon:

- Path terpanjang untuk mencapai level terbawah dilalui pada node paling kanan pada setiap level: n → 2/3 n → $(2/3)^2$n → $(2/3)^3$n → … → 1
- Maka: 1 = $(2/3)^i$ n → i = $\log_{3/2} n$.

Biaya untuk setiap level: cn

Total biaya : cn (1 + $\log_{3/2} n$) = O(cn $\log_{3/2} n$) = O(n lg n)

Pembuktian dengan metode substitusi (langsung ke langkah induksi):

- Misal berlaku untuk j = n/3 dan j = 2/3, maka:

$$
\begin{aligned}
T(n) &\le T(n/3) + T(2n/3) + cn \\
&\le d(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn \\
&= (d(n/3)\lg n - d(n/3)\lg 3) \\
&\quad + (d(2n/3)\lg n - d(2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn \\
&= dn\lg n - dn(\lg 3 - 2/3) + cn \\
&\le dn\lg n
\end{aligned}
$$

untuk $d \ge c/(\lg 3 - (2/3))$.

CONTOH 3

Selesaikan rekurensi: $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

Penyederhanaan:

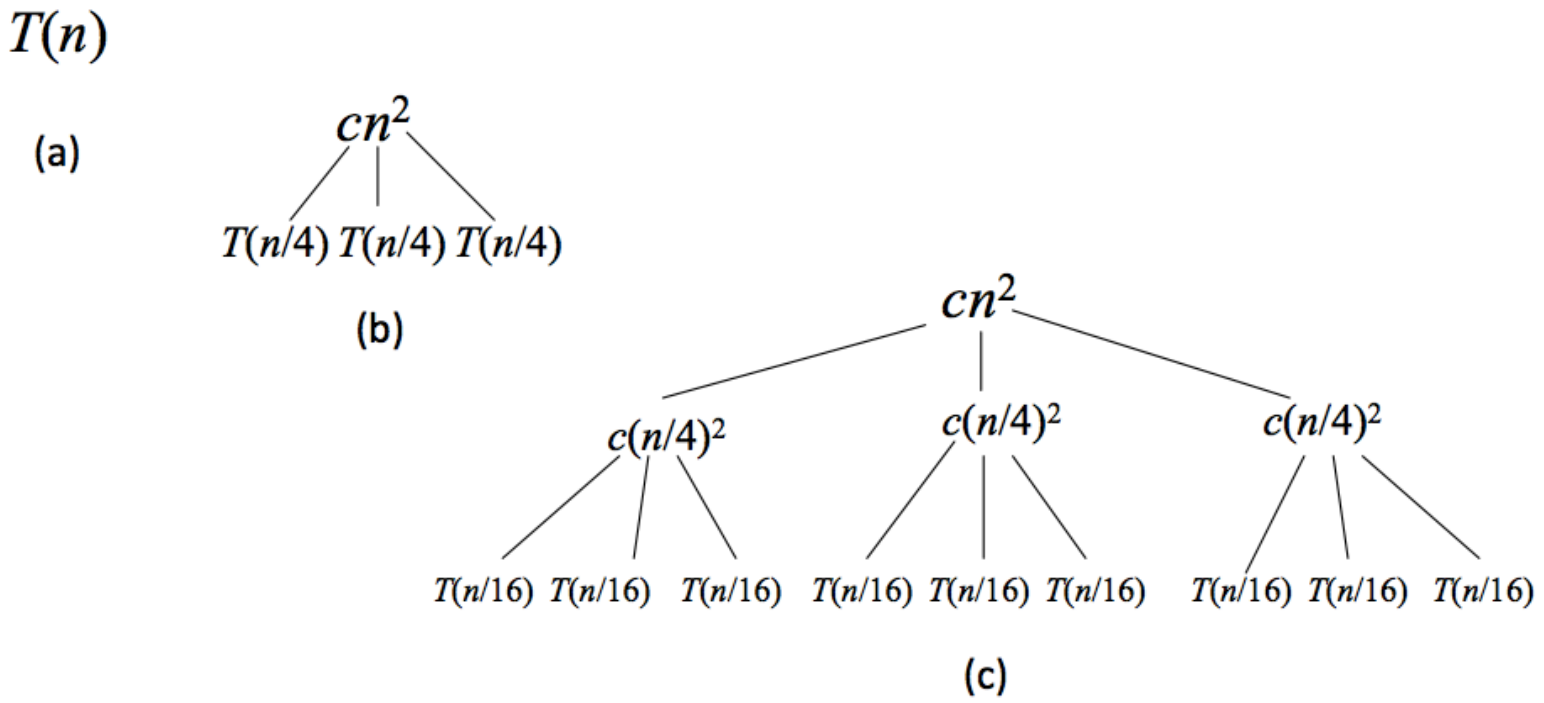
- Abaikan fungsi floor pada rekurensi
- Anggap bahwa *n* merupakan hasil kali dari 4, sehingga ukuran sub-problem merupakan bilangan bulat.

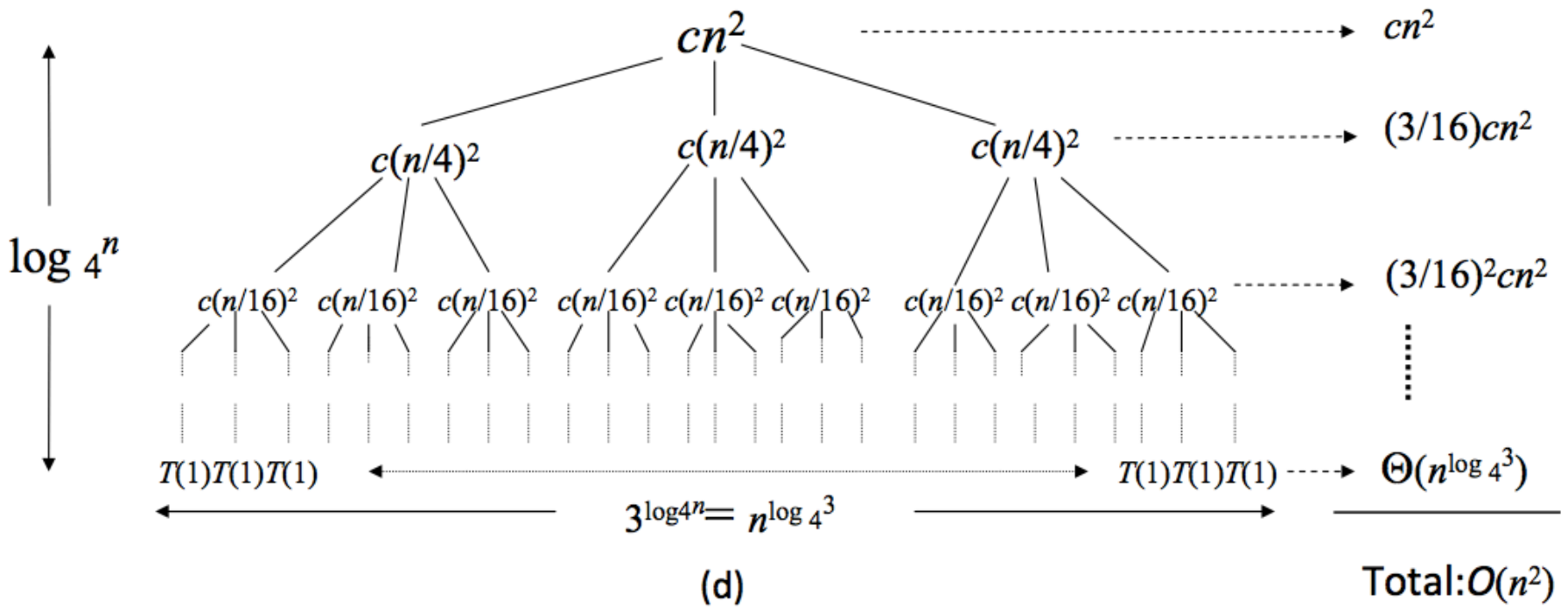
Tulis rekurensi sebagai $T(n) = 3T(n/4) + cn^2$

PEMBENTUKAN POHON REKURSI

$T(n) = 3T(n/4) + cn^2$

Ekspansi langkah demi langkah pohon rekursi:




cn^2
cn^2
$c(n/4)^2$
$c(n/4)^2$
$c(n/4)^2$
$(3/16)cn^2$
$\log_4 n$
$c(n/16)^2$ $c(n/16)^2$ $c(n/16)^2$ $c(n/16)^2$ $c(n/16)^2$ $c(n/16)^2$ $c(n/16)^2$ $c(n/16)^2$ $c(n/16)^2$
$(3/16)^2cn^2$
$T(1)T(1)T(1)$
$T(1)T(1)T(1)$
$\Theta(n^{\log_4 3})$
$3^{\log_4 n} = n^{\log_4 3}$
(d)
Total:$O(n^2)$

MENENTUKAN COST (BIAYA) DARI POHON

Ukuran sub-problem untuk node pada level *i* adalah $n/4^i$.

- Ukuran sub-problem sama dengan 1 (level terendah) pada saat $n/4^i = 1 \rightarrow n = 4^i \rightarrow i = \log_4 n$.
- Maka, pohon terdiri dari $\log_4 n + 1$ level (0, 1, 2,..., $\log_4 n$).

Setiap node pada level *i* memiliki cost $c(n/4^i)^2$ untuk $0 \leq i \leq \log_4 n - 1$.

Maka, total cost semua node pada level *i* adalah $3^i * c(n/4^i)^2 = (3/16)^i cn^2$.

Level terakhir, yaitu level $\log_4 n$, memiliki $3^{\log_4 n} = n^{\log_4 3}$ nodes.

Cost dari keseluruhan pohon:

$$T(n) = cn^2 + \frac{3}{16}cn^2 + (\frac{3}{16})^2 cn^2 + \ldots + (\frac{3}{16})^{\log_4 n-1} cn^2 + \theta(n^{\log_4 3})$$

$$= \sum_{i=0}^{\log_4 n-1} (\frac{3}{16})^i cn^2 + \Theta(n^{\log_4 3})$$

$$< \sum_{i=0}^{\infty} (\frac{3}{16})^i cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3})$$

$$= O(n^2)$$

Maka, tebakan untuk penyelesaian $T(n)=O(n^2)$.

Note: $\Sigma_{k=0}^{\infty} x^k = 1/(1-x)$

VERIFIKASI KEBENARAN DARI TEBAKAN

Selanjutnya dapat digunakan metode substitusi untuk memverifikasi bahwa tebakan kita benar.

- Akan ditunjukkan bahwa T(n) ≤ dn^2 untuk suatu konstanta $d > 0$ dan $n \geq n_0$.

Base case:

- Untuk n = 1 → $T(n) = 3T(\lfloor n/4 \rfloor) + cn^2$, diasumsikan T(0) = 0
  - T(1) = 3T(0) + c 1 ≤ $d\ (1)^2$ → berlaku untuk d > 0, d ≥ c.

Langkah induksi:

- Asumsikan bahwa T(j) ≤ dj^2 berlaku untuk suatu j = n/4

$$
\begin{aligned}
T(n) &\leq 3T(\lfloor n/4 \rfloor) + cn^2 \\
&\leq 3d\lfloor n/4 \rfloor^2 + cn^2 \\
&\leq 3d(n/4)^2 + cn^2 \\
&= 3/16dn^2 + cn^2 \\
&\leq dn^2,
\end{aligned}
$$

dengan $d \geq (16/13)c$.

Dengan demikian, terbukti bahwa $T(n)=O(n^2)$.

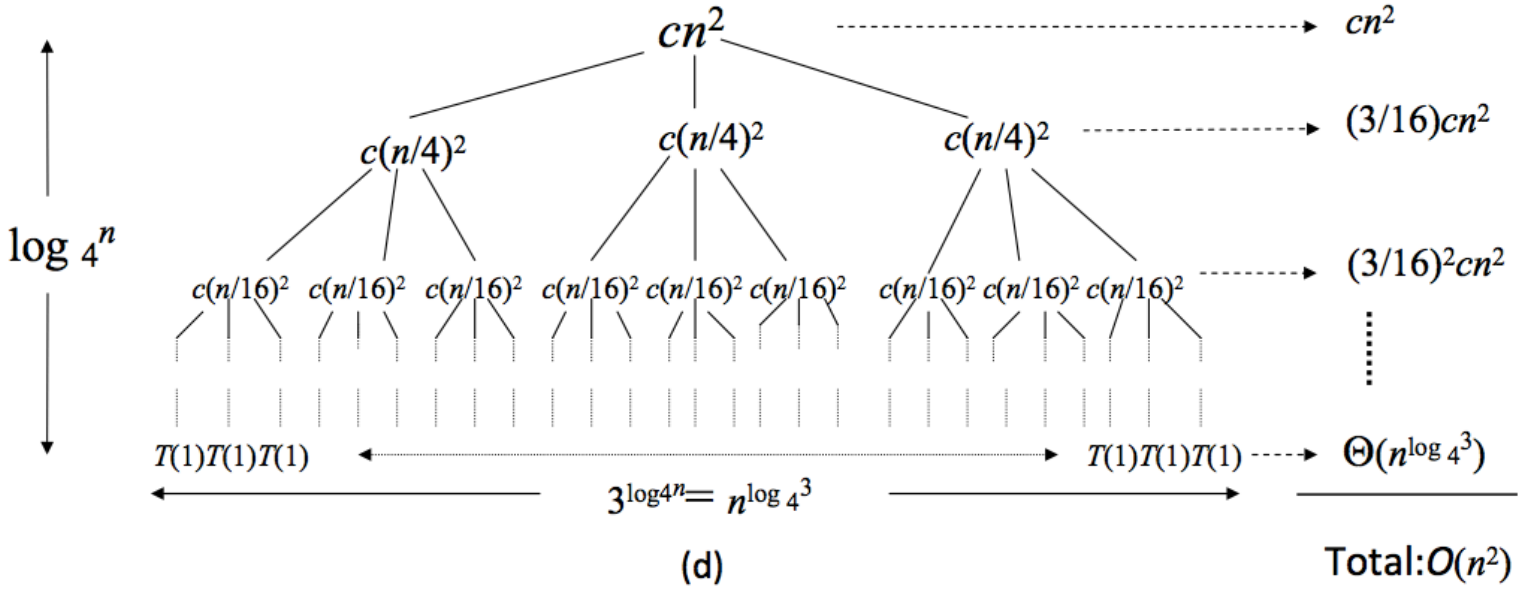
GARIS BESAR POHON REKURENSI UNTUK BENTUK TERTENTU

Misalkan terdapat rekurensi: T (n) = a T (n/b) + f (n)

- Contoh di bawah: $T(n) = 3T(n/4) + cn^2$

Pohon rekursinya akan berbentuk sbb:

- Root akan memuat nilai f(n).
- Jumlah child adalah a, masing-masing memuat nilai f(n/b). Kemudian, setiap node pada child ini mempunyai child dengan nilai f(n/b^2).
- Secara umum, level i terdiri dari a^i node dengan nilai f(n/b^i).



- Maka jumlah cost pada level i adalah $a^i f(n/b^i)$.
- Jika diasumsikan bahwa $T(1) = f(1) = \Theta(1)$, maka depth dari tree adalah: $\log_b n$, jadi terdapat : $\log_b n+1$ level.

Maka: $T(n) = f(n)+a\,f(n/b)+a^2\,f(n/b^2)+\cdots+a^i f(n/b^i)+\cdots+a^L f(n/b^L)$

- $L = \log_b n$ adalah depth dari pohon.
- Karena $f(1) = \Theta(1)$, maka term terakhir dari jumlahan adalah: $\Theta(a^L) = \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$.

METODE MASTER

- **Algoritma-algoritma divide and conquer sering menghasilkan rekurensi untuk running time dalam bentuk: T(n) = aT(n/b) + f(n), dengan a dan b konstanta dan f(n) fungsi lain dari n.**
- **Master Method (Metode Master) dapat dimanfaatkan untuk menyelesaikan rekurensi dengan bentuk demikian jika f(n) berupa polinomial sederhana.**
- **Cara penyelesaiannya berdasarkan Teorema Master.**
  - Merupakan kasus khusus dari penggunaan pohon rekursi.
- **Teorema Master tidak bisa digunakan untuk semua fungsi f(n) → Metode Master kurang 'powerful' dibandingkan pohon rekurensi**
  - Tidak semua T(n) berbentuk seperti T(n) = aT(n/b) + f(n).
- **Jika teorema dapat digunakan, rekurensi dapat diselesaikan dengan cara cepat.**

TEOREMA MASTER

Misalkan $a \geq 1$ dan $b > 1$ adalah konstanta, f(n) suatu fungsi dan T(n) didefinisikan sebagai bilangan bulat non-negatif dengan rekurensi:

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

Maka T(n) memiliki batas asimtotik sbb:

- Jika $f(n) = O(n^{\log_b a - k})$ utk suatu $k > 0$, maka $T(n) = \Theta(n^{\log_b a})$.
  - f(n) dibandingkan dengan $n^{\log_b a}$, dan ternyata $n^{\log_b a}$ lebih besar.
- Jika $f(n) = \Theta(n^{\log_b a})$, maka $T(n) = \Theta(n^{\log_b a} \lg n)$.
  - f(n) dibandingkan dengan $n^{\log_b a}$, dan ternyata keduanya memiliki ukuran yang sama.
- Jika $f(n) = \Omega(n^{\log_b a+k})$ utk suatu $k > 0$, dan jika $a\,f(n/b) \leq c\,f(n)$ untuk suatu konstanta $c < 1$ dengan n cukup besar, maka $T(n) = \Theta(f(n))$.
  - f(n) dibandingkan dengan $n^{\log_b a}$, dan ternyata f(n) lebih besar.

Syarat-syarat dapat dipakainya teorema Master:

- Pada kasus 1, selain f(n) harus lebih kecil dari $n^{\log_b a}$, f(n) juga harus lebih kecil secara polinomial. Jadi f(n) harus lebih kecil secara asimtotik dengan faktor n^k untuk suatu $k > 0$.
- Pada kasus 3, selain f(n) harus lebih besar dari $n^{\log_b a}$, f(n) juga harus lebih besar secara polinomial, juga harus memenuhi kondisi bahwa $a\, f(n/b) \leq c\, f(n)$.

Teorema Master tidak dapat dipakai jika:

- f(n) lebih kecil dari $n^{\log_b a}$, tetapi f(n) tidak lebih kecil secara polinomial.
- f(n) lebih besar dari $n^{\log_b a}$, tetapi f(n) tidak lebih besar secara polinomial atau tidak memenuhi kondisi $a\, f(n/b) \leq c\, f(n)$.

CONTOH 1

Misalkan terdapat rekurensi $T(n) = 9\,T(n/3) + n$.

Maka $a = 9$, $b = 3$, $f(n) = n$, sehingga:

$$n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$$

Karena $f(n) = O(n^{\log_3 9 - k})$ dengan $k = 1$, kita dapat menggunakan kasus 1 Teorema Master.

Jadi penyelesaiannya adalah $T(n) = \Theta(n^2)$.

CONTOH 2

Misalkan terdapat rekurensi $T(n) = T(2n/3) + 1$.

Maka $a = 1$, $b = 3/2$, $f(n) = 1$, sehingga:

$$n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1.$$

Karena $f(n) = O(n^{\log_b a})$, kita dapat menggunakan kasus 2 Teorema Master.

Jadi penyelesaiannya adalah $T(n) = \Theta(1 \times \lg n) = \Theta(\lg n)$.

CONTOH 3

Misalkan terdapat rekurensi $T(n) = 2\,T(n/2) + n \lg n$.

Maka $a = 2$, $b = 2$, $f(n) = n \lg n$, sehingga:

$$n^{\log_b a} = n^{\log_2 2} = n$$

Apakah $f(n)$ secara polinomial lebih besar daripada $n^{\log_b a}$?

Rasio $f(n) / n^{\log_b a} = (n \lg n) / n = \lg n$, secara asimtotik lebih kecil dibandingkan dengan n^k untuk $k > 0$.

Jadi teorema Master tidak dapat dipakai untuk menyelesaikan permasalahan ini.

LATIHAN

1. **Gunakan pohon rekursi untuk menentukan upper bound dari rekurensi $T(n) = 2T(n - 1) + 1$, $T(1) = 1$. Gunakan metode substitusi untuk memverifikasi jawaban Anda.**
2. **Gunakan metode master untuk mencari tight asymptotic bound dari rekurensi $T(n) = 2T(n/4) + 1$.**