Design and Analysis of Algorithms

L10b: Recurrence Relation

Master Theorem

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT/KSSEM
rprustagi@ksit.edu.in

Divide and Conquer: Recurrence Relation

$$T(n) = \begin{cases} g(n) & n \ small \\ T(n_1) + T(n_2) + \ldots + T(n_k) + f(n) & otherwise \end{cases}$$

- `T(n)`: time complexity for a problem of input size n
- `g(n)`: time complexity for solving directly for small inputs
- `f(n)`: Time complexity for dividing the problem into `k` subproblems and combining again from the solutions of `k` sub problems.
- `k` would vary depending upon the problem
  - Generally, $n_1=n_2=\ldots=n_k$
  - Assuming `a` instances, each of size `n/b`

$$T(n) = \begin{cases} T(1) & n = 1 \\ aT(n/b) + f(n) & n > 1 \end{cases}$$

Matrix Multiplication

- Conventional matrix multiplication

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ : & & & : \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ : & & & : \\ b_{n1} & b_{n2} & \dots & bn_{n} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ : & & & : \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{bmatrix}$$

where the element c_{ij} is computed as

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + a_{i3}b_{3j} + \dots + a_{in}b_{nj}$$

- computations required for c_{ij}
  - Multiplications: n
  - additions: n
- Total computations required for matrix multiplication:
  - $2n^3$ i.e.
  - $\Theta(n^3)$

Matrix Multiplication

- Conventional matrix multiplication

$$\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} * \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

$$= \begin{bmatrix} A_1B_1+A_2B_3 & A_1B_2+A_2B_4 \\ A_3B_1+A_4B_3 & A_3B_2+A_4B_4 \end{bmatrix}$$

- Multiplications: 8 (=2^3), Additions: 4
- Recurrence relation:

$$T(n)=8T(n/2)+4(n/2)^2$$
$$= 2^3T(n/2)+O(n^2) \quad = \Theta(n^3)$$

- Recurrence relation general form:

$T(n)=aT(n/b)+\Theta(n^d)$ for $n=b^k$, $k=1,2,$

$T(1) = c$, where, $a\geq1$, $b\geq2$, $c>0$

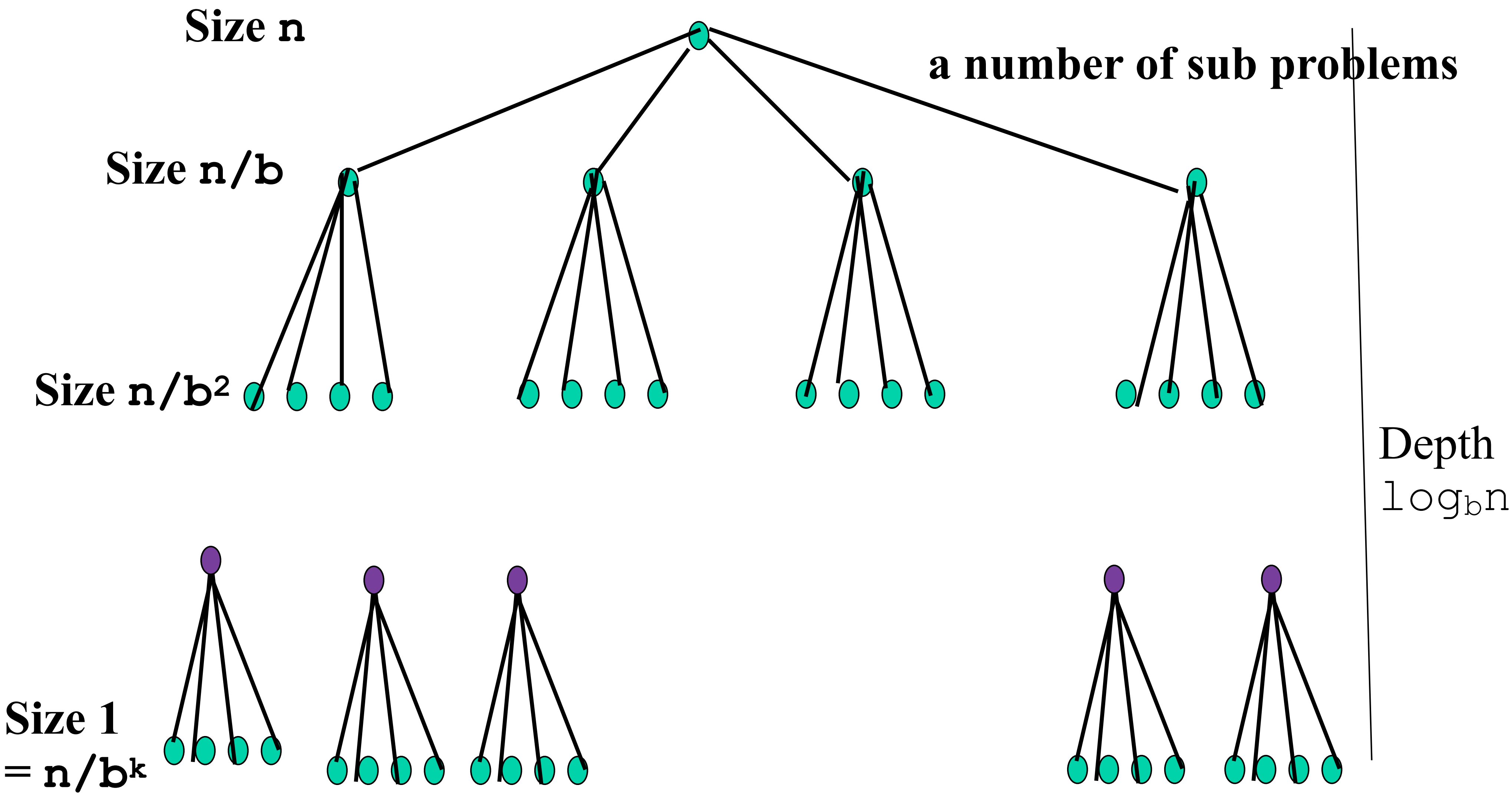
Recurrence Relation: Master Theorem

T(n)=aT(n/b)+Θ(n^d) for **n=b^k, k=1,2,**

T(1) = c, where, **a≥1, b≥2, c>0**

$$T(n) \in \begin{cases} \Theta(n^d) & if\ a < b^d \\ \Theta(n^d log n) & if\ a = b^d \\ \Theta(n^{log_b a}) & if\ a > b^d \end{cases}$$

Proof Outline: Master Theorem



Width $a^{\log_b n} = n^{\log_b a}$

Proof Outline: Master Theorem

- With each level of recursion
  - Size of subproblem decreases by a factor of b
  - Reaches base case after $\log_b n$ levels.
    - Height of recursion tree
  - Branch factor is a i.e. number sub problems increase by a factor of a.
- Number of subproblems
  - Level 1: a, each of size n/b^1
  - Level 2: a^2, each of size n/b^2
  - :
  - Level k: a^k, each of size n/b^k
- Work done at level k is
  - $a^k * O(n/b^k)^d = O(n^d) * (a/b^d)^k$

Proof Outline: Master Theorem

- Work done at level k is
  - $a^k * O(n/b^k)^d = O(n^d)*(a/b^d)^k$
- As k from 0 (the root) to $\log_b n$ (the leaves)
  - The numbers form geometric series with ratio (a/b^d)
- Geometric sum boils to 3 cases
- Ratio less than i.e. $(a/b^d)<1$
  - Series is decreasing, thus sum is given by 1st term $O(n^d)$
- Ratio is exact 1 i.e. $(a/b^d)=1$
  - All $\log_b n$ terms of series are equal to $O(n^d)$
    - Sum becomes $O(n^d \log_b n)$
- Ratio is greater than 1 i.e. $a/b^d)>1$
  - Series is increasing and sum is given by last term

$$n^d*(a/b^d)^k = n^d*(a^{\log_b n}/(b^{\log_b n})^d)$$
$$= n^d*(a^{\log_b n}/n^d)=a^{\log_b n} = n^{\log_b a}$$

Recurrence Relation: Master Theorem

T(n)=aT(n/b)+Θ(n^d) for **n=b^k, k=1,2,**
T(1) = c, where, **a≥1, b≥2, c>0**

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Recurrence :MaxMin Algo

- Recurrence relation for MaxMin

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

$T(1) = 0$

$T(2) = 1$

Using the Master theorem

$a=2\ (a\geq1),\ b=2\ (b\geq2),\ c=T(1)=0$, and

$f(n)=2\ \in\Theta(n^d)\ \Rightarrow\ f(n)\in\Theta(1)\ \Rightarrow\ d=0$

Thus, $b^d=b^0=1\ \Rightarrow a>b^d$ #3rd case in Master Theorem

Further, $\log_b a=\log_2 2=1$, thus from Master Theorem

$T(n)=\Theta(n^{\log_b a})\ =\ \Theta(n^1)\ =\ \Theta(n)$

which corresponds to $3n/2\ -\ 2$

Solving Recurrence Relation

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

- Let `n=b`^{k}, then

$$T(b^k) = aT(b^{k-1}) + f(b^k)$$

$$= a[aT(b^{k-2}) + f(b^{k-1})] + f(b^k)$$

$$= a^2T(b^{k-2}) + af(b^{k-1}) + f(b^k)$$

$$= a^3T(b^{k-3}) + a^2f(b^{k-2}) + af(b^{k-1}) + a^0f(b^k)$$

$$\vdots$$

$$= a^kT(b^{k-k}) + a^{k-1}f(b^{k-(k-1)} + a^2f(b^{k-2}) + af(b^{k-1}) + a^0f(b^k)$$

$$= a^kT(1) + a^{k-1}f(b^1) + a^{k-2}f(b^2) + \ldots + a^0f(b^k)$$

$$= a^k[T(1) + f(b^1)/a^1 + f(b^2)/a^2 + \ldots + f(b^k)/a^k]$$

Solving Recurrence Relation

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

$$T(b^k) = aT(b^{k-1}) + f(b^k)$$

$$= a^k[T(1) + f(b^1)/a^1 + f(b^2)/a^2 + \ldots + f(b^k)/a^k]$$

$$= a^k[T(1) + \sum_{j=1}^{k} \frac{f(b^j)}{a^j}]$$

- Thus, `T(n)` depends upon `a`, `b`, and `f()`

As $n=b^k$, then $k=\log_b n$, thus

$a^k = a^{\log_b n} = n^{\log_b a}$, the recursion equation becomes

$$T(n) = n^{log_b a}[T(1) + \sum_{j=1}^{log_b n} \frac{f(b^j)}{a^j}] \qquad (1)$$

Recurrence :MaxMin Algo

- Recurrence relation for MaxMin

```
T(n)=2T(n/2)+2
T(1) = 0
T(2) = 1
```

Note: we can't apply general recurrence relation to T(2) i.e. we can't write

`T(2)=2T(2/2)+2=2T(1)+2=0+2=2,`

Hence we need to stop at `T(2)` and can't go to `T(1)`.

Thus, recurrence relation becomes

$$T(n) = n^{log_b a}[\frac{T(2)}{a} + \sum_{j=2}^{log_b n} \frac{f(b^j)}{a^j}]$$

$$= n^{log_2 2}[\frac{1}{2} + \sum_{j=2}^{log_b n} \frac{2}{2^j}] \quad = n[\frac{1}{2} + \sum_{j=2}^{log_b n} \frac{1}{2^{j-1}}]$$

Recurrence :MaxMin Algo

$$T(n) = n[\frac{1}{2} + \sum_{j=2}^{log_b n} \frac{1}{2^{j-1}}]$$

$$= n[\frac{1}{2} + \frac{\frac{1}{2} - (\frac{1}{2})^{log_2 n}}{1 - \frac{1}{2}}] \qquad = n[\frac{1}{2} + \frac{\frac{1}{2} - (\frac{1}{n})}{\frac{1}{2}}]$$

$$= n[\frac{1}{2} + 1 - \frac{2}{n}] \qquad = n[\frac{3}{2} - \frac{2}{n}]$$

$$= \frac{3}{2}n - 2$$