COMP3251
Lecture 3: Master Theorem
(Chapter 2.2)

A General Theorem for Solving Recurrence Relations

The most important formula that you need to remember in this chapter.

Master Theorem. If $T(n) = a\,T(n/b) + O(n^d)$ for some constants $a > 0$, $b > 1$, and $d \geq 0$, then

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

A General Theorem for Solving Recurrence Relations

The most important formula that you need to remember in this chapter.

divide the problem into *a* subproblems

Master Theorem. If $T(n) = a\,T(n/b) + O(n^d)$ for some constants $a > 0$, $b > 1$, and $d \geq 0$, then

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

A General Theorem for Solving Recurrence Relations

The most important formula that you need to remember in this chapter.

each subproblem has size ***n/b***

divide the problem into ***a*** subproblems

Master Theorem. If $T(n) = a\,T(n/b) + O(n^d)$ for some constants $a > 0$, $b > 1$, and $d \geq 0$, then

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

A General Theorem for Solving Recurrence Relations

The most important formula that you need to remember in this chapter.

each subproblem has size ***n/b***

divide the problem into ***a*** subproblems

dividing and combining takes ***O(n^d)*** time

Master Theorem. If $T(n) = a\,T(n/b) + O(n^d)$ for some constants $a > 0$, $b > 1$, and $d \geq 0$, then

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

A General Theorem for Solving Recurrence Relations

The most important formula that you need to remember in this chapter.

each subproblem has size ***n/b***

divide the problem into ***a*** subproblems

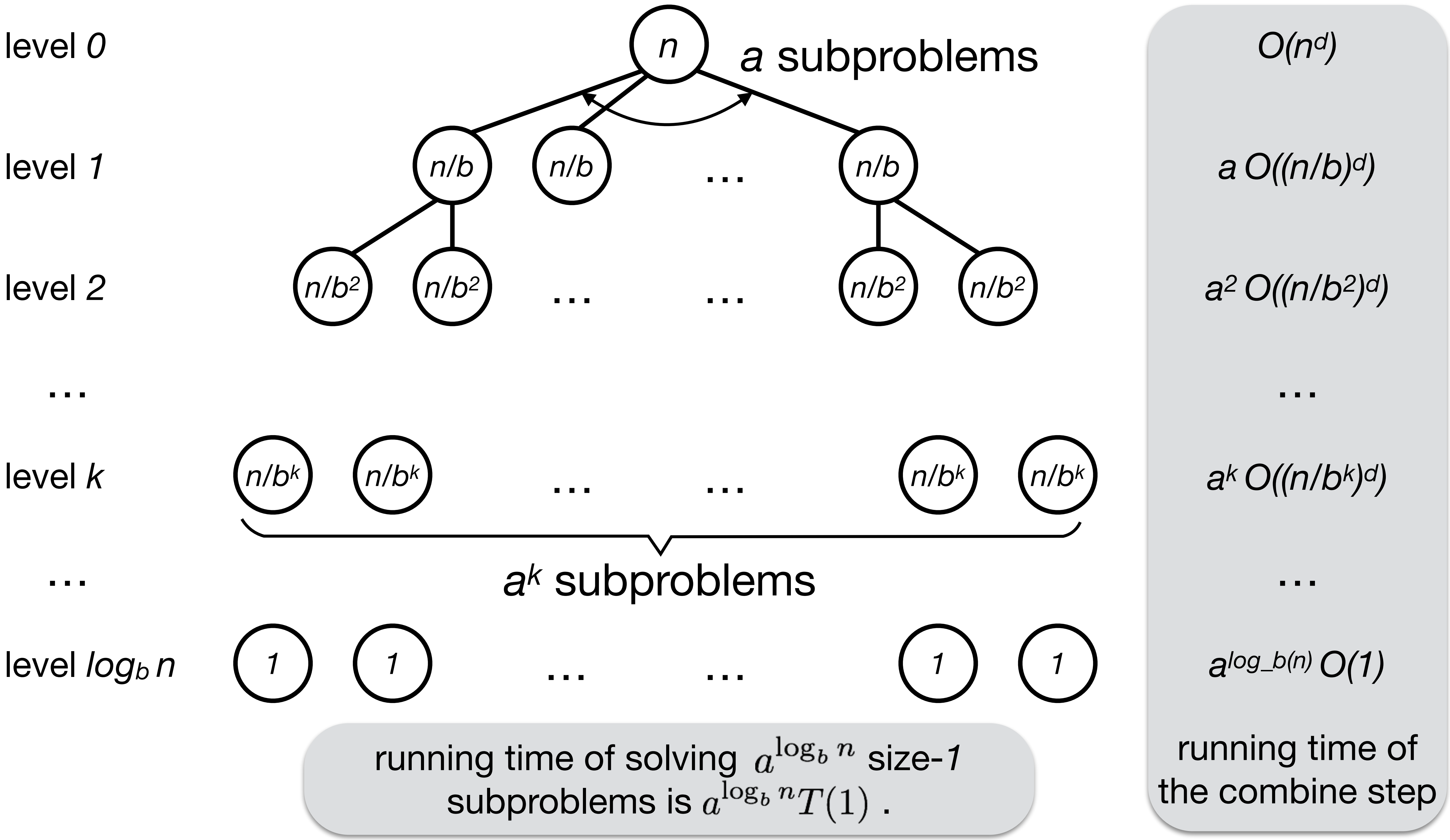
dividing and combining takes ***O(n^d)*** time

Master Theorem. If $T(n) = a\,T(n/b) + O(n^d)$ for some constants $a > 0$, $b > 1$, and $d \geq 0$, then

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

Implicit assumptions: (1) constant-size subproblems can be solved in $O(1)$ time; (2) all subproblems have the same size.

Proof of the Master Theorem for $T(n) = a\,T(n/b) + O(n^d)$



Proof of the Master Theorem for $T(n) = a\,T(n/b) + O(n^d)$

- The running time of solving size-*1* subproblems is

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

- The total running time of the combine steps (for all levels) is

$$O(n^d) + a \cdot O\big((n/b)^d\big) + \cdots + a^k \cdot O\big((n/b^k)^d\big) + \cdots + a^{\log_b n} \cdot O\big((n/b^{\log_b n})^d\big)$$

- After simplification, it is equal to

$$O(n^d)\big(1 + a/b^d + \cdots + (a/b^d)^k + \cdots + (a/b^d)^{\log_b n}\big)$$

Case 1: $a < b^d$, namely, $\log_b a < d$.
Then, the first term dominates the above sum:

- Running time of the combine steps is $O(n^d)$.
- So $T(n) = O(n^{\log_b a}) + O(n^d) = O(n^d)$.

Proof of the Master Theorem for $T(n) = a\,T(n/b) + O(n^d)$

- The running time of solving size-*1* subproblems is

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

- The total running time of the combine steps (for all levels) is

$$O(n^d) + a \cdot O\big((n/b)^d\big) + \cdots + a^k \cdot O\big((n/b^k)^d\big) + \cdots + a^{\log_b n} \cdot O\big((n/b^{\log_b n})^d\big)$$

- After simplification, it is equal to

$$O(n^d)\big(1 + a/b^d + \cdots + (a/b^d)^k + \cdots + (a/b^d)^{\log_b n}\big)$$

Case 2: $a > b^d$, namely, $\log_b a > d$.
Then, the last term dominates the above sum:

- Running time of the combine steps is
$O\big(n^d (a/b^d)^{\log_b n}\big) = O\big(n^d a^{\log_b n} / n^d\big) = O\big(a^{\log_b n}\big) = O\big(n^{\log_b a}\big)$
- So $T(n) = O\big(n^{\log_b a}\big) + O\big(n^{\log_b a}\big) = O\big(n^{\log_b a}\big)$.

Proof of the Master Theorem for $T(n) = a\,T(n/b) + O(n^d)$

- The running time of solving size-*1* subproblems is

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

- The total running time of the combine steps (for all levels) is

$$O(n^d) + a \cdot O\big((n/b)^d\big) + \cdots + a^k \cdot O\big((n/b^k)^d\big) + \cdots + a^{\log_b n} \cdot O\big((n/b^{\log_b n})^d\big)$$

- After simplification, it is equal to

$$O(n^d)\big(1 + a/b^d + \cdots + (a/b^d)^k + \cdots + (a/b^d)^{\log_b n}\big)$$

Case 3: $a = b^d$, namely, $\log_b a = d$.
Then, all *log n* terms equal *1* in the above sum:

- Running time of the combine steps is $O(n^d \log n)$.
- So $T(n) = O\big(n^{\log_b a}\big) + O\big(n^d \log n\big) = O\big(n^d \log n\big)$.

An Alternative Form of the Master Theorem

each subproblem has size ***n/b***

divide the problem into ***a*** subproblems

dividing and combining takes ***O(n^d)*** time

Master Theorem. If $T(n) = a\,T(n/b) + O(n^d)$ for some constants $a > 0$, $b > 1$, and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^{\max\{d,\ \log_b a\}}) & \log_b a \neq d \\ O(n^d \log n) & \log_b a = d \end{cases}$$

Example: Running Time of Merge Sort

Master Theorem. If $T(n) = a\,T(n/b) + O(n^d)$ for some constants $a > 0$, $b > 1$, and $d \geq 0$, then

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

Recall that:

- Merge Sort divides the problem into two *n/2*-size subproblems.
- Merging two *n/2*-size sorted lists takes O(n) time.

Hence,

- $a = 2$, $b = 2$, $d = 1$.
- The running time of Merge Sort is $T(n) = O(n \log n)$.