COMP20007 Design of Algorithms

Master Theorem

Lars Kulik

Lecture 10

Semester 1, 2020

Divide and Conquer

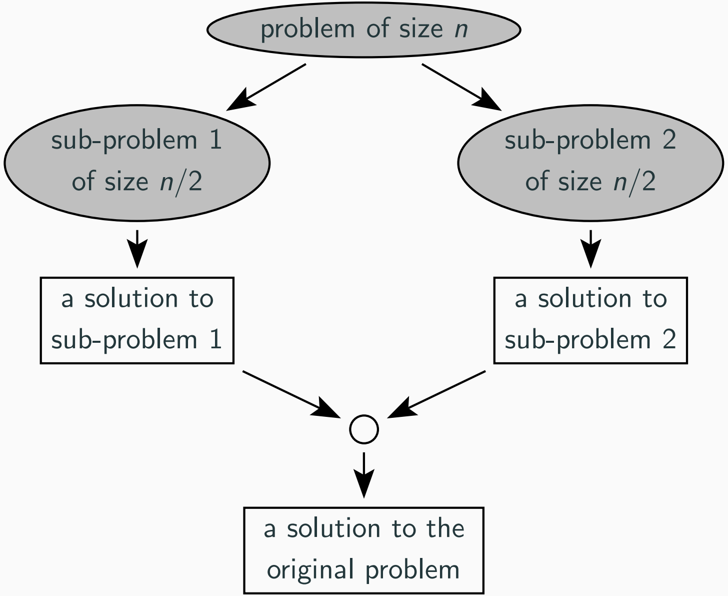
We earlier studied recursion as a powerful problem solving technique.

The divide-and-conquer strategy tries to make the most of this:

1. Divide the given problem instance into smaller instances.
2. Solve the smaller instances recursively.
3. Combine the smaller solutions to solve the original instance.

This works best when the smaller instances can be made to be of equal size.

Split-Solve-and-Join Approach



Divide-and-Conquer Algorithms

You have seen:

- Tree traversal
- Closest pair

You will learn later:

- Mergesort
- Quicksort

Divide-and-Conquer Recurrences

What is the time required to solve a problem of size n by divide-and-conquer?

For the general case, assume we split the problem into b instances (each of size n/b), of which a need to be solved:

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

where $f(n)$ expresses the time spent on dividing a problem into b sub-problems and combining the a results.

(A very common case is $T(n) = 2T(n/2) + n$.)

How do we find closed forms for these recurrences?

The Master Theorem

(A proof is in Levitin's Appendix B.)

For integer constants $a \geq 1$ and $b > 1$, and function f with $f(n) \in \Theta(n^d)$, $d \geq 0$, the recurrence

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

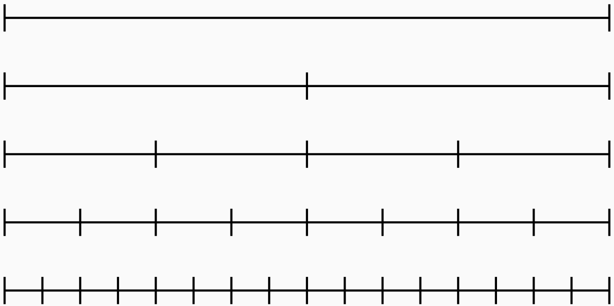
(with $T(1) = c$) has solutions, and

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

Note that we also allow a to be greater than b.

Master Theorem: Example 1

$$T(n) = 2T(n/2) + n \qquad\qquad a = 2, b = 2, d = 1$$



$1 \times n$

$2 \times n/2$

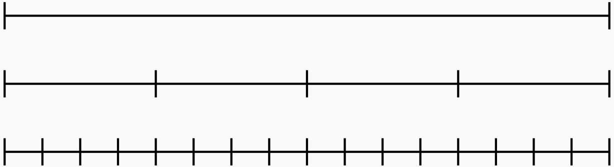
$4 \times n/4$

$\vdots$

$(\log_2 n \text{ times})$

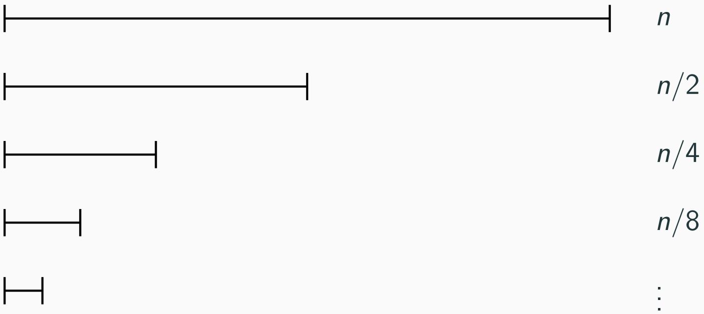
Master Theorem: Example 2

$$T(n) = 4T(n/4) + n \qquad\qquad a = 4, b = 4, d = 1$$



Master Theorem: Example 3

$$T(n) = T(n/2) + n \qquad a = 1, b = 2, d = 1$$



Master Theorem: Example 4

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

$$a = 2, b = 2, d = 2$$

Here $a < b^d$ and we simply get n^d.