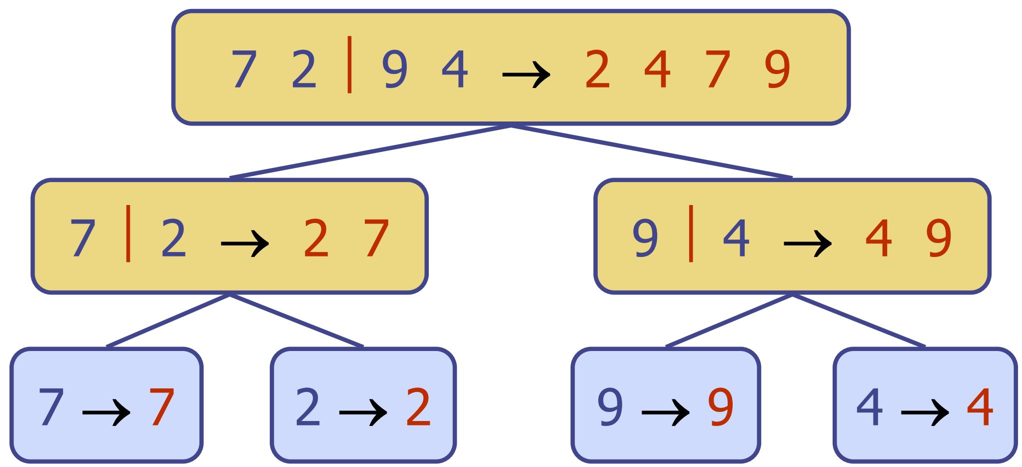
Recurrences



Outline

- Recurrence Equations
- Solving Recurrence Equations
  - Recursion trees
  - Iterative substitution

Recurrence Equation Analysis

- The conquer step of MergeSort consists of merging two sorted sequences, each with $n/2$ elements and takes $O(n)$ steps
- The basis case ($n < 2$) will take $O(1)$ steps, i.e. constant time.
- If we let $T(n)$ denote the running time of MergeSort on n items:

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

- We analyze the running time of MergeSort by finding a **closed form solution** to the above equation, i.e. a solution that has $T(n)$ only on the left-hand side.

Recursion Tree Method

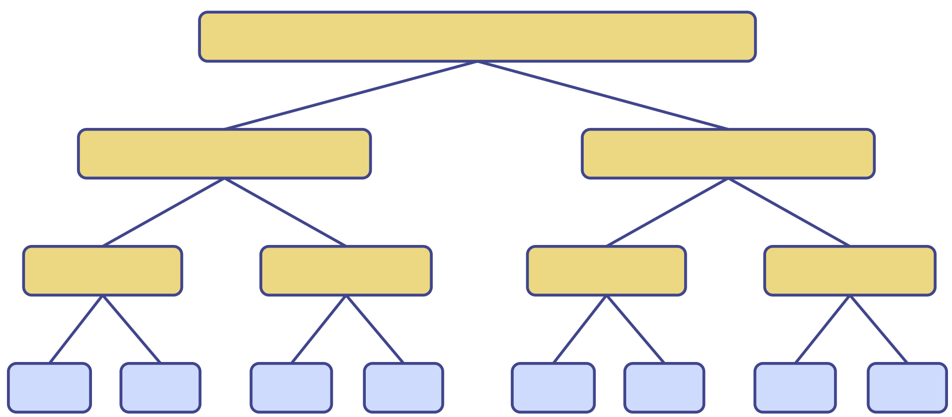
- Draw the recursion tree for the recurrence relation and look for a pattern and then try to prove it is true by induction:

Recursion Tree Method

- Draw the recursion tree for the recurrence relation and look for a pattern and then try to prove it is true by induction:

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

i	T's	size	time
0	1	n	n
1	2	$n/2$	n
…	…	…	n
i	2^i	$n/2^i$	…



Total time = $n \lg n$

Iterative Substitution Method

- In the iterative substitution technique, we iteratively apply the recurrence equation to itself and see if we can find a pattern.

Iterative Substitution Method

- In the iterative substitution technique, we iteratively apply the recurrence equation to itself and see if we can find a pattern.

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2(2T(n/4) + n/2) + n \\
&= 4T(n/4) + 2n \\
&= 8T(n/8) + 3n \\
&= 2^4 T(n/2^4) + 4n \\
&= \ldots \\
&= 2^i T(n/2^i) + in
\end{aligned}
$$

- We reach the end of the recursion when $2^i = n$. That is, $i = \lg n$.
- So, $T(n) = n + n \lg n$
- Thus, T(n) is O(n lg n). (Θ(n lg n), if we can argue that the algorithm behaves the same no matter what the input looks like, which, we can.)

Should Prove by Induction

- Parallels the recursion process
- We won’t do that. ☹

Master Method

- Many divide-and-conquer recurrence equations have the form:

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \ge d \end{cases}$$

- The Master Theorem:

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}$ $)$, then $T(n)$ is $\Theta(n^{\log_b a}\log_2 \quad n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$,

   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.