6.1 Guessing+Induction

First we need to get rid of the $\mathcal{O}$-notation in our recurrence:

$$T(n) \le \begin{cases} 2T(\lceil \frac{n}{2} \rceil) + cn & n \ge 2 \\ 0 & \text{otherwise} \end{cases}$$

Assume that instead we had

$$T(n) \le \begin{cases} 2T(\frac{n}{2}) + cn & n \ge 2 \\ 0 & \text{otherwise} \end{cases}$$

One way of solving such a recurrence is to guess a solution, and check that it is correct by plugging it in.

6.1 Guessing+Induction

Suppose we guess $T(n) \le dn \log n$ for a constant d. Then

$$\begin{aligned} T(n) &\le 2T\Big(\frac{n}{2}\Big) + cn \\ &\le 2\Big(d\frac{n}{2}\log\frac{n}{2}\Big) + cn \\ &= dn(\log n - 1) + cn \\ &= dn\log n + (c-d)n \\ &\le dn\log n \end{aligned}$$

if we choose $d \ge c$.

Formally one would make an induction proof, where the above is the induction step. The base case is usually trivial.

6.1 Guessing+Induction

$$T(n) \le \begin{cases} 2T(\frac{n}{2}) + cn & n \ge 16 \\ b & \text{otw.} \end{cases}$$

Guess: $T(n) \le dn \log n$.
Proof. (by induction)

- **base case** $(2 \le n < 16)$: true if we choose $d \ge b$.
- **induction step** $2 \ldots n-1 \to n$:

  Suppose statem. is true for $n' \in \{2, \ldots, n-1\}$, and $n \ge 16$. We prove it for n:

$$\begin{aligned} T(n) &\le 2T\Big(\frac{n}{2}\Big) + cn \\ &\le 2\Big(d\frac{n}{2}\log\frac{n}{2}\Big) + cn \\ &= dn(\log n - 1) + cn \\ &= dn\log n + (c-d)n \\ &\le dn\log n \end{aligned}$$

- Note that this proves the statement for $n \in \mathbb{N}_{\ge 2}$, as the statement is wrong for $n = 1$.
- The base case is usually omitted, as it is the same for different recurrences.

Hence, statement is true if we choose $d \ge c$.

6.1 Guessing+Induction

Why did we change the recurrence by getting rid of the ceiling?

If we do not do this we instead consider the following recurrence:

$$T(n) \le \begin{cases} 2T(\lceil \frac{n}{2} \rceil) + cn & n \ge 16 \\ b & \text{otherwise} \end{cases}$$

Note that we can do this as for constant-sized inputs the running time is always some constant (b in the above case).

6.1 Guessing+Induction

We also make a guess of $T(n) \le dn\log n$ and get

$$
\begin{aligned}
T(n) &\le 2T\left(\left\lceil \frac{n}{2} \right\rceil\right) + cn \\
&\le 2\left(d\left\lceil \frac{n}{2} \right\rceil \log\left\lceil \frac{n}{2} \right\rceil\right) + cn \\
\boxed{\left\lceil \tfrac{n}{2} \right\rceil \le \tfrac{n}{2}+1} \quad &\le 2(d(n/2+1)\log(n/2+1)) + cn \\
\boxed{\tfrac{n}{2}+1 \le \tfrac{9}{16}n} \quad &\le dn\log\left(\frac{9}{16}n\right) + 2d\log n + cn \\
\boxed{\log \tfrac{9}{16}n = \log n + (\log 9 - 4)} \quad &= dn\log n + (\log 9 - 4)dn + 2d\log n + cn \\
\boxed{\log n \le \tfrac{n}{4}} \quad &\le dn\log n + (\log 9 - 3.5)dn + cn \\
&\le dn\log n - 0.33dn + cn \\
&\le dn\log n
\end{aligned}
$$

for a suitable choice of d.

6.2 Master Theorem

Note that the cases do not cover all possibilities.

Lemma 1

Let $a \ge 1, b \ge 1$ and $\epsilon > 0$ denote constants. Consider the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)\ .$$

Case 1.

If $f(n) = \mathcal{O}(n^{\log_b(a)-\epsilon})$ then $T(n) = \Theta(n^{\log_b a})$.

Case 2.

If $f(n) = \Theta(n^{\log_b(a)}\log^k n)$ then $T(n) = \Theta(n^{\log_b a}\log^{k+1} n)$, $k \ge 0$.

Case 3.

If $f(n) = \Omega(n^{\log_b(a)+\epsilon})$ and for sufficiently large n $af(\frac{n}{b}) \le cf(n)$ for some constant $c < 1$ then $T(n) = \Theta(f(n))$.

6.2 Master Theorem

We prove the Master Theorem for the case that n is of the form b^ℓ, and we assume that the non-recursive case occurs for problem size 1 and incurs cost 1.

The Recursion Tree

The running time of a recursive algorithm can be visualized by a recursion tree: