Tutorial 2

1 Solving recurrences

Solve the following recurrences to obtain a closed-form big-Θ expression for $T(n)$. In each recurrence, you can assume that $T(1) = 1$. And you may assume that n is a power of 2 if that assumption is helpful.

(a) $T(n) = T(n-1) + 1$.

Solution. $T(n) = \Theta(n)$.

Proof. The recursion tree for this recurrence is a simple line tree:

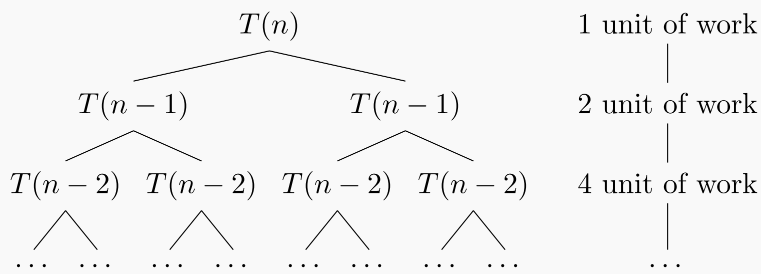
$$
\begin{array}{cc}
T(n) & \text{1 unit of work} \\
| & | \\
T(n-1) & \text{1 unit of work} \\
| & | \\
T(n-2) & \text{1 unit of work} \\
| & | \\
\cdots & \cdots \\
| & | \\
T(1) & \text{1 unit of work}
\end{array}
$$

The tree has depth n and performs 1 unit of work at each level, for a total amount of work $T(n) = n$. □

(b) $T(n) = 2\,T(n-1) + 1$.

Solution. $T(n) = \Theta(2^n)$.

Proof. The recursion tree for this recurrence is now a binary tree:



The tree has depth n and performs a total of

$$1 + 2 + 4 + 8 + \cdots + 2^n = 2^{n+1} - 1 = \Theta(2^n)$$

units of work. $\square$

(c) $T(n) = T(\frac{n}{2}) + 1$.

Solution. $T(n) = \Theta(\log n)$.

Proof. The solution of this problem can be obtained with the Master Theorem, with the parameters $a = 1$, $b = 2$, $c = 0$. Then $\frac{a}{b^c} = \frac{1}{2^0} = 1$, so$T(n) = \log_b(n) = \log_2(n) = \Theta(\log n)$. We can also solve this problem using the recursion tree. As with the first problem, the recursion tree for this recurrence is a simple line tree:

$$
\begin{array}{cc}
T(n) & \text{1 unit of work} \\
| & | \\
T(\frac{n}{2}) & \text{1 unit of work} \\
| & | \\
T(\frac{n}{4}) & \text{1 unit of work} \\
| & | \\
\cdots & \cdots \\
| & | \\
T(1) & \text{1 unit of work}
\end{array}
$$

The tree again performs 1 unit of work at each level, but now the depth of the tree is $\log_2(n)$, so $T(n) = \log_2(n) \cdot 1 = \Theta(\log n)$. $\square$