---

1. Using the Big-O and Big-Omega definitions, show that the function $f(n) = n^3 + n$ is $\mathrm{O}(n^3)$ and $\Omega(n)$.

   Solution:

   To show that $n^3 + n = \mathrm{O}(n^3)$, we need to give two constants c and n_0 such that $n^3 + n \leq cn^3$ for all $n \geq n_0$.

   $n^3 + n \leq cn^3 \implies (c-1)n^3 - n \geq 0 \implies (c-1)n^2 - 1 \geq 0$. Let us choose $c = 2$. $n^2 - 1 \geq 0$ for all positive integers n. So we can choose $n_0 = 1$.

   To show that $n^3 + n = \Omega(n)$, we need to give two constants c and n_0 such that $n^3 + n \geq cn$ for all $n \geq n_0$.

   $n^3 + n \geq cn \implies n^3 + (1-c)n \geq 0 \implies n^2 + (1-c) \geq 0$. Let us choose $c = 0.5$. $n^2 + 0.5 > 0$ for all positive integers n. So we can choose $n_0 = 1$.

   Grading: 5 points for Big-O + 5 points for Big-Omega
   2 points for Big-O/Big-Omega definitions, 2 points for giving valid c and n_0, 1 point for other work in support of the solution.

2. Given the terms in the Fibonacci sequence, find a constant $c < 1$ such that $F_n \leq 2^{cn}$ for all $n \geq 0$. Show using induction that your answer is correct.

Solution:

We are asked to find a value of c that is less than 1. Observe that $c = 1$ satisfies the inequality (we can also prove it using induction). Let us choose $c = 1 - \epsilon$, where ϵ is a small value tending to 0, such as 10^{-5}. We then have that $2^c \approx 2$.

We need to show using induction that $F_n \leq 2^{cn}$ for all $n \geq 0$ and for $c = 1 - \epsilon$.

Base case: If $n = 0$, $F_n = 0 \leq 2^0 = 1$. If $n = 1$, $F_n = 1 \leq 2^c$, since 2^c is greater than 1 for all $c > 0$.

Inductive step: Let us assume that $F_{n-1} \leq 2^{c(n-1)}$ and $F_{n-2} \leq 2^{c(n-2)}$. We now wish to show that $F_n \leq 2^{cn}$.

$F_n = F_{n-1} + F_{n-2} \leq 2^{c(n-1)} + 2^{c(n-2)} = 2^{cn}(2^{-c} + 2^{-2c})$. We are done if we can show that $2^{-c} + 2^{-2c} < 1$ for the c we have chosen. When $c = 1$, $2^{-c} + 2^{-2c} = \frac{1}{2} + \frac{1}{4} = 0.75 < 1$. If $c = 1 - \epsilon$ and $\epsilon = 10^{-5}$, then $2^{-c} + 2^{-2c} > 0.75$, but certainly less than 1.

Optional analysis: For what values of c will the above result hold? If $c = 0.5$ (which means $\epsilon = 0.5$), then $2^{-c} + 2^{-2c} = \frac{1}{\sqrt{2}} + \frac{1}{2} > 1$. So the proof breaks down. The crossover point is somewhere between 0.5 and 1. Observe that $2^{-c} + 2^{-2c}$ is a monotonically decreasing function for $c \in (0.5, 1)$. Thus, to find the crossover point, or the smallest c for which this condition holds, we just set $2^{-c} + 2^{-2c}$ to 1 and solve for c. Let us denote the crossover value of c to be c_m. Then, $2^{-c_m} + 2^{-2c_m} = 1$. Let $x = 2^{-c_m}$. Then, $x + x^2 = 1$, or $x^2 + x - 1 = 0$, or $x = \frac{\sqrt{5}-1}{2}$ (the positive root). $2^{-c_m} = \frac{\sqrt{5}-1}{2} \implies 2^{c_m} = \frac{2}{\sqrt{5}-1} \implies c_m = 1 - \log_2(\sqrt{5} - 1) \approx 0.694$. So, ϵ can be as large as $\log_2(\sqrt{5} - 1)$ for the inequality to hold.

Grading: 3 points for providing a value of c (either at the start or after some analysis), 3 points for showing base case, 4 points for inductive step with correct inductive hypothesis.

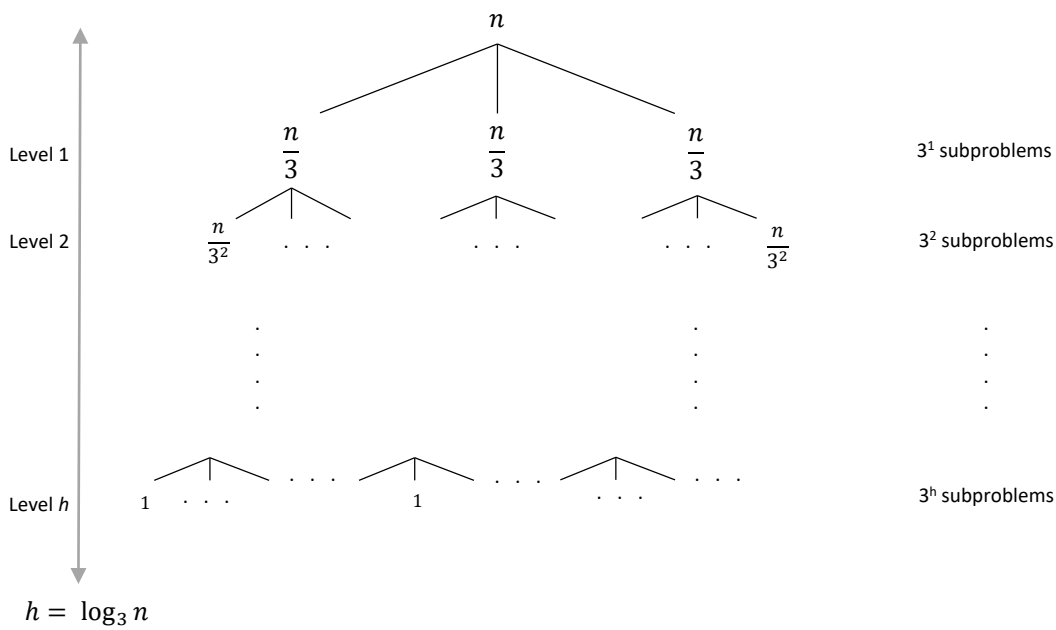
3. Solve the recurrence $T(n) = 3T(n/3) + n$ using the recursion tree method. Express your answer in the Θ-notation. In the recursion tree method, you

- draw a recursion tree,
- determine its height,
- estimate the work associated with nodes at each level, and
- simplify the summation to come up with a closed-form expression for the running time.

Assume that $T(1) = \Theta(1)$ and that n is a power of 3.

Solution:

The recursion tree looks like this:



The work associated with a node at level i is $\frac{n}{3^i}$.

The running time $T(n) = \sum_{i=0}^{h} \frac{n}{3^i} \cdot 3^i = n \sum_{i=0}^{h} 1 = n(h+1) = n(\log_3 n + 1) = \Theta(n \log n)$.

Grading: 4 points for recursion tree (2 points for correct work expressions at each node, 1 point for indicating number of subproblems, 1 point for correct branching), 2 points for height (1 point deducted if base 3 is missing), 4 points for final answer (2 points for correct summation, 2 points for simplification).

4. Solve the recurrence $T(n) = T(n-1) + \log n$ using the iterative substitution method.
Solution:
$T(n) = T(n-1)+\log n = T(n-2)+\log(n-1)+\log n = T(n-3)+\log(n-2)+\log(n-1)+\log n$.

So we have $T(n) = T(n-3) + \log(n(n-1)(n-2))$.

The general form appears to be $T(n) = T(n-k) + \log(n(n-1)\cdots(n-(k-1))$ for an integer k.

Plugging in $k = n-1$ and $T(1) = c$ for some $c > 0$, we have

$T(n) = T(1) + \log n! = c + \log n! = \Theta(\log n!)$.

Grading: 4 points for iterative substitution steps, 3 points for general form, and 3 points for steps towards final solution in Big-Theta notation.