Practical Questions

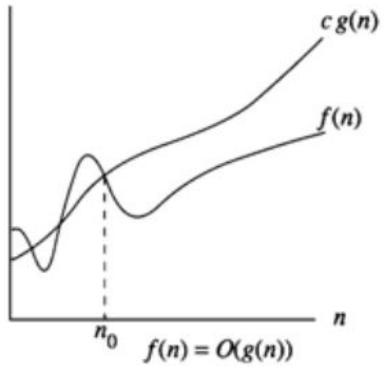
CSCA48– Week 11

We define Big-Oh as the following.

"Big Oh." Let $g \in \mathcal{F}$. $\mathcal{O}(g)$ is the *set* of functions $f \in \mathcal{F}$ such that

$$\exists c \in \mathbb{R}^+, \exists b \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq b \rightarrow f(n) \leq c \cdot g(n)$$

Pictorally*:



Where n0 is basically b.

Example. Let $f(n) = n^3 - n^2 + 5$.

Q. What should $g(n)$ be such that $f(n) \in \mathcal{O}(g(n))$?

Proof.

$$\begin{aligned} n^3 - n^2 + 5 &\leq \\ &\leq \\ &= \end{aligned}$$

Q. What is b?

Q. What is c?

Determine whether the following statements are True or False.

a) There exists a sufficiently large k such that x is O((log x)^k).

b) log(x^5) is O(log(x)).

c) If *f1* is O(g1) and *f*2 is O(*g2*), then *f*1*f*2 is O(*g1g2*).

d) If *f1* is O(*g1*) and *f2* is O(*g2*), then *f1 + f2* is O(*g1 + g2*).

e) Let *f*(n) be the n-th Fibonacci number. Then *f*(n) is O(3^n).

Prove the following

f) Prove that running time $T(n) = n^3 + 20n + 1$ is $O(n^3)$

g) Prove that running time $T(n) = n^3 + 20n + 1$ is not $O(n^2)$

h) Prove that running time $T(n) = n^3 + 20n + 1$ is $O(n^4)$

Inequalities that can be stated without proof

- $c \leq log(n) \leq n \leq n^k \leq k^n$ for any constants c, k where $k > 1$

i) $Prove\ 3 \cdot sin^2(\frac{n}{\pi})\ is\ O(log(n))$

j) $Prove\ n^{-2}\ is\ not\ O(2^{-n})$

k) $Prove\ (n^2 - 1)^{\frac{1}{2}}\ is\ O(3n)$

1. **(Challenge)**

   (a) Prove or disprove that $\forall n \in \mathbb{N}, n! \in \mathcal{O}(2^n)$

   (b) Prove or disprove that $\forall n \in \mathbb{N}, 2^n \in \mathcal{O}(n!)$.

2. **(Challenge)**

(a) Give pseudocode for an algorithm that takes an array A of n natural numbers and outputs them in non-decreasing sorted order using the priority queue ADT.

(b) What is the best worst-case running time (in $\mathcal{O}$-notation) you can achieve for your algorithm in (a) if you implement the priority queue using a heap? Explain.

(c) Give pseudocode for an algorithm that takes an array A of n natural numbers and outputs the median number using the priority queue ADT.

(d) What is the best worst-case running time (in $\mathcal{O}$-notation) you can achieve for your algorithm in (c) if you implement the priority queue using a heap? Explain.

3. **(Challenge)**

A *priority tree* is a binary tree where, whenever v is a child of u, $priority(u) \geq priority(v)$. That is, it is a heap which is not necessarily a complete tree.

(a) Let H_1 and H_2 be two heaps represented as trees. Describe an efficient way to produce a priority tree that contains all the elements of H_1 and H_2. The operation should take time $\mathcal{O}(\log(|H_1| + |H_2|))$, where $|H|$ means the number of elements in heap H.

(b) Now describe a way to combine k heaps $H_1, \ldots, H_k$ into a priority tree. What is the worst-case running time of this operation?