Natalio Panzarini
CS 146 - Nada Attar
9/23/19

Problem 1. (True or False): No comparison-based sorting algorithm can do better than Ω(n log n) in the worst-case

True

Problem 2. We can extend our notion to the case of two parameters n and m that can go to infinity independently at different rates.
For a given function g(n, m), we donate by O(g(n, m)) the set of functions:

> O(g(n,m)) = { f(n, m) : there exist positive constants c, n0, and m0, such that 0 <= f(n, m) <= c g(n, m) for all n >= n0 or m >= m0}

Give corresponding definitions for Ω(g(n, m)) and Θ(g(n, m)):

> ***Ω***(g(n,m)) = {f(n,m) : there exists positive constants c, n0, m0, such that
> f(n,m) >= c * g(n,m) >= 0 for all n > n0 and m > m0}
>
> Θ(g(n,m)) = {f(n,m) : there exists positive constants c, n0, m0, such that
> f(n,m) is O(g(n,m)) and f(n,m) is ***Ω***(g(n,m)) for all n > n0 and m > m0}

Problem 3. Consider functions f(n) and g(n) as given below. Use the most precise asymptotic notation to show how function f is related to function g in each case (i.e. , f ∈ ?(g)). For example, if you were given the pair of functions f (n) = n and g(n) = n2 then the correct answer would be: f ∈ o(g). To avoid any ambiguity between O(g) and o(g) notations due to writing, use Big-O(g) instead of O(g).

f(n)	**g(n)**	**Relation**
n^3 + 2n + 1	(1/100)n^3 + nlogn	f ∈ Θ(g)
2^n	n^1000	f ∈ ***Ω***(g)
log n (base 2)	log n (base 3)	f ∈ Θ(g)
2^n	3^n	f ∈ Θ(g)
.5^n	1	f ∈ O(g)

Problem 4. Indicate, for each pair of expressions (A, B) in the table below, whether A is O, Ω, Θ, o, ω of B. Assume that k>= 1, ϵ > 0 and c >1 are constant. Your answer should be in the form of the table with "yes" or "no" written in each box.

	A	B	O	o	Ω	ω	Θ
a.	$lg^k n$	n^ϵ					
b.	n^k	c^n					
c.	$\sqrt{n}$	$n^{\sin n}$					
d.	2^n	$2^{n/2}$					
e.	$n^{\lg c}$	$c^{\lg n}$					
f.	$\lg(n!)$	$\lg(n^n)$					

a) No, no, yes, yes, no
b) No, no, yes, yes, no
c) Yes, yes, no, no, no
d) Yes, yes, yes, yes, yes
e) Yes, yes, yes, yes, yes
f) No, no, yes, yes, no

Problem 5. In each of the pairs of the functions below, verify using the definition of "big Oh", whether f(n) = O(g(n)), g(n) = O(f(n)) or both.

a) f(n) = 20n2 + 14n + 7, g(n) = 50n -> g(n) = O(f(n))
b) f(n) = n^2, g(n) = 2^n -> f(n) = O(g(n))