HW2

Shane Drafahl

30 August,2017

1. Prove or disprove the following

(a) $5n^2 - 2n + 26 \in O(n^2)$

We will prove this with the def of big oh. The def of $O(n^2)$ is there exists positive constants c and n_0 such that $0 \leq$ f(n) $\leq$ c * g(n) for all $n_0 \leq n$. In this case f(n) = $5n^2 - 2n + 26$ and g(n) = n^2. We can divide both sides by n^2 and we can go from $0 \leq 5n^2 - 2n + 26 \leq c * n^2$ to $0 \leq \frac{24}{n} + 5 \leq c$. $\frac{24}{n=1} + 5$ = 29 and if $f_1(n) = \frac{24}{n} + 5$ then $f_1(n+1) \leq f_1(n)$ because as natural number n increases it increases the denominator. C could be 29 or greater and $0 \leq 5n^2 - 2n + 26 \leq c * n^2$ would be true so therefore $5n^2 - 2n + 26 \in O(n^2)$ because the property is true.

(b) $\forall_a \geq 1 : a^n \in O(n!)$

We will prove this with def of big oh. So the given statement is equivalent to $0 \leq a^n \leq c * n!$. Using induction we can prove it.

Basis: Starting at n = 1 because the performance of an algorithm with n = 0 is irrelevent. 0 $leqa \leq c$ is true because for any a c can be a constant of c = (a + 1). Inductive Hypothesis: Suppoose $0 \leq a^n \leq c * n!$ is true.

Inductive Step: We need to prove $0 \leq a^{n+1} \leq c * n! * (n+1)$. a^{n+1} increases by some a multiplied by $a * a^n$ from the IH. While the right side $c * n! * (n+1)$ from the IH is multiplied by (n + 1) for $c * n! * (n+1)$. In this case of a, n increases meaning at some point it will increase by more when a < n. So the right side is increasing at a faster rate then the left side of the comparison. This means that there is a point where $n! \leq a^n$ for some a for a given range of n. We can just say $c = a^n = n!$ for the n where they equal. So for $0 \leq a^{n+1} \leq c * n! * (n+1)$ if $a^n > n!$ then the constant c multiplied by $n! * c$ will be greater than or equal to a^n because c is equal to the the value at which n! overtakes a^n so if for n! n is beyond the point where n! overtakes than it will already overtake and be a greater value. The other case is that $a^n \leq n!$ for some n then it wont matter what c is because n! will be increasing at a greater rate. So $0 \leq a^{n+1} \leq c * n! * (n+1)$. a^{n+1} is true and therefore $\forall_a \geq 1 : a^n \in O(n!)$ is true.

(c) $\forall_a \geq : 2^{n+a} \in O(2^n)$

To prove for big oh We must prove for $0 \leq 2^{n+a} \leq c * 2^n$. We can reduce this to $0 \leq 2^n 2^a \leq c * 2^n$. Because for whatever a is we can say that c = 2^a for this value of a so $0 \leq 2^n 2^a \leq 2^a 2^n$. So obviously this is true You cannot produce a negative number from the exponents either. So therefore $\forall_a \geq : 2^{n+a} \in O(2^n)$ is true for all a and n.

(d) $\forall_a > 1 : (f(n) \in O(log_2 n)) => (f(n) \in O(log_a n))$

To prove that this is wrong I will use proof of contradiction. Assuming that $\forall_a > 1 : (f(n) \in O(log_2 n)) => (f(n) \in O(log_a n))$ Then either $f(n) \leq log_a(n) \leq log_2(n)$ or $f(n) \leq log_2(n) \leq log_a(n)$. For $f(n) \leq log_2(n) \leq log_a(n)$ a could equal a number greater than 2 and that would be a contradiction $log_2(n) \leq log_3(n)$ if n is greater than 1. Otherwise $f(n) \leq log_1(n) \leq log_2(n)$. For this case if f(n) = $log_3(n)$ then for $log_a(n)$, a would have to be between 3 and 2 and a contradiction because it cannot be all values. So therefore $\forall_a > 1 : (f(n) \in O(log_2 n)) => (f(n) \in O(log_a n))$ is not true.

(e) $2^n \in O(n^{log^2 n})$

To disprove this we will use proof of contradiction we will assume that $2^n \in O(n^{log^2 n})$ is true so $0 \leq 2^n \leq c * n^{log(n)^2}$. If n = 5 then $2^5 = 32$, and $5^{log(5)^2} = 2.19527...$ So C would need to be 14.5767 or greater for this case. Since we assume it is true we also assume that there is some C for that multiplied by $n^{log(n)^2}$ that would be greater than 2^n for all values of n. We will assume that $0 \leq 2^n \leq c_1 * n^{log(n)^2}$ is true where c_1 would not need to be increased because its true for all values of n. If we increase n by 1 then $2^{n+1} = 2^n * 2$ would increase 2 multiplied by 2^n. On the right side $(n+1)^{log(n+1)^2}$. For an increase of 1 for log(n) n would have to be increase by log(n * 10) so log(n + 1) would increase less than 1 from log(n). So $(n+1)^{log(n+1)^2}$ would not double in size meaning that C would have to increase but by contradiction because we assumed that C was the greatest value it needed to be $2^n \in O(n^{log^2 n})$ is not true.

(f) $2^{2^{n+1}} \in O(2^{2^n})$

To disprove this $0 \leq 2^{2^{n+1}} \leq 2^{2^n} * C$ we will use proof by contradiction and assume this is true and that C is as larg as it needs to be. If we increase n by 1 we get $0 \leq 2^{2^n * 4} \leq 2^{2^n * 2} * C$. $2^{2^n * 4}$ increases by more than $2^{2^n * 2} * C$ so C would need to be increased but this is a contradiction so therefore by proof of contradiction $2^{2^{n+1}} \in O(2^{2^n})$ is not true.

2. With respect to the input n, what is the worst-case time complexity of the following algorithm?

First we will find in terms of n how many times the outer loop will go. Its going to start with 1 and go to n^2 so we know the outer loop is n^2 time complexity. The inner loop Takes the value of j and takes the ceil of it. We know that every time it id dividing by 2 untill it reaches 1. We know that then $\frac{j}{2^y}$ where y is the total number of times the iner loop loops. This is also aproximate to 2^y = j or $log_2(j)$ = y. So we know that the loop will go for $n^2 * log_2(j)$. We also know that j will reach n so the worst-case time complexity is $O(n^2 log_2(n))$.