Advanced Algorithm HW1

Seungho Jang

August 2020

1 Q1

O(1) < O(lg(n)) = O(klg(n)) < O(n) = O(2n) = O(kn) = O(100000*n) < O(n*lgn) < O(n^2) < O($n^1$00000) < O(n!)

Explanation :

- If limit test were done for lg(n) and klg(n), the result will be constant as k, that means that the ranking is almost same.
- If limit test were done for n, 2n, kn, and 100000n, the result will be constant, that means that the ranking is almost same.
- However, if the limit test were done to n^{100000} and n^2, the result will be n^{9999}. That means that n^{100000} is fast growing.

2 Q2

"The running time of Algorithm A is at least O(n^2) is meaningless". The reason why this statement is meaningless is because of the following :

If we were to set up this, T(n) $\geq$ O(n^2), which refers T(n) $\geq$ f(n) meaning that T(n) or cg(n) is upper-bound of f(n). Because of f(n) can be any functions that is smaller than n^2 such as n or constant, this lead to conclusion of the running time of Algorithm A could be at least non-negative and constant. Therefore, this statement does not deliver or tell us anything about the running time of algorithm A.

3 Q3

1. To prove that $2^{n+1} = O(2^n)$, we have to find constant(c) and $n_0 > 0$ such that $0 \leq 2^{n+1} \leq c * 2^n$, for all $n \geq n_0$. Since $2^{n+1} = 2 * 2^n$, we can conclude this statement is satisfied when c = 2 and $n_0 = 1$
2. To prove that $2^{2n} = O(2^n)$, we have to find constant(c) and $n_0 > 0$ such that $0 \leq 2^{2n} \leq c * 2^n$, for all $n \geq n_0$. Since $2^{2n} == 4^n$, which is greater than 2^n, so this statement is invalid.

4 Q4

Problem : Rank the following functions by order of growth;
$2^{2^{n+1}}$, 2^{2^n}, (n+1)!, n!, e^n, $n*2^n$, 2^n, $n^{lglgn} = lgn^{lgn}$, (lgn)!, $n^2 = 4^{lgn}$, n^2, lg(n!) and n*lgn, n, $\sqrt{2}^{lgn} = \sqrt{n}$, lgn^2, lnn, $\sqrt{lgn}$, 2=$n^{1/lgn}$ and 1

Time Complexity To calculate the time of the execution, this was codded in the link : Algorithm Homework1.ipynb

Graph To plot those functions, the plot code is in the link above.

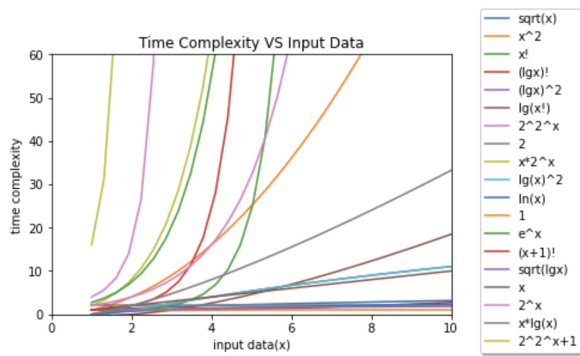

Figure 1: Plot for Time Complexity vs Input data(n)