Practice Problems

CS 412-R1 Algorithms: Design & Analysis

Spring 2023

Properties:

1. if $f(n) = O(g(n))$ and $g(n) = O(h(n))$ then $f(n) = O(h(n))$
   (Transitive property holds for all big-O notations $\{O, o, \theta, \Theta, \Omega\}$)
2. if $f(n) = O(g(n))$ and $h(n)$ is non-negative then $f(n).h(n) = O(g(n).h(n))$
3. $f(n) + g(n) = O(Max(f(n), g(n)))$

Functions in increasing growth-rate (in O sense):

$$1 < \log(n) < \sqrt{n} < n < n\log(n) < n\log^k(n) < n^2 < 2^n < n! < n^n$$

Prove or Disprove:

1. $(n+1)^2 = n^2 + O(n)$
2. $2^{n+1} = O(2^n)$
3. $(n + O(n^{1/2})).(n + O(\log n))^2 = n^3 + O(n^{5/2})$
4. $2^{(1+O(\frac{1}{n}))^2} = 2 + O(\frac{1}{n})$
5. $n^{O(1)} = O(e^n)$
6. $O(e^n) = n^{O(1)}$
7. $n^{\log n} = O((\log n)^n)$
8. if $f(n) = O(g(n))$ then $2^{f(n)} = O(2^{g(n)})$
9. $2^{2n} = O(2^n)$
10. $\sqrt{n}\log(n) = O(n)$
11. $\log(n!) = \Theta(n\log(n))$
12. $n! = o(n^n)$
13. $n! = \omega(2^n)$