CSCI 570 - HW 1

Isaac Delgado
4189182267

1. Arrange the following functions in increasing order of growth rate with g(n) following $f(n)$ in your list if and only if $f(n) = O(g(n))$; $n + 10, 100^n, \sqrt{n}, n^{2.5}, 10^n, n^2 \log n$

My Answer:

$\sqrt{n}, \quad n + 10, \quad n^2 \log n, \quad n^{2.5}, \quad 10^n, \quad 100^n$

2. Arrange the following functions in increasing order of growth rate with g(n) following $f(n)$ in your list if and only if $f(n) = O(g(n))$; $2^{\log n}, 2^n, n(\log n)^3, n^{4/3}, 2^{2^n}, n \log n, 2^{n^2}$

My Answer:

$2^{\log n}, \quad n \log n, \quad n^{4/3}, \quad n(\log n)^3, \quad 2^n, \quad 2^{n^2}, \quad 2^{2^n}$

3. Suppose that $f(n)$ and $g(n)$ are two positive non-decreasing functions such that $f(n) = O(g(n))$. Is it true that $\log f(n) = O(\log g(n))$? Give a proof or counterexample. Having $g(n)=1$ is NOT a counterexample.

$f(n) = O(g(n))$ implies the inequality $f(n) \leq (g(n) * c)$ where c is some constant
Thus we get $\log(f(n)) \leq \log(g(n) * c)$
Reduces to $\log(f(n)) \leq \log(g(n)) + \log(c)$

C is a constant thus we can always find a constant big enough to validate the inequality.
$\log f(n) = O(\log g(n))$ is TRUE.

4. Give a liner time algorithm based on BFS to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. It should not output all cycles in the graph, just one of them. You are NOT allowed to modify BFS, but rather use it as a black box.

Run initial BFS on a given Graph

Save the spanning tree nodes,

While there are nodes to traverse too in initial BFS spanning tree

    Run BFS on an adjacent node to root node,

    If there is a similar node between spanning trees, except the prior adjacent node, then there is a cycle

    return the duplicate node found and the node that BFS was ran on

Isaac Delgado
4189182267

5. Let $G = (V, E)$ be a connected undirected graph and let v be a vertex in G. Let T be the depth-first search tree of G starting from v, and let U be the breadth-first search tree of G starting from v. Prove that the depth of T is at least as great as the depth of U.

Proof By Induction:

Base Case:

We have a graph G(V, E) where V = 2 and E = 1. In this case the inequality holds that $T \geq U$.

Inductive Hypothesis:

$T \geq U$ for G(V, E) where V = m, and E = n

Inductive Step:

Suppose we have a graph G(V, E) which includes vertex *v*. There are two cases involved if we add an additional vertex *x* to the graph that will be the new furthest vertex from *v*.

Case 1: The new furthest vertex *x* added to the graph does NOT create a cycle and thus T = U since both DTS and BFS will have to travel through the additional edge connecting vertex *x* and the vertex prior to *x*.

Case 2: The new furthest vertex *x* added to the graph does create a cycle and U has a chance of being smaller than T but otherwise they will be equal as well.

Additionally, BFS guarantees the shortest path between two nodes, thus no other spanning tree (in this case DTS) can have a shorter distance but may have the same.

Depth of $T \geq U$; TRUE.