University of Washington
Department of Computer Science and Engineering
CSE 421, Winter 2019

January 17, 2019

Homework 2, Due Wednesday, January 23, 1:29 pm, 2019

Turnin instructions: Electronics submission on canvas using the CSE 421 canvas site. Each numbered problem is to be turned in as a separate PDF.

Problem 1 (10 points):
Order the following functions in increasing order by their growth rate:

1. $(\log n)^{\log n}$
2. n^4
3. $2^{\sqrt{\log n}}$
4. $2^{n/10}$

Explain how you determined the ordering.

Problem 2 (10 points):
Prove that $5n^3 + 8n^2 + 11$ is $O(n^3)$.

Problem 3 (10 points):
Suppose that $f(n)$ is $O(s(n))$ and $g(n)$ is $O(s(n))$. Let $h(n) = f(n)g(n)$ and $t(n) = s(n)s(n)$. Prove that $h(n)$ is $O(t(n))$.

Problem 4 (10 points):
Give an algorithm for efficiently computing the *number* of shortest paths in an undirected graph between a a pair of vertices. Suppose that you have an undirected graph $G = (V, E)$ and a pair of vertices v and w. Your algorithm should compute the number of shortest $v - w$ paths in G. Since this graph is unweighted, the length of a path is defined to be the number of edges in the path.
Your algorithm should have run time $O(n + m)$ for a graph of n vertices and m edges.

You should explain why your algorithm is correct and justify the run time of the algorithm.

Problem 5 (10 points):
Let G be an n node undirected graph[1], where n is even. Suppose that every vertex has degree at least $n/2$. Prove that G is connected.

Problem 6 (10 points):
Describe an algorithm to determine if an undirected graph $G = (V, E)$ with n vertices and m edges has a cycle. Your algorithm should run in $O(n)$ time *independent* of how many edges the graph has. For this problem, you can assume an adjacency list representation of the graph. The set of vertices is provided in an array $Vert[n]$, and each vertex has a list of its adjacent edges.

You should explain why your algorithm is correct and justify the run time of the algorithm.

(This is an unusual problem, in that you may not be able to look at all of the edges. You do not need to output the cycle if you find one.)

[1] You can assume, unless specified otherwise, that graphs do not have self loops or parallel edges.