Homework 2

Due: Friday, Sept 12, 11:59 PM PT

1. Arrange these functions under the Big-O notation in increasing order of growth rate, with $g(n)$ following $f(n)$ in your list if and only if $f(n) = O(g(n))$; mention equality (of growth-rate) if $f(n) = O(g(n))$ and also $g(n) = O(f(n))$. Here, $\log(x) = \log_2(x)$, i.e., denotes the logarithm with base 2):

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

(10 points)

2. For each of the following, indicate whether $f(n) \in O(g(n))$, $f(n) \in \Theta(g(n))$, or $f(n) \in \Omega(g(n))$. Here, $\log(x) = \log_2(x)$, i.e., denotes the logarithm with base 2. (10 points)
   (a) $f(n) = n!$ and $g(n) = 2^{2^n}$
   (b) $f(n) = e^n$ and $g(n) = n^{\log n}$
   (c) $f(n) = 2^n$ and $g(n) = 2^{3n}$
   (d) $f(n) = n^2$ and $g(n) = 2^{\sqrt{\log n}}$
   (e) $f(n) = \log n$ and $g(n) = \log(\log(6^{2n}))$

3. We have a connected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a DFS tree rooted at u, and obtain a tree T (remember that a DFS tree includes all nodes of G). Suppose we then compute a BFS tree rooted at u, and obtain the same tree T. Prove by **contradiction** that G is the same as T, that is, G cannot contain any edges that do not belong to T. (10 points)
   Hint: For any edge (x, y) in G, how much can the level of x in the BFS T differ from the level of y in it? Further, what can we say about the locations of x and y relative to each other in the DFS T?

4. Given an unweighted and undirected graph $G = (V, E)$ and an edge $e \in E$. Design an algorithm to determine whether the graph G has a cycle containing that specific edge e. Also, determine the time complexity of your algorithm with explanation. **Note**: To be eligible for full credits on this problem, the running time of your algorithm should be bounded by $O(|V| + |E|)$. (10 points)

Ungraded Problems:

1. Given functions f_1, f_2, g_1, g_2 such that $f_1(n) = \mathcal{O}(g_1(n))$ and $f_2(n) = \mathcal{O}(g_2(n))$. For each of the following statements, decide whether you think it is true or false, and give a proof or counterexample. (12 points)

   (a) $f_1(n) + f_2(n) = \mathcal{O}(\max(g_1(n), g_2(n)))$

   (b) $f_1(n) \cdot f_2(n) = \mathcal{O}(g_1(n) + g_2(n))$

   (c) $f_1(n)^2 = \mathcal{O}(g_1(n)^2)$

   (d) $\log f_1(n) = \mathcal{O}(\log(g_1(n)))$

2. Consider the following prime filtering algorithm that outputs all the prime numbers in $2, \ldots, n$ (the pseudo code is presented in Algorithm 1). (10 points)

Algorithm 1 Prime Filtering

Input: a positive integer $n \geq 2$
initialize the Boolean array $isPrime$ such that $isPrime(i) =$ **True** for $i = 2, \ldots, n$
for $i = 2 \ldots n$ **do**
    for $j = 2 \ldots \lfloor \frac{n}{i} \rfloor$ **do**
        if $i \times j \leq n$ **then**
            $isPrime(i \times j) \leftarrow$ **False**
        end if
    end for
end for

   (a) Please prove this algorithm is correct (that is, a positive integer k that $2 \leq k \leq n$ is a prime if and only if *isPrime*$(k) =$ **True**). (5 points)

   (b) Please calculate the time complexity under the Big-$\mathcal{O}$ notation. (5 points)

3. A directed graph $G = (V, E)$ is **singly connected** if G contains at most one simple path from u to v for all vertices $u, v \in V$. Construct an algorithm using DFS to determine whether or not a directed graph is singly connected. Also, determine the time complexity of your algorithm and justify your answer. (10 points)