CS 513
Solutions to #1 Math Fundamentals

1. Prove: A binary tree with n nodes has depth at least $\lfloor \log n \rfloor$ and at most $n-1$. (Hint: Show that if a binary tree has depth d and has n nodes, then $n \leq 2^{d+1} - 1$.)

   Proof: We first prove the hint by induction on d.

   Base case: $d = 0$, then $n = 1$, and $1 \leq 2^{0+1} - 1 = 1$.

   Inductive Hypothesis: Suppose that for all binary trees with depth d' less than d and with n nodes, $n \leq 2^{d'+1} - 1$. Now consider a binary tree T with depth d, and let T_0 and T_1 be the (possibly empty) subtrees rooted at the children of the root of T. Let their depths be d_0 and d_1, respectively, and let their number of nodes be n_0 and n_1, respectively.

   Now, $d = \max\{d_0, d_1\} + 1$ and $n = n_0 + n_1 + 1$. By induction,

   $$n = n_0 + n_1 + 1 \leq 2^{d_0+1} - 1 + 2^{d_1+1} - 1 + 1 = 2^{d_0+1} + 2^{d_1+1} - 1.$$

   But $2^{d_0+1} + 2^{d_1+1} \leq 2 \cdot 2^d = 2^{d+1}$. Combining these inequalities gives $n \leq 2^{d+1} - 1$. Solving for d gives

   $$d \geq \log(n+1) - 1.$$

   Then,

   $$d \geq \lfloor \log n \rfloor.$$

   The last inequality is trivially true if $\log(n+1) - 1 \geq \lfloor \log n \rfloor$. Otherwise, since $\log n \geq \log(n+1) - 1 \geq \log n - 1$ and also $\log n \geq \lfloor \log n \rfloor > \log n - 1$, then $\log n \geq \lfloor \log n \rfloor > \log(n+1) - 1 \geq \log n - 1$. Therefore, the claim is true because d has to be an integer.

   The upper bound is proved similarly:

   Base case: $d = 0$, then $n = 1$, and $0 \leq 1 - 1 = 0$.

   Inductive Hypothesis: Suppose that for all binary trees with depth d' less than d and with n nodes, $d' \leq n - 1$. Now consider a binary tree T with depth d, and let T_0 and T_1 be the subtrees rooted at the children of the root of T. Let their depths be d_0 and d_1, respectively, and let their number of nodes be n_0 and n_1, respectively.

   Assume, w.l.o.g., that $d_0 \geq d_1$. By induction, we get that

   $$d = d_0 + 1 \leq n_0 - 1 + 1 = n - n_1 - 1 \leq n - 1. \square$$

2. Prove that $\log(n!) \in \Theta(n \log n)$.

   Proof: $n! = 1 \cdot 2 \cdot \cdots n \geq \lceil n/2 \rceil \cdot (\lceil n/2 \rceil + 1) \cdots n \geq n/2 \cdot n/2 \cdots n/2 = (n/2)^{n/2}$. But $\log n/2^{n/2} = n/2 \log n/2 = \frac{1}{2} n \log n - \frac{1}{2} > \frac{1}{2} n \log n$. Therefore, $\log(n!)$ is $\Omega(n \log n)$.

   $n! = 1 \cdot 2 \cdot \cdots n \leq n \cdot n \cdots n = n^n$. Now, $\log n^n = n \log n$. Therefore, $\log(n!)$ is $O(n \log n)$.

   Since $\log(n!)$ is both $\Omega(n \log n)$ and $O(n \log n)$, it is $\Theta(n \log n)$.$\square$

3. Prove by induction on k: (Remember these formulas, they may be used later)

   (a) $\sum_{i=1}^{k} i(i+1) = \frac{1}{3}k(k+1)(k+2)$
   Base case: $k = 1$: $\sum_{i=1}^{1} i(i+1) = 2 = \frac{1}{3}1(1+1)(1+2)$.
   Inductive Hypothesis: Suppose that for all $k' < k$, that $\sum_{i=1}^{k'} i(i+1) = \frac{1}{3}k'(k'+1)(k'+2)$. Then $\sum_{i=1}^{k} i(i+1) = k(k+1)+\sum_{i=1}^{k-1} i(i+1) = k(k+1)+\frac{1}{3}(k-1)(k)(k+1) = \frac{1}{3}(3k(k+1)+(k-1)(k)(k+1)) = \frac{1}{3}(k(k+1)(3+(k-1)) = \frac{1}{3}(k(k+1)(k+2))\square$.

   (b) $\sum_{i=0}^{k} i2^i = (k-1)2^{k+1} + 2$
   Base case: $k = 1$: $\sum_{i=0}^{1} i2^i = 2 = (1-1)2^{1+1} + 2$.
   Inductive Hypothesis: Suppose that for all $k' < k$, that $\sum_{i=0}^{k'} i2^i = (k'-1)2^{k'+1} + 2$. Then $\sum_{i=0}^{k} i2^i = k2^k + \sum_{i=0}^{k-1} i2^i = k2^k + (k-2)2^k + 2 = (2k-2)2^k + 2 = 2(k-1)2^k + 2 = (k-1)2^{k+1} + 2.\square$

   (c) $\sum_{i=0}^{k} \frac{i}{2^i} = 2 - \frac{k+2}{2^k}$
   Base case: $k = 1$: $\sum_{i=0}^{1} \frac{i}{2^i} = 1/2 = 2 - \frac{1+2}{2^1}$.
   Inductive Hypothesis: Suppose that for all $k' < k$, that $\sum_{i=0}^{k'} \frac{i}{2^i} = 2 - \frac{k'+2}{2^{k'}}$. Then $\sum_{i=0}^{k} \frac{i}{2^i} = \frac{k}{2^k} + \sum_{i=0}^{k-1} \frac{i}{2^i} = \frac{k}{2^k} + 2 - \frac{k+1}{2^{k-1}} = 2 - \frac{2k+2-k}{2^k} = 2 - \frac{k+2}{2^k}$. $\square$

4. Place the following functions into increasing asymptotic order. If two or more of the functions are of the same asymptotic order, then indicate this. Prove the correctness of your ordering. (In other words, if you claim that $g(n)$ is greater than $f(n)$ then show that $f(n) \in O(g(n))$ but $f(n)$ is not in $\Theta(g(n))$.

$$4n, \quad n^2, \quad n\log n, \quad n\ln n, \quad \log n, \quad e^n$$

   Note: log means logarithm base 2, and ln means logarithm base e (natural log).

   These functions can be ordered as:

$$\log n < 4n < n\ln n = n\log n < n^2 < e^n.$$

   Proof:

$$\lim_{n\to\infty} \frac{\log n}{4n} = \lim_{n\to\infty} \frac{1}{4n\ln 2} = 0.$$

   So $\log n$ is $O(4n)$ but not $\Theta(4n)$.

$$\lim_{n\to\infty} \frac{4n}{n\ln n} = \lim_{n\to\infty} \frac{4}{\ln n + 1} = 0.$$

   So $4n$ is $O(n\ln n)$ but not $\Theta(n\ln n)$.

$$\lim_{n\to\infty} \frac{n\ln n}{n\log n} = \lim_{n\to\infty} \frac{\ln n}{\log n} = \lim_{n\to\infty} \ln 2 = \ln 2.$$

   So $n\ln n$ is $\Theta(n\log n)$.

$$\lim_{n\to\infty} \frac{n \ln n}{n^2} = \lim_{n\to\infty} \frac{\ln n}{n} = \lim_{n\to\infty} \frac{1}{n} = 0.$$

So $n \ln n$ is $O(n^2)$ but not $\Theta(n^2)$.

$$\lim_{n\to\infty} \frac{n^2}{e^n} = \lim_{n\to\infty} \frac{2n}{e^n} = \lim_{n\to\infty} \frac{2}{e^n} = 0.$$

So n^2 is $O(e^n)$ but not $\Theta(e^n)$.

5. Let $G = (V, E, W)$ be a weighted graph such that no two different edges in E have the same weight ($\forall e_1, e_2 \in E, e_1 \neq e_2 \Rightarrow W(e_1) \neq W(e_2)$). Prove that G has a unique unrooted minimal spanning tree (The set of edges participating in a MST for G is unique).

   Proof: Suppose that G has two distinct MSTs, $T_1 = (V, E_1)$ and $T_2 = (V, E_2)$. Let E_1' be the edges unique to T_1 (i.e. $E_1 - E_2$) and E_2' be the edges unique to E_2. Let m_1 be the maximum weight edge in E_1' (i.e. $m_1 = \max\{w(e)|e \in E_1'\}$), and m_2 be the maximum weight edge in E_2'. Suppose now, without loss of generality, that $m_2 > m_1$ (they obviously can't be equal). Let x and y be such that $w(\overline{xy}) = m_2$. Let $P = (x = v_0), v_1, \ldots, (v_k = y)$ be the path from x to y in T_1.

   Now, there must be an edge $\overline{v_i v_{i+1}}$ of P in E_1', since if every edge of P where in E_2, then T_2 would have a cycle. But $w(\overline{v_i v_{i+1}}) \leq m_1 < m_2$, so we can reduce the weight of T_2 by removing $\overline{xy}$ and adding $\overline{v_i v_{i+1}}$. The lemma from class tells us that this new graph is a tree, so we can conclude that T_2 was not an MST. This contradiction tells us that we can't pick two distinct MSTs T_1 and T_2, so G must have a unique MST.□

6. Suppose there is a C++ library on your machine which has the following function:

   $HamC(G)$**:**

   Input: Graph G

   Output: Yes, if G has a simple cycle which traverses all nodes; No, otherwise.

   Such a cycle is known as a *Hamiltonian Cycle*. Let $HamC(G)$ implement the above routine. Suppose further, that $HamC()$ runs in polynomial time, that it, it runs in time $O(n^c)$ for some constant c on an n node graph.

   A *Hamiltonian Path from u to v* is a simple path from u to v which traverses all the nodes of a graph. Your mission is to write an algorithm with the following input and output:

   $HamP(G, u, v)$**:**

   Input: Graph G, and nodes u and v in G.

   Output: Yes, if there is a Hamiltonian Path from u to v in G; No otherwise.

   Your algorithm should run in polynomial time. (Hint: You'll have a tough time doing this without using the $HamC()$ subroutine.)

Answer: Suppose you want to compute $HamP(G, u, v)$. Add a new node w to G and connect it to u and v. Call the new graph G'. Now compute $HamC(G')$. Now, G has a hamiltonian path from u to v iff G' has a hamiltonian cycle. One direction is easy. If G has a ham. path, then completing the cycle by adding edges $\overline{uw}$ and $\overline{vw}$ gives a hamiltonian cycle in G'. If G' has a hamiltonian cycle, then it must contain the node w, so it must contain the egdes $\overline{uw}$ and $\overline{vw}$. Removing these gives the needed hamiltonian path in G. This establishes the correctness of the algorithm. The running time is simply a constant plus the running time of $HamC(G')$, and so if $HamC$ runs in polynomial time, so does $HamP$.