CSE421: Introduction to Algorithms

Homework 3

Due: October 17, 2018

1. Prove, by induction, that $1^1 + 2^2 + 3^3 + \cdots + n^n = O(n^n)$.

2. Consider the interval scheduling problem in the class. Instead of selecting the first compatible job to finish as in the class, we consider the greedy algorithm that picks the job j that is compatible with all other jobs picked with largest $s(j)$. Prove that this yields an optimal solution or give an example to disprove this algorithm.

3. Given a sequence $d_1, \ldots, d_n$ of integers design a polynomial time algorithm that construct a tree such that the degree of vertex i is d_i. If no such tree exists your algorithm must output "Impossible".

   Hint: Show that for every sequence $d_1, \ldots, d_n$ there exists a tree with this degree sequence if and only if $\sum_i d_i = 2(n-1)$ and for all i, we have $d_i \geq 1$. Also, you may have to argue that if the sum of n integers is less than $2n$ then one of them is at most 1.

4. Given a tree T with n vertices and a set of pairs $(u_1, v_1), \ldots, (u_k, v_k)$, design an $O(n + k \log n)$ time algorithm to find the lowest common ancestor of u_i and v_i in T for all $i = 1, 2, \ldots, k$.

   Hint: Let dfsnum$[v]$ be the sequence number for when it was first discovered by depth-first search. This is the number that we printed next to each discovered vertex in DFS-Tree slides. Let st$[\cdot]$ contains all nodes in the stack where their DFS call is still running. Again see the slide for the example. Show that for any pair u_i, v_i such that u_i is discovered first in the DFS, the lowest common ancestor of u_i, v_i is the largest j such that st$[j] \leq$ dfsnum$[u_i]$ at the time that we call dfs(v_i).

5. **Extra Credit:** Let $[n] = \{1, 2, 3, \cdots, n\}$ and $\mathcal{I}$ be a collection of subsets of $[n]$. We call any set $I \in \mathcal{I}$ is nice.

   We know that $\mathcal{I}$ satisfy two main axioms:

   (a) If $X \subset Y$ and $Y \in \mathcal{I}$, then $X \in \mathcal{I}$. Namely, any subset of a nice set is nice.

   (b) If $X \in \mathcal{I}$, $Y \in \mathcal{I}$ and $|Y| > |X|$, then there exists $i \in Y \backslash X$ such that $X \cup \{i\} \in \mathcal{I}$. Namely, if X is nice and there exists a larger nice set Y, then X can be extended to a larger nice set by adding an element of $Y \backslash X$.

   The collection $\mathcal{I}$ may have exponentially size and is only defined implicitly. However, we assume that we can test if a set I is nice or not in polynomial time.

   Given a cost $c_1, c_2, c_3, \cdots, c_n$, design a greedy polynomial time algorithm to find a nice set X with maximum total cost $c(X) = \sum_{x \in X} c_x$.