15-850: Advanced Algorithms **CMU, Spring 2024**
Exercises with Homework #1

Exercises are for fun and edification, please do not submit. (You may discuss these exercises with others.) The ones below are grouped by topic and their subparts do not necessarily build on one another (for example, you do not need to do (a) to do (b)).

1. Exercises about MSTs.

   (a) *(Linear-time Cleanup)* Show how to implement the "contract" algorithm in $O(m)$ time. This algorithm takes as input a graph $G = (V, E)$ with some edges colored blue, and outputs a new graph $G' = (V', E')$ in which vertices $v_C \in V'$ correspond to blue connected components C in G, there are no self-loops, and there is a (single) edge $e_C = (v_C, v_{C'})$ if there exists some edge between the corresponding components C, C' in G, and the weight of edge $(v_C, v_{C'}) := \min_{\{x,y\}\in E: x\in C, y\in C'} w_{xy}$.

   (b) Boruvka reduces the number of nodes by a constant factor in each round, but what about the number of edges? Show a graph with n nodes and m edges where the number of edges in G_i remains $\Omega(m)$ for $\Omega(\log n)$ rounds, even after cleaning up.

   (c) Show that running the contraction version of Boruvka (where we contract components and clean up after every round) on a planar graph runs in linear time $O(m)$. Hint: a simple planar graph on n vertices has at most $3n - 6$ edges. What happens if you contract an edge in a planar graph?

   (d) Show that running $\log\log n$ rounds of Boruvka, followed by the Fibonacci heaps implementation of Prim's, gives an $O(m \log\log n)$ time algorithm for MSTs.

2. Exercises about arborescences (directed spanning trees)

   (a) In the *rooted* min-cost arborescence problem, we are given a digraph $G = (V, A)$ with edge weights w_e and a root $r \in V$, and want to find the min-cost r-arborescence. In the *unrooted* min-cost arborescence problem, we are given a digraph $G = (V, A)$ with edge weights w_e and want to find the min-cost arborescence in the graph, regardless of which vertex it is rooted at.

   Show the rooted and unrooted problems are equivalent. I.e., given an algorithm for one problem, show how to solve the other. Your reductions should take linear time.

   (b) Give a deterministic *linear-time* algorithm for finding a min-cost r-arborescence in a DAG (if one exists).

   (c) For a digraph G, consider two arc-weight functions w_1 and w_2 such that

   $$w_1(a) \le w_1(a') \qquad \Longleftrightarrow \qquad w_2(a) \le w_2(a')$$

   for all arcs $a, a' \in E$. Give an example to show that the min-cost arborescences w.r.t. w_1 and w_2 may be different. (Compare with HW#0 Exercise 2.)

   (d) Show an instance where a^* is the heaviest arc in G, and there exist r-arborescences that do not use a^*, but the cheapest r-arborescence uses a^*. (This is not possible for MSTs.)

3. **(Centroid.)** Given a tree T on n nodes, prove there exists a *centroid* vertex $v \in V(T)$ whose deletion breaks the tree into components with size at most $n/2$. Give an example to show you cannot replace $n/2$ by a smaller value. (Hint: $n = 4$.)

Now, given a value z, find a set of $O(n/z)$ nodes such that deleting these nodes breaks the tree into pieces each of size in $[z, 2z]$. This is called the *centroid decomposition*.

4. **(Amortization.)** You start with n linked lists, each with one item. Each timestep you are asked to merge two of the current lists (thereby reducing the number of lists by 1). The cost of this merge is the length of the shorter of the two lists. Show that the total cost of merging is at most $n \log_2 n$, no matter what sequence of merges are requested. *(Hint: initially give each item a level of zero. Each time you merge two lists, raise the "level" of the items in the smaller list by one. What is the maximum level of any item?)*

5. **(Las Vegas and Monte Carlo.)** A *Las Vegas* randomized algorithm always produces the correct answer, but its running time $T(n)$ is a random variable. For instance, quicksort when choosing a random pivot. A *Monte Carlo* algorithm has a fixed running time, but doesn't always produce the correct answer. E.g., randomized primality testing algorithms, when given a number N, may only be correct with probability 99/100, say. However, one can move from one to the other:

   (a) Given a Las Vegas algorithm with expected runtime $\mathbf{E}[T(n)] \leq f(n)$, show how to get a Monte Carlo algorithm with (i) worst-case running time at most $4f(n)$ and (ii) probability of success at least 3/4.

   (b) Given (a) MC algorithm A for some problem with runtime $\leq f(n)$ and correctness probablity p, and (b) a "checking" algo C that checks whether a given output of A is a correct solution, give a LV algorithm that runs in expected time $\frac{1}{p}(f(n) + g(n))$.

6. **(Convexity, Here we Come.)** Solve the convexity exercises in this worksheet. Show that the intersection of two convex sets is convex, but their union may not be. Show that the sum of two convex functions is convex.