Homework Out: January 24, 2017
Due Date: February 5, 2017

The HW contains some exercises (fairly simple problems to check you are on board with the concepts; dont submit your solutions), and problems (for which you should submit your solutions, and which will be graded). Some problems have sub-parts that are exercises. For this problem set, its OK to work with others. (Groups of 2, maybe 3 max.) That being said, please think about the problems yourself before talking to others. Please cite all sources you use, and people you work with. Submissions due at beginning of class on the due date. Please email *LaTeXed* solutions to Tao at uthaipon3@gatech.edu.

Exercise 1

Exercises about MSTs.

a. 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.

b. For a graph G, consider two edge-weight functions w_1 and w_2 such that

$$w_1(e) \leq w_1(e') \Leftrightarrow w_2(e) \leq w_2(e')$$

for all edges $e, e' \in E$. Show that T is an MST wrt w_1 iff it is an MST wrt w_2. (In other words, only the sorted order of the edges matters for the MST.)

c. Suppose graph G has integer weights in the range $\{1, ..., W\}$, where $W \geq 2$. Let G_i be the edges of weight at most i, and κ_i be the number of components in G_i. Then show that the MST in G has weight exactly $nW + \sum_{i=i}^{W1} \kappa_i$.

d. 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 $3n6$ edges.

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

Exercise 2

Exercises about arborescences (directed spanning trees).

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

   Show these problems are equivalent. I.e., given an algorithm for one problem, show how to solve the other problem. Your (simple) 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

$$w_1(a) \leq w_1(a') \iff w_2(a) \leq w_2(a')$$

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.

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.)

Problem 1

(Yaos $O(m \log\log n)$-time MST Algorithm). The idea behind this algorithm: in Boruvkas algorithm we scan all the edges in the graph in each pass, and we should avoid this repetition. Assume that G is a connected simple graph, and edge weights are distinct.

a. Suppose for each vertex, the edges adjacent to that vertex are stored in non-decreasing order of weights. Show a slight variant of Boruvkas algorithm that runs in time $O(m + n \log n)$ time.

b. (k-partial sorting) Given a parameter k and a list of N numbers, give an $O(N \log k)$-time algorithm that partitions this list into k groups $g_1, g_2, ..., g_k$ of size at most $\lceil N/k \rceil$ each such that all elements in g_i are smaller than those in g_{i+1} for each i.

c. Adapt your algorithm from the first part above to handle the case where the edges adjacent to each vertex are not completely sorted but only k-partially-sorted. Ideally, your run-time should be $O(\frac{m}{k} \log n + n \log n)$.

d. Use the two parts above (setting $k = \log n$), preceded by some additional rounds of Boruvka, to give an $O(m \log\log n)$-time MST algorithm.

Problem 2

(Fun with Boruvka) In this problem, we will tie up some loose ends from lecture:

a. (Linear-time Cleanup) Show how to implement the shrink algorithm in $O(m)$ time. This algorithm takes as input a graph $G = (V, E)$ with some of the edges colored blue, and outputs a new graph $G' = (V', E')$ in which each vertex $v_C \in V'$ corresponds to a blue connected component C in G, there are no self-loops, and there is a (single) edge $e_C = (v_C, v'_C)$ if there exists an edge between the corresponding components C, C' in G. Moreover, the weight of edge $(v_C, v'_C) = \min_{\substack{\{x,y\} \in E: \\ x \in C \\ y \in C'}} w_{x,y}$.

b. (Tighter Boruvka) Consider the naive implementation of Boruvkas algorithm, which for each node in the current graph picks the cheapest edge incident to the node, and then shrinks the graph (as in the previous part). We saw in Lecture #1 that since we spend $O(m)$ time per round, and there are $\leq \log_2 n$ rounds, the algorithm takes $O(m \log n)$ time. Show a tighter bound of $\min\{O(n^2), O(m(1 + \log \frac{n^2}{m}))\}$.

c. (Boruvka on a tree) Suppose $T = (V, E)$ is a tree on n vertices, and we run Boruvkas algorithm on T. (Let $V_1 = V$ be the original vertices at the beginning of round 1, V_i be the vertices in round i, and say there L rounds so that $|V_{L+1}| = 1$). We build a tree T' as follows: the vertices are the union of all the V_i. There is an edge from $v \in V_i$ to $w \in V_{i+1}$ if the vertex v belongs to a component in round i that is contracted to form $w \in V_{i+1}$; the weight of this edge (v, w) is the min-weight edge out of v in round i. (Note that all vertices in V are now leaves in T' .)

For nodes u, v in a tree T, let $\text{maxwt}_T(u, v)$ be the maximum weight of an edge on the (unique) path between u, v in the tree T. Show that for all $u, v \in V$

$$\text{maxwt}_T(u, v) = \text{maxwt}_{T'}(u, v).$$

This transformation is useful in the MST verification algorithm of Komlos, and others.

Problem 3

(Matroids and the Greedy Algorithm) Given a finite universe U of elements, a collection of subsets $\mathcal{I} \subseteq 2^U$ is subset-closed if $B \in \mathcal{I}$ and $A \subseteq B \subseteq U$ implies $A \in \mathcal{I}$. A set system $(U, \mathcal{I})$ is called an independence system if **(I1)**$\emptyset \in \mathcal{I}$ and **(I2)** $\mathcal{I}$ is subset-closed. (We call a set A independent if $A \in \mathcal{I}$.)
An matroid is an independence system that satisfies the additional property **(I3)** that if $A, B \in \mathcal{I}$ and $|A| < |B|$ there is an element $x \in B\ A$ such that $A \cup \{x\} \in \mathcal{I}$.
Some more notation: a maximal independent set is called a base. A cycle (or circuit) is a minimal dependent set. A cut is a minimal set that intersects every base. The rank of a matroid is the cardinality of any maximal (and hence maximum) independent set in the matroid.

a. (Dont hand in) Show that the following set systems are matroids:

   (a) $\mathcal{I}$ is the collection of all subsets of U with cardinality at most r.

   (b) U is the disjoint union of sets $U_1, U_2, \ldots, U_\ell$ (denoted $U = U_1 \uplus U_2 \uplus \ldots \uplus U_\ell$), and

   $$\mathcal{I} = \{A \subseteq U \mid |A \cap U_i| \leq r_i \forall i \in [\ell]\}$$

   (c) U is the set of edges of graph $G = (V, E)$ and $\mathcal{I}$ contains all acyclic subsets of edges (i.e., forests),

   (d) $U = \mathbb{F}^r$ for some field $\mathbb{F}$, and $\mathcal{I}$ contains every set of vectors in U that are linearly independent over the field $\mathbb{F}$.

   Show that the ranks of these matroids is $r, \sum_i r_i, |V| - 1$, and r respectively.

b. (Do not hand in) Verify that the following independence systems do not form matroids in general (by giving counterexamples):

   (a) U is the set of edges of an undirected bipartite graph $G = (V, E)$, and $\mathcal{I}$ contains all matchings in G (i.e., subsets of edges such that no two edges share a common endpoint).

   (b) U is the set of arcs of a directed graph $G = (V, A)$ and $\mathcal{I}$ contains all subsets of arcs that do not contain a directed cycle.

c. (Do not hand in) Show that the cardinality of any two bases in a matroid is the same.

d. Given $M = (U, \mathcal{I})$, define the dual matroid $M = (U, \mathcal{I}^*)$ where the bases of $\mathcal{I}^*$ are the complements of bases in M (and hence the independent sets of $\mathcal{I}^*$ are subsets of these complements). Show that M is also a matroid.

   Given element weights $w : U \to \mathbb{R}$, the min-weight base problem seeks to find a base $B \in \mathcal{I}$ of minimum total weight $w(B) := \sum_{e \in B} w(e)$. (You may assume distinct edge weights for the following parts.)

   (a) (Do not hand in) Show that there is a unique min-weight base.

   (b) (Do not hand in) Show that repeatedly applying the cut rule (pick a cut and color the lightest edge in the cut blue) and the cycle rule (pick any cycle and color the heaviest edge on the cycle red) to any matroid colors all the edges red/blue, and the resulting blue edges form a min-weight base.

(c) Consider the natural generalization of Kruskals algorithm: let $S \leftarrow \emptyset$, sort elements by weight and consider them in non-decreasing order, when considering an element e add it to the S if $S \cup \{e\}$ is independent. Show that this algorithm solves the min-weight base problem. (You may assume an oracle that given $S \subseteq U$ answers the membership query $S \in \mathcal{I}$? in constant time.)

(d) Show that for an independence system $(U, \mathcal{I})$, if the greedy algorithm correctly solves the min-weight base problem for all settings of weights, then the independence system is a matroid. Hence the greedy algorithm characterizes matroids.

Problem 4

(Clustering on a Tree.) For a tree $T = (V, E)$ with positive edge lengths, define $d)_T(u, v)$ to be the shortest-path distance between u and v in T according to these edge-lengths. Given tree $T = (V, E)$ with n nodes, and an integer $k \in \mathbb{Z} \geq 0$, you want to pick a set C of k centers from T such that

$$\sum_{v \in V} (\text{distance of } v \text{ to the closest center in } C)$$

is minimized.[1] Give a dynamic-programming solution that runs in time poly(k, n).
(Hint: it may help to consider the special case where each node in T has degree 3 or less. Also, this problem does not use any ideas weve been talking about in the class thus far.)

[1]If we define $d_T(v, C) := min_{c \in C} d_T(v, c)$, the goal is to find C with $|C| = k$ to minimize $\sum_v d_T(v, C)$.