Homework 4 – Due Thursday, September 22, 2016 on Canvas

Please refer to HW guidelines from HW1, course syllabus, and collaboration policy.

Exercises (Do not hand in.)

- More exercises in KT, Chapter 4.
  - (Minimum spanning trees) Problems 1, 2, 8, 9, 10, 11, 19 (20 is similar), 21, 22, 26, 28.
  - (Other greedy graph algorithms) Problems 28, 29, 30.
- (MST with negative-weight edges) Give a $O(m \log n)$ algorithm which takes as input a connected, undirected graph $G = (V, E)$ and distinct weights w, *some of which may be negative*, and outputs a minimum spanning subgraph $F \subset E$, i.e. it outputs a set of edges F such that (V, F) is connected and $\sum_{e \in F} w_e$ is minimized. Note that F need not be a tree.
- Suppose G is a connected, undirected graph in which all edge weights w_e are positive and distinct. Consider the graph G' with the same vertices and edges as G, but with edge weights $w'_e = w_e^2 + 6$. Prove or disprove:
  - For every $u, v \in V$, the shortest path from u to v is the same in G as in G'.
  - The minimum spanning trees of G and G' are the same.
- **(Distributed MST algorithms)** Read about, and prove the correctness of, Borvka's algorithm for the MST problem.

Problems to be handed in, 10 points each, 2-page limit per problem (Don't forget to prove correctness and analyze time/space requirements of your algorithm.)

1. (**Greedy: An Exchange Argument**) Chapter 4, problem 13.
2. **(Shortest paths for other measures of distance)** Dijkstra's algorithm assumes that the cost of a path is the sum of the lengths of the edges in the graph. However, path costs are not always like that. For example,
   (a) In a road network, the travel time on a path might have to include extra time for turning off a busy road. Thus, the cost of path P might be $\sum_{e \in P} \ell(e) + c_1(\#\text{ of left turns}) + c_2(\#\text{ of right turns})$, where c_1 and c_2 represent the extra time needed for each kind of turn.
   (b) If the edges are links in a computer network and edge weights are bandwidths, then the "value" of a path would be the minimum of the bandwidths along the path. Note that in this case we would want to *maximize* the value, not minimize it.
   (c) Even if the path cost is still a simple sum, the edge weights may change over time. For example, in the "Canadian travel" problem (KT, Chapter 4, problem 18), the time of travel along an edge (u, v) is a function of the time t at which you depart from u. See the problem statement in the book for details.

For each of these three types of "distances", show how to modify Dijkstra's algorithm so that it computes the best path according to the new distance measure. In each case, indicate

- which steps of the algorithm you would change,
- how these changes affect the running time, and
- what changes are necessary to proof of correctness we saw in class to conclude that the modified algorithm is correct.

For part (2a) above, assume that given the edges (=streets) by which a path enters and leaves a given vertex (=intersection), there is a constant-time procedure that determines if the driver needed to turn left, turn right, or go straight.

[Note: After completing this problem, do problems 19 and 20 in chapter 4 as exercises; do not hand them in.]

3. (**Space-Efficient Algorithm for MST**, 10 points, 2-page limit) You are designing an MST algorithm that runs on a machine with a small workspace. The input to your algorithm is an undirected graph stored on a remote server and presented in the following form: an integer $n = |V|$, followed by a list of m edges along with their weights, that is, a list of triples (u, v, w_{uv}), where $u, v \in \{1, ..., n\}$ and $w_{uv} > 0$.

   Give an algorithm which makes a single pass through the list of edges, using only $O(n)$ space, and outputs a minimum spanning tree at the end of its pass. Your algorithm should work no matter what order the edges are listed in.

   (Hint: as each new edge is considered, decide whether or not to add it to your tree and, if necessary, which edges from the current tree to discard).

   You may assume that the graph is connected and that all edge weights are distinct.

4* (**Optional, no collaboration**) For any edge e in any graph G, let $G \setminus e$ denote the graph obtained by deleting e from G.

   (a) Suppose we are given a directed graph G in which the shortest path from vertex s to vertex t passes through every vertex of G. Describe an algorithm to compute the shortest-path distance from s to t in $G \setminus e$, for every edge e of G, in $O(E \log V)$ time. Your algorithm should output a set of E shortest-path distances, one for each edge of the input graph. You may assume that all edge weights are non-negative. [Hint: If we delete an edge of the original shortest path, how do the old and new shortest paths overlap?]

   (b) Let s and t be arbitrary vertices in an arbitrary directed graph G. Describe an algorithm to compute the shortest-path distance from s to t in $G \setminus e$, for every edge e of G, in $O(E \log V)$ time. Again, you may assume that all edge weights are non-negative.