# Algorithms Lab 11

(MST)

csci 2200, Laura Toma. Bowdoin College

## This week's topics

- MST: properties, the cut theorem; Kruskal's and Prim's algorithms

Review the algorithms discussed in class this week. This is a great time to understand all details and ask questions.

## In lab: (collaboration level 0: everything allowed)

1. **Prim's algorithm:** Following yesterday's discussion, we need to store the edges with one endpoint in $T$ so that we can select the minimum-weight edge fast. We'll use a priority queue. The immediate idea is for the priority queue to store *edges*. Note however that for any vertex $v$ not in $T$, even though it may have more than one edge connecting it to $T$, we'll only use the **smallest** one. So why store all? Just store the minimum edge connecting a vertex to $T$. This gives the idea for the priority queue to store *vertices*: each vertex $v$ not in $T$ yet has priority equal to the weight of the smallest edge that connectes it to a vertex u in $T$.

   - The priority queue will store all the vertices that are not in $T$ yet.
   - A vertex $v$ in the PQ has priority equal to the weight of the minimum edge that connects $v$ with a vertex already in $T$. The other endpoint of this edge is stored in $visit(v)$.
   - Essentially the priority queue stores all edges that *cross* the cut between the vertices in $T$ and the vertices in $V - T$ (If a vertex not in $T$ is connected by several edges to vertices in $T$, the pq will store only one of these edges, the smallest).
   - Initially $T$ is empty and PQ contains all vertices in $G$ with priority $\infty$, except an arbitrary vertex $v$ which has priority 0.

   (a) Show how Prim's algorithm run on the example graph in the notes (or textbook).

(b) What is the role of checking whether $v \in PQ$?

(c) How many INSERT operations are performed by the algorithm?

(d) How many DELETE-MIN operations are performed by the algorithm?

(e) How many DECREASE-KEY operations are performed by the algorithm?

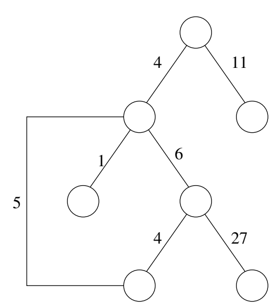
(f) Assuming the priority is implemented as a heap, what is the complexity of the algorithm?

2. (CLRS 23.1-1) Consider an undircted, connected, weighted graph $G$. Is it true that the minimum-weight edge in $G$ belongs to some MST of $G$? If yes, why?

3. (CLRS 24.2-4) Suppose that all edge weights in a graph are integers in the range from 1 to $|V|$. How can you take advantage of this in Kruskal's algorithm, and how fast can you make it run? What if the edge weights are integers from 1 to $W$ for some constant $W$?

4. Is the path between a pair of nodes $u$ and $v$ in a minimum spanning tree (i.e. the path composed of edges in the MST) necessarily the minimum weight path between $u$ and $v$ if we consider all the edges in the graph $G$? If yes, prove it. If no, provide a counterexample.

5. Consider an undirected weighted graph which is formed by taking a binary tree and adding an edge from *exactly one* of the leaves to another node in the tree. We call such a graph a *loop-tree*. An example of a loop-tree could be the following:




Let $n$ be the number of vertices in a loop-tree and assume that the graph is given in the normal edge-list representation without any extra information. In particular, the representation does not contain information about which vertex is the root.

(a) How many edges are in a graph of $n$ vertices?

(b) How long time would it take Prim's or Kruskal's algorithms to find the minimal spanning tree of a loop-tree?

(c) Describe and analyze a more efficient algorithm for finding the minimal spanning tree of a loop-tree. Remember to argue that the algorithm is correct.