Lecture #12: Minimum Spanning Trees
(Tucker Section 4.2)
David Mix Barrington
3 October 2016
Minimum Spanning Trees

• Definitions and Motivation
• The Prim and Kruskal Algorithms
• An Example
• Correctness of the Algorithms
• Implementation of the Algorithms
• Using an MST to Approximate TSP
• A Better Approximation
Definitions and Motivation

• Suppose we have a weighted undirected graph where nodes are towns, edges are roads, and weights are the cost of plowing a road.

• We have a limited budget and can only plow some of the roads, but we need to make it possible to get from any town to any other.

• We want a subset of the edges forming a tree containing all the nodes, a spanning tree.
Minimum Spanning Trees

• A spanning tree has a weight, the sum of the weights of its edges. A minimum spanning tree is one such that no other has less weight.

• There may be more than one MST for a given graph. In fact, if each edge has weight 1, then every spanning tree is an MST.

• We’ll assume for the rest of the lecture that the original graph is connected and that all the weights are positive.
Two Algorithms for MST

• We’ll present two algorithms to find an MST. Both are greedy algorithms, considering all the edges of a certain type and taking the edge of minimum weight.

• Prim’s Algorithm builds a single tree by starting with one node, then repeatedly adding the cheapest edge that connects a node in the tree to one outside of it.

• Kruskal’s Algorithm always takes the cheapest edge that does not form a cycle with the existing edges.
An Example

- Here’s a weighted graph with 25 nodes and 40 edges.
- We’ll run both algorithms in turn to get a minimum spanning tree for it.
- We’ll start Prim with the first 1-edge.
An Example

- The nodes in green and edges in red are in the tree.
- Prim goes on to take edges with one green endpoint, cheapest first. The next five it takes, in order 2, 4, 2, 1, 2, are shown in green.
An Example

- At this point we have a choice of size-3 edges.
- That choice doesn’t affect the tree, but a later one could.
- It turns out that either of the purple edges could be in the MST.
An Example

- It turns out that either of the purple edges could be in the MST.
- Once we make this choice, we’ll get the next edges in green.
An Example

• Now there are two ways to get the last two nodes, both with the same cost.

• We’ve used 24 edges of total weight 65.
An Example

- To begin Kruskal, we take all the edges of weight 1 because we don’t form a cycle.
- We start building up connected components.
- The weight-2 edges also don’t form any cycle.
An Example

- The weight-2 edges also don’t form any cycle. (We’ve now added them in the diagram.)

- But the 3-edges will. We can’t add both the vertical 3-edges in purple, but we can add either one.
An Example

• Here we did not take the purple 3-edge because we took the 3-edge to its left first.

• Now we have only 7 connected components left. The five 4-edges don’t form cycles, and reduce us to only two.
An Example

• We’ll complete the spanning tree with one of the two 5-edges with one red and one green endpoint.

• There were four possible trees we might have produced, each of total weight 65.
Correctness of Prim

• Assume (for simplicity) that all the edge weights are distinct.

• We’ll get a contradiction from the assumption that some tree $T'$ has smaller weight than the tree $T^*$ produced by Prim.

• Let $e = (u, v)$ be the first edge not in $T'$ that we put into $T^*$, and consider the point at which we did so. Let $X$ be the nodes in $T^*$ at that point. We know that $e$ is cheaper than any other edge with one endpoint in $X$. 
Correctness of Prim

• Since $T'$ is a spanning tree, there is a path from $u$ to $v$ in it, and this path must leave $X$ by some edge $e'$.

• Since $e'$ has one endpoint in $X$ and one out of it, it must have weight larger than that of $e$.

• Now replacing $e'$ with $e$ gives a spanning tree with weight less than that of $T'$, a contradiction. (On HW#3 you will show that this replacement always yields a tree.)
Correctness of Kruskal

- We can similarly show that Kruskal is correct. Again consider an MST $T'$. As Kruskal adds edges to its edge set, $e = (u, v)$ be the first edge it adds that is not in $T'$, and let $Z$ be the edges in the set at that point (before $e$ is added).

- Node $u$ and $v$ are in different connected components of the forest $Z$. $T'$ must have a path from $u$ to $v$, and this path must contain an edge $e'$ that connects two components of $Z$.

- Once again replacing $e'$ with $e$ in $T'$ gives a spanning tree smaller than the alleged MST $T'$. 
Implementation and Time

• How do we implement these algorithms?

• For Prim we can use a priority queue like that in UCS. We put edges into the PQ when they are found. At each stage we start pulling out the minimum-weight edge in the PQ, rejecting them if both endpoints are in X, until we find exactly the edge we need. If not it is the edge needed by Prim.

• Overall we have $O(e)$ PQ operations for $O(e \log e)$ total time.
Implementation and Time

- With Prim we could tell whether to reject an edge using a flag on each vertex to say whether it was in $X$.

- But with Kruskal we want to reject an edge if it forms a cycle with existing edges, which happens if the endpoints are in the same connected component of the forest made by the existing edges.
Implementation and Time

• This is the dynamic transitive closure problem, to maintain the set of connected components as new edges are introduced.

• We don’t want to, say, DFS the graph again for each new edge.

• In CS 311 you will probably see the union-find algorithm, which solves this problem in nearly linear time. With that, the running time of Kruskal is also about $O(e \log e)$. 
Applying MST to TSP

• Recall that Tucker presented an algorithm to approximate the least-weight Hamilton circuit in a weighted graph, getting a tour whose weight was at most twice the optimal weight. The weights were symmetric and obeyed the triangle inequality.

• In such a graph, we can always shortcut two edges \((u, v)\) and \((v, w)\) with a single edge \((u, w)\), without adding weight.
Applying MST to TSP

• Consider an MST of this weighted graph. Change each edge into two directed edges, and make an Euler tour E of the resulting directed graph.

• The weight of E is twice that of the MST. And since dropping any edge from the optimal tour $C^*$ gives a spanning tree, the weight of E is at most twice that of $C^*$.

• Shortcutting E leads to a Hamilton tour with no more weight than that of E.
A Better Approximation

• The nodes $O$ of odd degree in the MST must have a perfect matching. (Why?) Later we will see how to find the minimum-weight matching in polynomial time.

• If we add the matching edges to the MST, we get a graph where all vertices have even degree, and there must be an Euler tour. Shortcutting this tour gets us a Hamilton tour.
A Better Approximation

• We already know that the weight of the MST is less than that of the optimal tour $C^*$.

• The nodes of $O$ divide $C^*$ into an even number of paths. If we two-color these, one color set has weight less than half that of $C^*$. And the shortcutting of this is a matching with no more weight.

• So the total weight of our eventual Hamilton tour is at most $3/2$ that of $C^*$. 