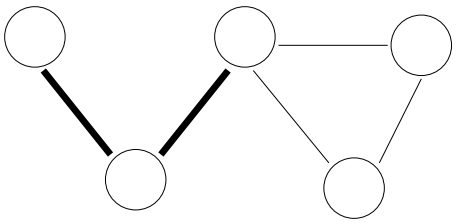
CSC 263 Lecture 9

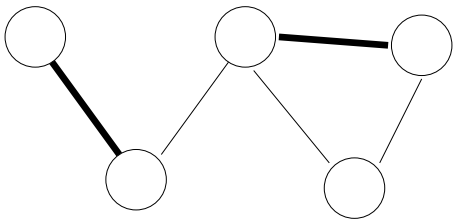
September 13, 2006

16 Minimum Cost Spanning Trees (MCSTs)

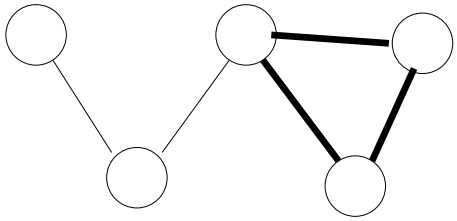
Let $G = (V, E)$ be a connected, undirected graph with edge weights $w(e)$ for each edge $e \in E$. A *tree* is a subset of edges $A \subset E$ such that A is connected and contains no cycles. The following diagram shows a graph with three different subsets A (the thick edges are in A, the thin ones are not). One is a tree and the other two aren't.



Tree

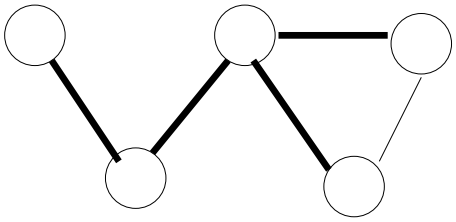


Not a tree (not connected)



Not a tree (has a cycle)

A *spanning tree* is a tree A such that every vertex $v \in V$ is an endpoint of at least one edge in A. Notice that any spanning tree must contain $n - 1$ edges, where $|V| = n$ (proof by induction on n).

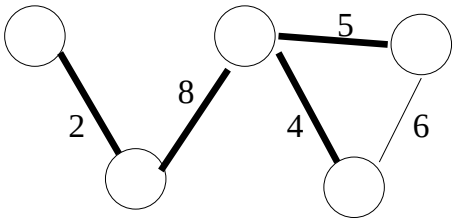


Spanning Tree

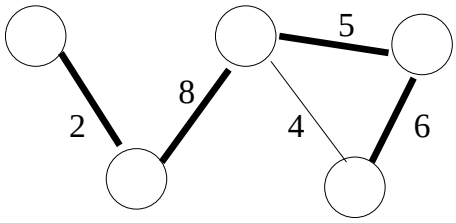
A *minimum cost spanning tree* is a spanning tree A such that

$$w(A) = \sum_{e \in A} w(e)$$

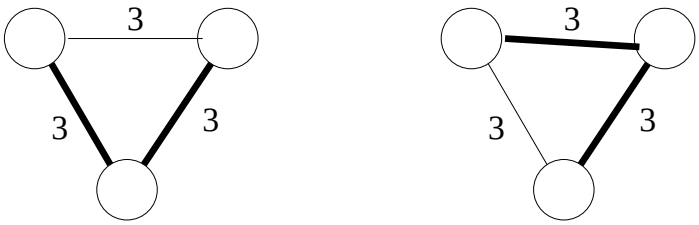
is less than or equal to $w(B)$, for all other spanning trees B.



Minimum Cost Spanning Tree



Spanning tree, but not minimum cost



Two different MCSTs on the same graph

Finding MCSTs is important in practice: imagine you have a network of computers that are connected by various links. Some of these links are faster, or more reliable, than others. You might want to pick a minimal set of links that connects every computer (in other words, a spanning tree) such that these links are overall the best (they have minimum cost). Once you have found these links, you never have to use the remaining slower, or less reliable, links.

We will look at two algorithms for contructing MCSTs. The first is Prim's Algorithm.

16.1 Prim's Algorithm

Prim's algorithm uses a Priority Queue ADT. This operates on a set S where each element $x \in S$ has a priority $p(x)$ which comes from a well-ordered universe (usually the natural numbers). There are three operations on this set:

- `INSERT(S,x)`: insert an element x in the set S.
- `ISEMPTY(S)`: return true if S is empty.
- `EXTRACT-MIN(S)`: remove and return an element $x \in S$ with minimum priority.

In addition, we will need the operation `DECREASE-PRIORITY(x, p)` will set the priority of x, which is in the queue, to be p (p is less than x's current priority).

```
PRIM-MST(G=(V,E),w:E->Z)
    A := {};
    initialize a priority queue Q;
    for all v in V do
       priority[v] := infinity;
       p[v] := NIL;
       INSERT(Q,v);
    pick some arbitrary vertex s in V and let priority[s] := 0;
    for each v in adjacency-list[s] do
       if v in Q and w(u,v) < priority[v] then
          DECREASE-PRIORITY(v, w(s, v))
          p[v] := s;
    while ( not ISEMPTY(Q) ) do
       u := EXTRACT-MIN(Q);
       A := A U {(p[u],u)};
       for each v in adjacency-list[u] do
          if v in Q and w(u,v) < priority[v] then
             DECREASE-PRIORITY(v, w(u,v));
             p[v] := u;
          end if
       end for
    end while
 END PRIM-MST
```

Prim's algorithm grows an MCST A starting with an empty set. Even though A is technically a set of edges, we can consider the vertices in A as the set of vertices that are the endpoints of edges in A. When we start the algorithm, however, we consider the vertex s to be in A even though there are no edges in A. From now on, we just keep adding edges to A by finding the "lightest" edge that has one endpoint in A and the other endpoint outside of A.

16.2 Correctness

The correctness of Prim's algorithm is given by the following theorem.
Theorem: If $G = (V, E)$ is a connected, undirected, weighted graph, A is a subset of some MCST of G, and e is any edge of minimum weight with one endpoint in A and one endpoint outside of A, then $A \cup \{e\}$ is a subset of some MCST of G.
Proof: Let T be a MCST of G that contains A as a subset. If e is in T, then we are done. Otherwise, we construct a different MCST T' that contains e. If we add edge e to T, we create a

cycle in the resulting graph ($T \cup \{e\}$ is not a tree anymore). This cycle must contain another edge e' with one endpoint in A and one endpoint outside of A (otherwise, every endpoint of A is already in T which means that edge e cannot exist). Since we picked e to have minimum weight among such edges, it must be the case that $w(e) \leq w(e')$. Now, let $T' = T \cup \{e\} - \{e'\}$. T' is connected, and it is acyclic (since we removed one edge from the only cycle in $T \cup \{e\}$), so it is a spanning tree of G that contains A as a subset. Moreover, $w(T') - w(T) = w(e) - w(e') \leq 0$ so T' has total weight no greater than T, i.e., T' is an MCST that contains $A \cup \{e\}$ as a subset.