Minimum-cost spanning tree (Kruskal's algorithm).

**Input.** An connected undirected graph \( G = (V, E) \) with a cost function \( c \) on the edges.

**Output.** \( S = (V, T) \), a minimum-cost spanning tree for \( G \).

1. **Initialization:**
   - \( T \leftarrow \emptyset \)
   - \( VS \leftarrow \emptyset \)
   - Construct a priority queue \( Q \) containing all edges in \( E \):
   - For each vertex \( v \in V \), do add \( \{ v \} \) to \( VS \)
   - While ( \( |VS| > 1 \) ) {
     - Choose \((v, w)\), an edge in \( Q \) of lowest cost:
     - Delete \((v, w)\) from \( Q \)
     - If ( \( v \) and \( w \) are in different sets \( W_1 \) and \( W_2 \) in \( VS \) ) {
       - Replace \( W_1 \) and \( W_2 \) in \( VS \) by \( W_1 \cup W_2 \):
       - Add \((v, w)\) to \( T \)
     - }
   - }

**Minimum-cost spanning tree algorithm**

**Lemma 1:** Let \( G = (V, E) \) be a connected, undirected graph and \( S = (V, T) \) a spanning tree for \( G \). Then

a) For all \( v_1 \) and \( v_2 \) in \( V \), the path between \( v_1 \) and \( v_2 \) in \( S \) is unique, and

b) If any edge in \( E \setminus T \) is added to \( S \), a unique cycle results.

**Proof.**

a) Is trivial, since if there were more than one path there would be a cycle.

b) Is likewise trivial, since there must already be a path between the endpoints of the added edge.

**Lemma 2:** Let \( G = (V, E) \) be a connected, undirected graph and \( c \) a cost function on its edges. Let \( \{(V_1, T_1), (V_2, T_2), \ldots, (V_k, T_k)\} \) be any spanning forest for \( G \) with \( k > 1 \).

Let \( T = \bigcup_{i=1}^{k} T_i \).

Suppose \( e = (v, w) \) is an edge of lowest cost in \( E \setminus T \) such that \( v \in V_1 \) and \( w \) not in \( W \). Then there is a spanning tree for \( G \) which includes \( T \cup \{ e \} \) and is of as low a cost as any spanning tree for \( G \) that includes \( T \).

**Proof:** Suppose to the contrary that \( S' = (V, T') \) is a spanning tree for \( G \) such that \( T' \) includes \( T \) but not \( e \), and that \( S' \) is of lower cost than any spanning tree for \( G \) that includes \( T \cup \{ e \} \).

By Lemma 2 (b), the addition of \( e \) to \( S' \) forms a cycle. The cycle must contain an edge \( e' = (v', w') \), other than \( e \), such that \( e' \in V_1 \) and \( w' \in V_1 \),
By hypothesis $c(e) \leq c(e')$.

Consider the graph $S$ formed by adding $e$ to $S'$ and deleting $e'$ from $S'$. $S$ has no cycle, since the only cycle was broken by deletion of edge $e'$. Moreover, all vertices in $V$ are still connected, since there is a path between $v'$ and $w'$ in $S$. Thus $S$ is a spanning tree for $G$. Since $c(e) \leq c(e')$, $S$ is no more costly than $S'$. But $S$ contains both $T$ and $e$, contradicting the minimality of $S'$.

QED

Minimum Cost spanning Tree

Running time: $O(|E| \log(|V|))$

Explanation in next class. Also we can do better

**Prim's MST Algorithm**

Initialize $X = \{s\}$ [$s \in V$ chosen arbitrarily]

$T = \emptyset$ [invariant: $X =$ vertices spanned by tree-so-far $T$]

While ($X \neq V$) {

Let $e = (u,v)$ be the cheapest edge of $G$ with $u \in X, v$ not$\in X$.

Add $e$ to $T$

Add $v$ to $X$.}
Note: While loop increases the number of spanned vertices in cheapest way possible.

Definition: A cut of a graph $G = (V, E)$ is a partition of $V$ into 2 non-empty sets.

**Empty Cut Lemma**: A graph is not connected $\iff \exists$ cut $(A, B)$ with no crossing edges.

Proof: ($\Rightarrow$) Assume the RHS. Pick any $u \in A$ and $v \in B$. Since no edges cross $(A, B)$ there is no $u,v$ path in $G$. $\Rightarrow G$ not connected.

($\Leftarrow$) Assume the LHS. Suppose $G$ has no $u-v$ path. Define $A = \{\text{Vertices reachable from } u \text{ in } G\}$ (u’s connected component) $B = \{\text{All other vertices}\}$ (all other connected components)

Note: No edges cross cut $(A, B)$

**Double-Crossing Lemma**: Suppose the cycle $C \subseteq E$ has an edge crossing the cut $(A, B)$: then so does some other edge of $C$.

**Lonely Cut Corollary**: If $e$ is the only edge crossing some cut $(A, B)$, then it is not in any cycle. [If it were in a cycle, some other edge would have to cross the cut!]

**Claim**: Prim’s algorithm outputs a spanning tree. //Not claiming MST yet

Proof: (a) Algorithm maintains invariant that $T$ spans $X$ [straightforward induction - you check] lonely in this cut corollary

QED

(b) Can’t get stuck with $X \neq V$; otherwise the cut $(X, V - X)$ must be empty; hence by Empty Cut Lemma input graph $G$ is disconnected.

(3) No cycles ever get created in $T$.

Consider any iteration, with current sets $X$ and $T$. Suppose $e$ gets added. Key point: $e$ is the first edge crossing $(X, V - X)$ that gets added to $T \Rightarrow$ its addition can’t create a cycle in $T$ (by Lonely Cut Corollary). QED!

---

**The Cut Property**

**Assumption**: Distinct edge costs.

**CUT PROPERTY**: Consider an edge $e$ of $G$. Suppose there is a cut $(A, B)$ such that $e$ is the cheapest edge of $G$ that crosses it. Then $e$ belongs to the MST of $G$.

Proof: Suppose there is an edge $e$ that is the cheapest one crossing a cut $(A, B)$, yet $e$ is not in the MST $T^*$.

(Idea: Exchange $e$ with another edge in $T^*$ to make it even cheaper to get a contradiction).

Now suppose $e$ not in $T^*$, since $G$ is connected there must be an $f \in T^*$.

That crosses the cut $(A, B)$.

Consider $T^* \cup \{e\} - \{f\}$ and check if it is spanning tree of $G$. If so clearly the new tree $T^* \cup \{e\} - \{f\}$ is of lower cost, contradiction.
T*U\{e\}−\{f\} is not a spanning tree then adding edge e has created a cycle.

By the Double-Crossing Lemma Some other edge e' of C [with e' != e and e' ∈ T*] crosses (A,B).

You check: T = T*U\{e\}−\{e'\} is also a spanning tree. Since c(e) < c(e'),
T cheaper than purported MST T*,
QED.

**Theorem:** Prim’s algorithm always outputs a minimum-cost spanning tree.

**Proof:** Prim’s algorithm outputs a spanning tree T. (previous proof)
Every edge e ∈ T is explicitly justified by the Cut Property.
hence T is a subset of the MST ⇒ Since T is already a spanning tree,
it must be the MST.

**Running Time** \(O(|E| \log (|V|))\)