A spanning tree of an undirected, connected graph $G = (V, E)$ is a tree $T = (V, E')$ where $E' \subseteq E$.

A minimum spanning tree of an undirected, connected, weighted graph $G = (V, E)$ with weighting $w: E \rightarrow \mathbb{R}$ is a spanning tree $T^* = (V, E^*)$ such that for all spanning trees $T = (V, E')$ of $G$, $w(T^*) = \sum_{e \in E^*} w(e) \leq \sum_{e \in E'} w(e) = w(T)$.

Minimum spanning trees have a number of useful applications. For one, they are the minimal cost subgraph that connects all vertices. One can imagine this is useful for infrastructure construction (think building an intranet, electrical grid, water supply, etc.).

MSTs can be used for approximation algorithms. For the TSP that obeys the triangle inequality, one can generate a route that is no worse than twice the optimal route from an MST.

There are several ways to find MSTs, but we will look at two greedy algorithms: Prim's and Kruskal's. The only difference between the two is in the greedy choice (although implementation details vary).
Prim's algorithm works by starting with any seed vertex and growing an MST by selecting the lowest cost edge from the tree to outside the tree.

Ex.1

Start with E and grow our blossoming blob.

Select A

Select D (or C)

Select C

Select B (AB or BC)
If T is the ST selected by Prim's algorithm for G=(V,E) with weighting w, then T is an MST of G.

**Proof:** Assume T is not an MST. Let S = e₁, ..., e_{|V|-1} be the sequence of edges chosen in order. Let T* be an MST that contains the longest prefix of S possible.

Let e₁ = (u,v) be the first edge in T not in T*.

Let V₀ ⊆ V be the set of vertices chosen before e₁ by Prim's.

Since T* is a MST, there is a path u → v in T*.

Let e₁' = (a,b) be the first edge on this path such that a ∈ V₀ and b ∉ V₀.

If we remove e₁' from T* and add e₁ to get T', notice that T' is a ST of G. This yields three cases.

\[ w(e₁') > w(e₁) \]

In this case, \( w(T') < w(T*) \), hence T* was not an MST.

\[ w(e₁') \leq w(e₁) \]

In this case, Prim's would select e₁' instead of e₁.

\[ w(e₁') = w(e₁) \]

In this case, \( w(T') = w(T*) \), so T' is also an MST. Moreover, T' contains a longer prefix of S than T*.

All three cases lead to a contradiction, so our assumption that T is not a MST was incorrect.
Prim's \((G = (V, E), w)\)

Let \(P\) be a priority queue of \((V, E)\) ordered by increasing \(w(e)\)

Pick a start vertex \(v \in V\)

Let \(T = (V, \emptyset)\)

Let \(R = V\)

Add \((v, T)\) to \(P\)

While \(R \neq \emptyset\)

Poll \((u, e)\) from \(P\)

If \(u \not\in R\), continue

Remove \(u\) from \(R\)

Add \((e)\) to \(T\) \((t(e) \not= T)\)

For each neighbor \(v_i\) of \(u\) with \(v_i \in R\)

Add \((v, (u, v))\) to \(P\)

Return \(T\)

The WC RT here comes from potentially stuffing most edges into the priority queue for \(O(E \log E) = O(E \log V^2) = O(E \log V)\).

Further improvements can bring the runtime down to \(O(E + V \log V)\).