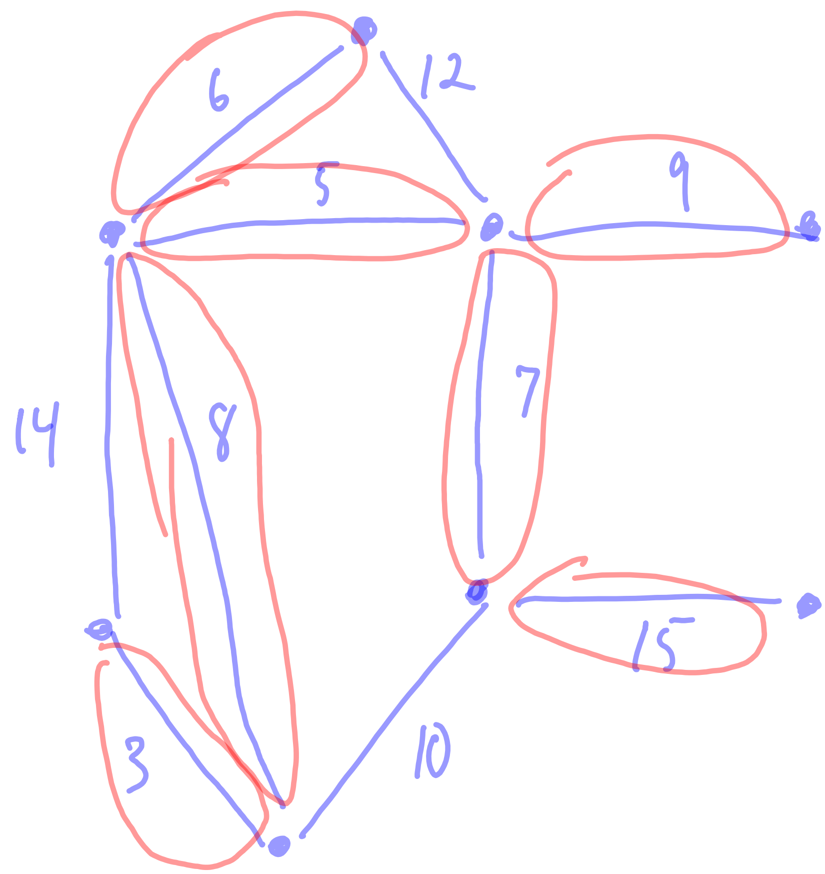
Suppose we have a set of $n$ locations, and we wish to build a connected network on top of them. The network should be connected (there should be a path between any two locations in the network), and subject to this constraint, we wish to build the network as cheaply as possible.

Note that a solution to this must be a tree (if the network contains a cycle, we can remove one of the connections to obtain a cheaper network and still satisfy the connectivity constraint).

In graph theory, a tree which contains every vertex of the graph is known as a **spanning tree**.

If we assign non-negative weights $w(u, v)$ to each edge $\{u, v\}$ in the graph (i.e. the cost to connect two locations in the network), then a **minimum spanning tree** (MST) is a spanning tree such that the sum of the weights of the edges in the tree is minimized.

Example of a MST:



A key observation of MSTs (for simplicity, assume the weights on the edges are distinct):

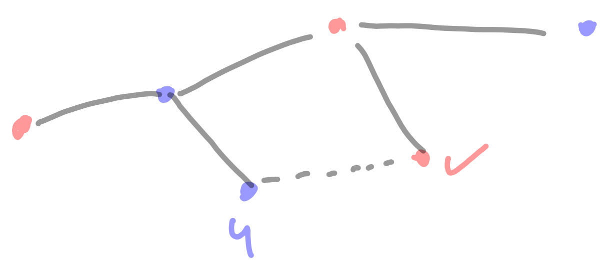
**Theorem**

Let $T$ be a MST of $G=(V,E)$, and let $A \subset V$.

Suppose $\{u,v\} \in E$ is the least-weight edge connecting $A$ to $V \setminus A$. Then $\{u,v\} \in T$.

**Proof**

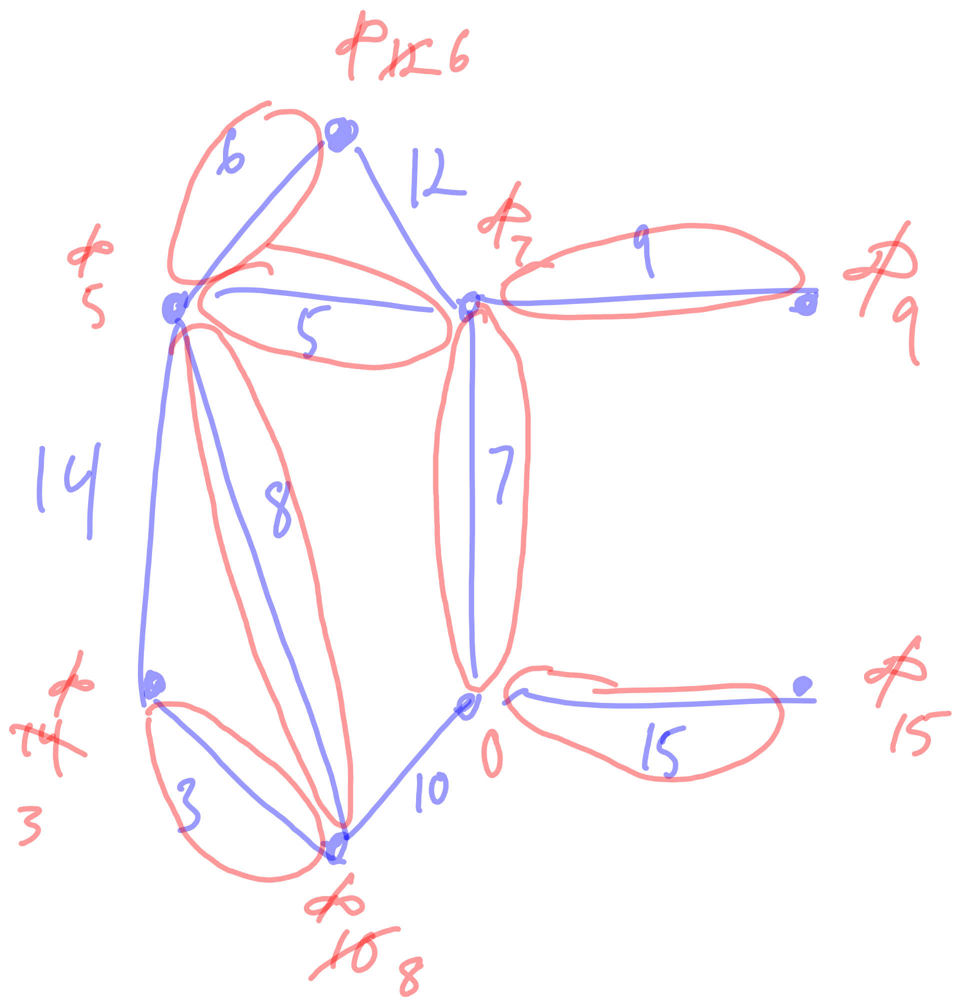
For the sake of contradiction, assume that it is not in $T$. Let blue vertices be $A$ and red vertices be $V \setminus A$.



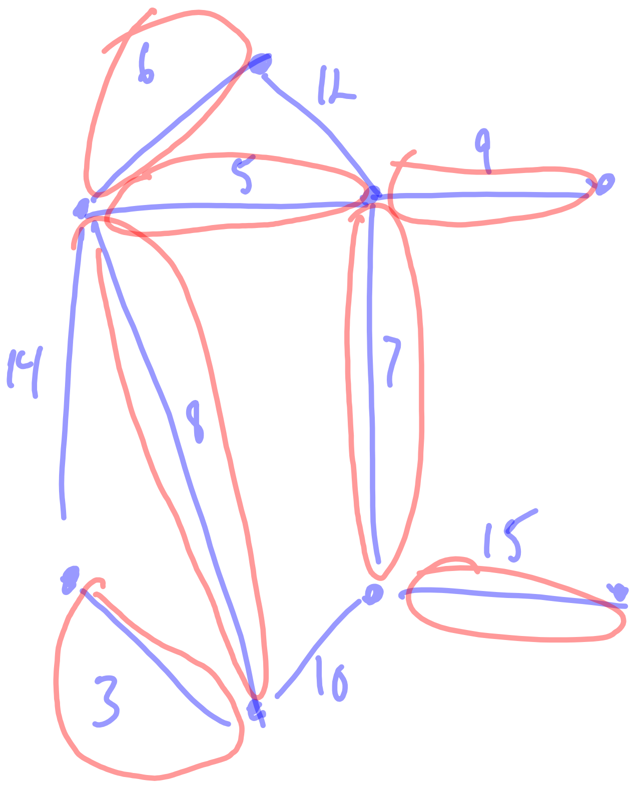

Follow path from u to v in $T$, and remove an edge connecting a red vertex to a blue vertex. Such an edge must exist because this path begins at a blue vertex and ends at a red vertex. We can then add $\{u,v\}$ to obtain a spanning tree that is cheaper than $T$, a contradiction.

**Prim's algorithm**:

- Maintain a key for each vertex (initially set to $\infty$). Arbitrarily pick a vertex and change its key to 0. Let $Q$ initially be the set of all vertices.
- Find the vertex $u$ in $Q$ with the smallest ~~weight~~ key, and remove $u$ from $Q$.
  - For each neighbor $v$ of $u$, check if $w(u, v) < key(v)$. If so, set $key(v) = w(u, v)$, and mark $u$ as the "parent" of $v$ in the tree.



**Kruskal's algorithm**: Repeatedly pick the edge with the smallest weight as long as it does not form a cycle.



Here is an overview of MST algorithms:

- Prim's algorithm:
  - Maintains one tree
  - Utilizing a proper data structure (binary heap), runs in time $O(m \log n)$.

- Kruskal's algorithm:
  - Maintains a forest.
  - When using a data structure which we will discuss later in the class, runs in time $O(m \log m)$.

    Union - Find

- There is a randomized algorithm due to Karger, Klein, and Tarjan [1993] which runs in expected time $O(n + m)$.