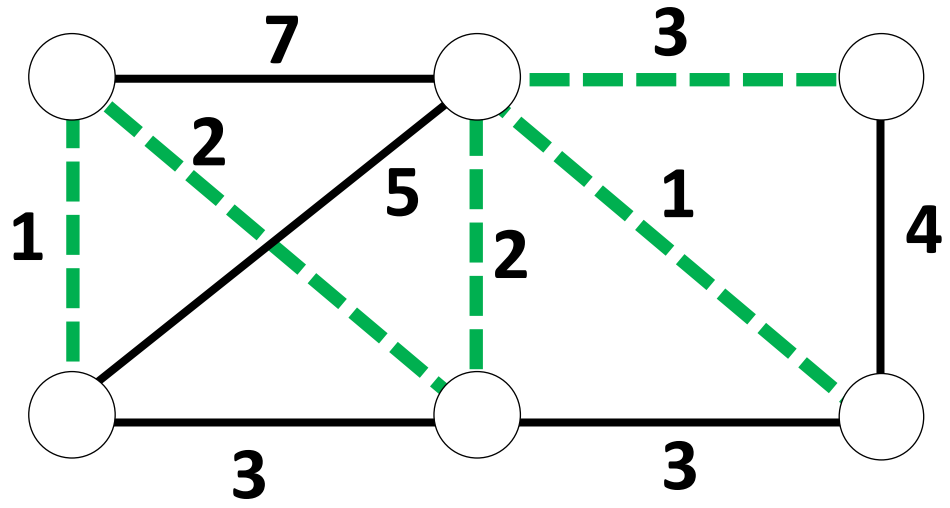
Greedy Algorithms
CSCI 432

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.



Basic algorithm analysis questions:

1. ~~Is the solution valid? (Does it actually find a spanning tree?)~~
2. Is the solution optimal? (Does it find the best spanning tree?)
3. What is the running time?

MST Cut Property

Assume unique edge costs.

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof:

We need a plan:

Suppose we show that every edge selected by the algorithm is part of the optimal MST. Can we conclude that the selected edges form the MST?

How could a set of edges not be the MST?

It could have edges not in the MST. ✗

It could be missing edges from the MST. ✗

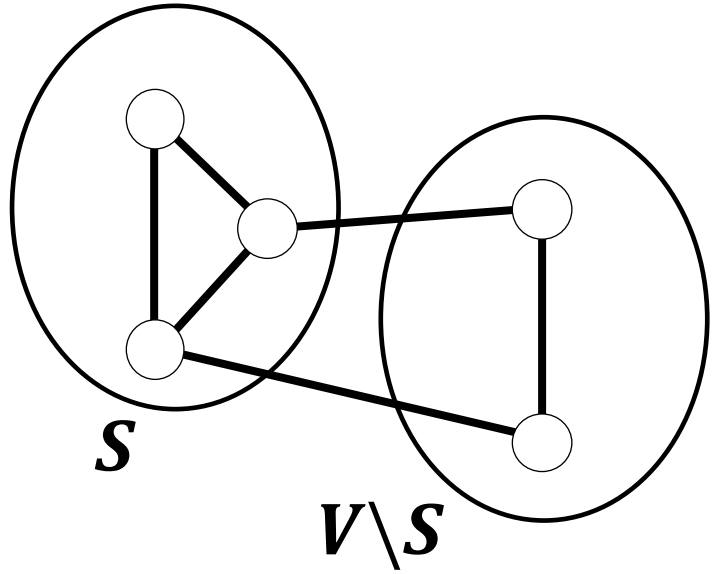
Thus, the spanning tree constructed *is* the MST.

Spanning Trees all have $|V| - 1$ edges.

MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

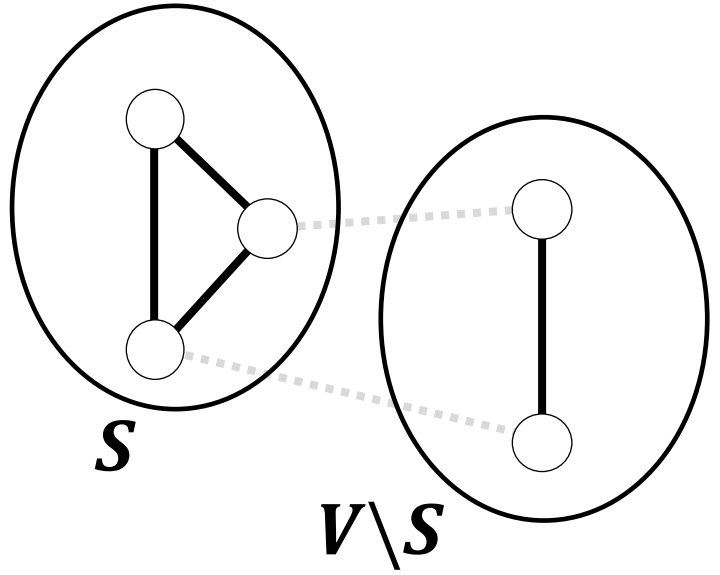
Proof:



MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

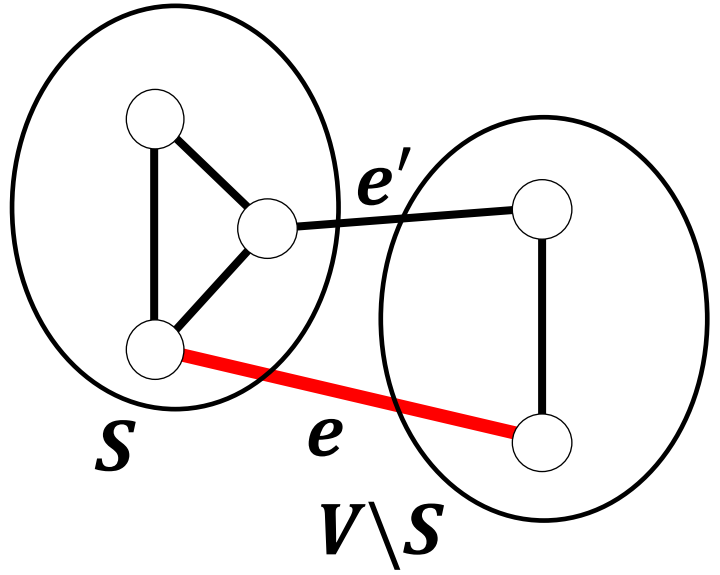


MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let $\boldsymbol{e}$ be the cheapest edge between S and $V \backslash S$.



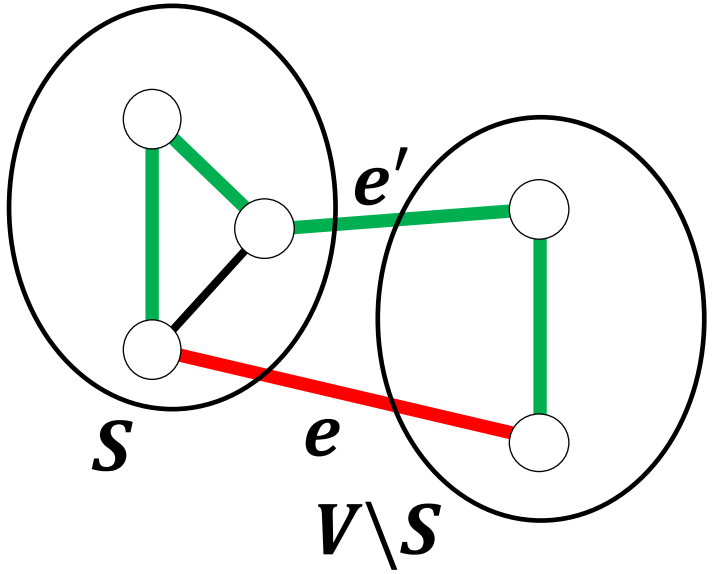
MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let $\boldsymbol{e}$ be the cheapest edge between S and $V \backslash S$.

Suppose $\boldsymbol{T}$ is the MST that does not include $\boldsymbol{e}$.



MST Cut Property

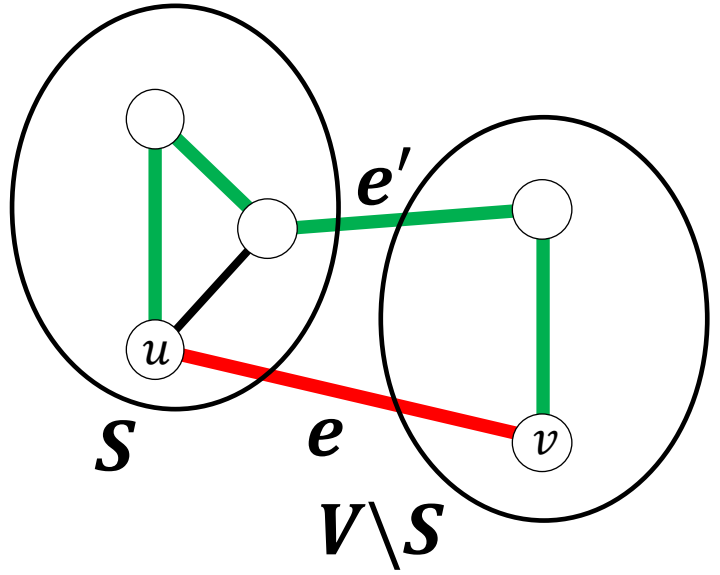
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let $\boldsymbol{e}$ be the cheapest edge between S and $V \backslash S$.

Suppose $\boldsymbol{T}$ is the MST that does not include $\boldsymbol{e}$. Then:

1. $\boldsymbol{T} \cup \{\boldsymbol{e}\}$ must have a cycle. Because?



MST Cut Property

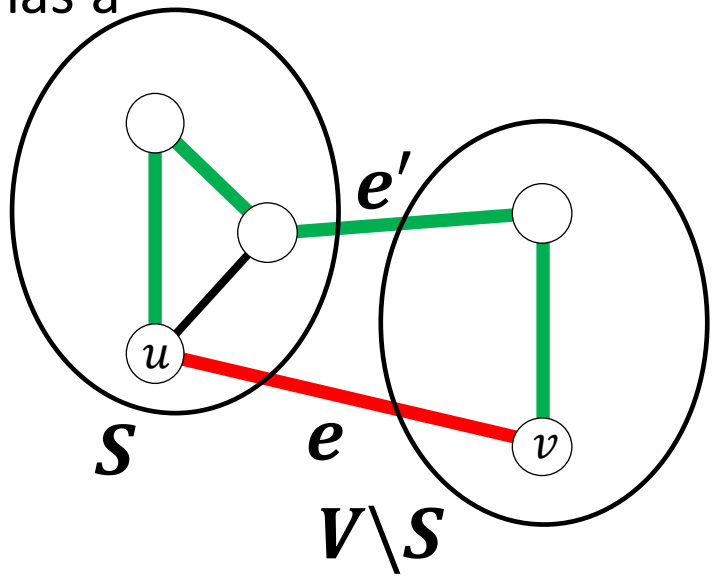
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let $\boldsymbol{e}$ be the cheapest edge between S and $V \backslash S$.

Suppose $\boldsymbol{T}$ is the MST that does not include $\boldsymbol{e}$. Then:

1. $\boldsymbol{T} \cup \{\boldsymbol{e}\}$ must have a cycle. (Since spanning tree $\boldsymbol{T}$ already has a path between u and v, adding $\boldsymbol{e}$ will create a cycle.)



MST Cut Property

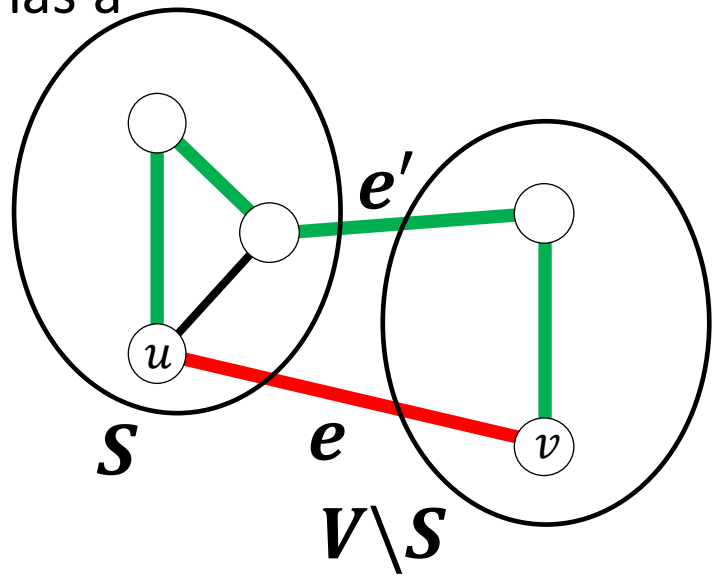
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let $\boldsymbol{e}$ be the cheapest edge between S and $V \backslash S$.

Suppose $\boldsymbol{T}$ is the MST that does not include $\boldsymbol{e}$. Then:

1. $\boldsymbol{T} \cup \{\boldsymbol{e}\}$ must have a cycle. (Since spanning tree $\boldsymbol{T}$ already has a path between u and v, adding $\boldsymbol{e}$ will create a cycle.)
2. That cycle must have another edge e' between S and $V \backslash S$. (Since there must be a path from $u \in S$ to $v \in V \backslash S$ in $\boldsymbol{T}$)



MST Cut Property

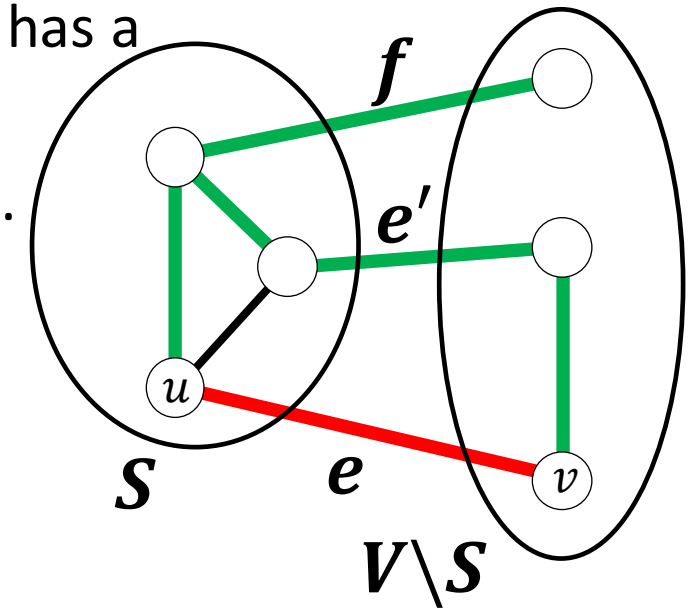
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let $\boldsymbol{e}$ be the cheapest edge between S and $V \backslash S$.

Suppose $\boldsymbol{T}$ is the MST that does not include $\boldsymbol{e}$. Then:

1. $\boldsymbol{T} \cup \{\boldsymbol{e}\}$ must have a cycle. (Since spanning tree $\boldsymbol{T}$ already has a path between u and v, adding $\boldsymbol{e}$ will create a cycle.)
2. That cycle must have another edge e' between S and $V \backslash S$. (Since there must be a path from $u \in S$ to $v \in V \backslash S$ in $\boldsymbol{T}$)



Need to make sure we pick an edge between S and $V \backslash S$ on the cycle!

MST Cut Property

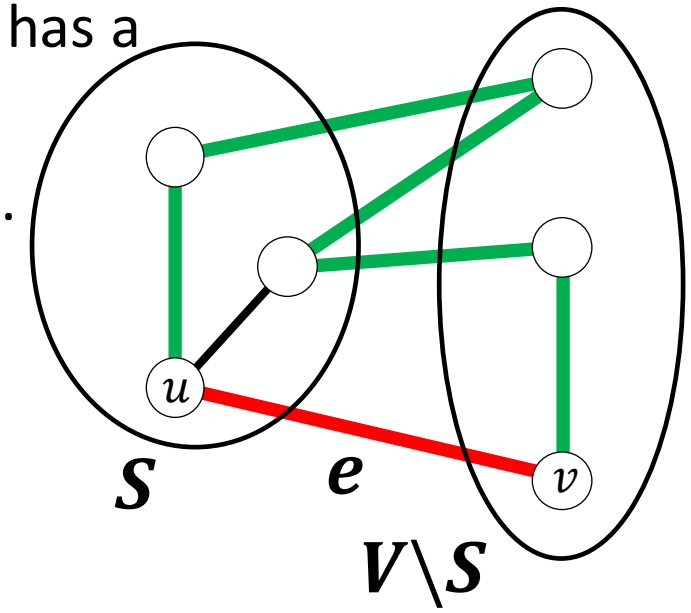
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let $\boldsymbol{e}$ be the cheapest edge between S and $V \backslash S$.

Suppose $\boldsymbol{T}$ is the MST that does not include $\boldsymbol{e}$. Then:

1. $\boldsymbol{T} \cup \{\boldsymbol{e}\}$ must have a cycle. (Since spanning tree $\boldsymbol{T}$ already has a path between u and v, adding $\boldsymbol{e}$ will create a cycle.)
2. That cycle must have another edge e' between S and $V \backslash S$. (Since there must be a path from $u \in S$ to $v \in V \backslash S$ in $\boldsymbol{T}$)



Need to make sure we pick an edge between S and $V \backslash S$ on the cycle!
(Which one doesn't matter.)

MST Cut Property

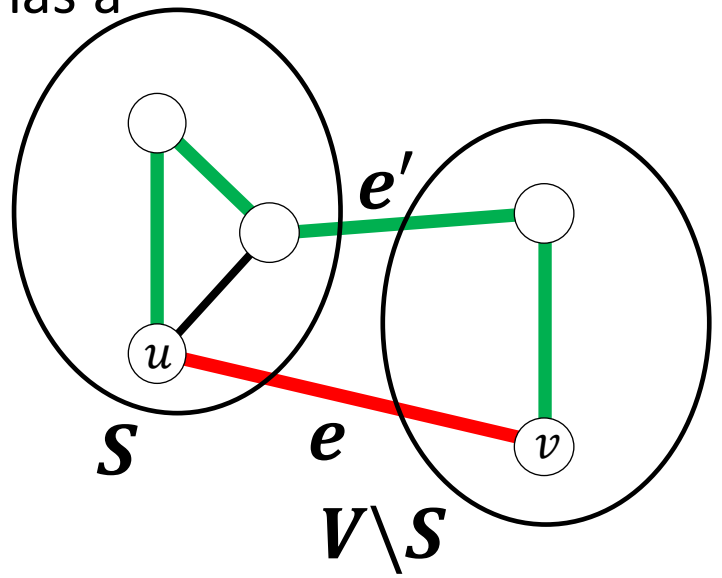
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let $\boldsymbol{e}$ be the cheapest edge between S and $V \backslash S$.

Suppose $\boldsymbol{T}$ is the MST that does not include $\boldsymbol{e}$. Then:

1. $\boldsymbol{T} \cup \{\boldsymbol{e}\}$ must have a cycle. (Since spanning tree $\boldsymbol{T}$ already has a path between u and v, adding $\boldsymbol{e}$ will create a cycle.)
2. That cycle must have another edge e' between S and $V \backslash S$. (Since there must be a path from $u \in S$ to $v \in V \backslash S$ in $\boldsymbol{T}$)



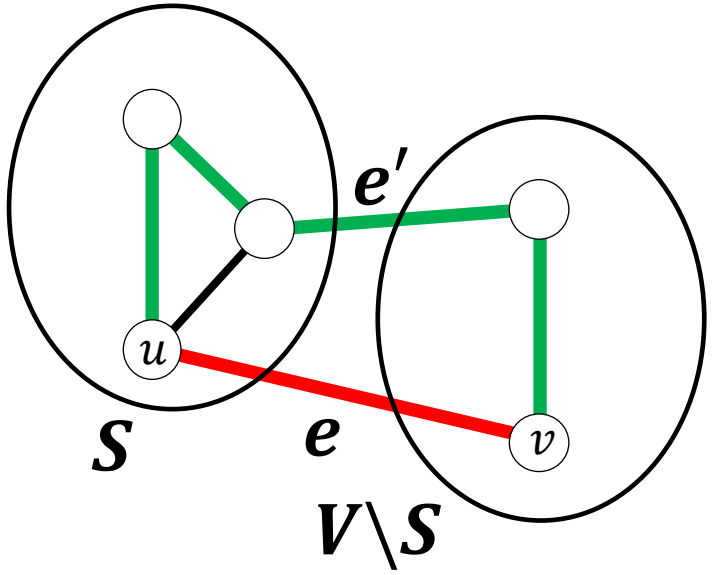
MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let $\boldsymbol{e}$ be the cheapest edge between S and $V \backslash S$.

Suppose $\boldsymbol{T}$ is the MST that does not include $\boldsymbol{e}$. $\boldsymbol{T} \cup \{\boldsymbol{e}\}$ must have a cycle and that cycle must have another edge e' between S and $V \backslash S$.



MST Cut Property

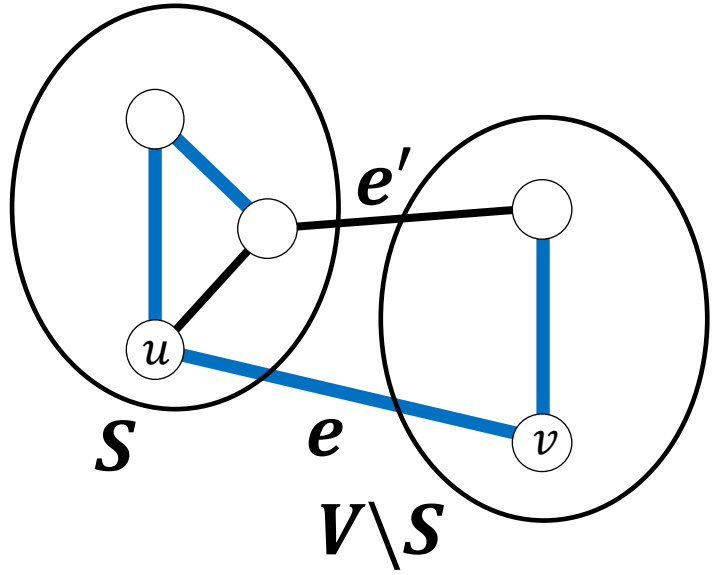
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let e be the cheapest edge between S and $V \backslash S$.

Suppose T is the MST that does not include e. $T \cup \{e\}$ must have a cycle and that cycle must have another edge e' between S and $V \backslash S$.

Swap e and e' to form $\boldsymbol{T' = T \cup \{e\} \backslash \{e'\}}$.



MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

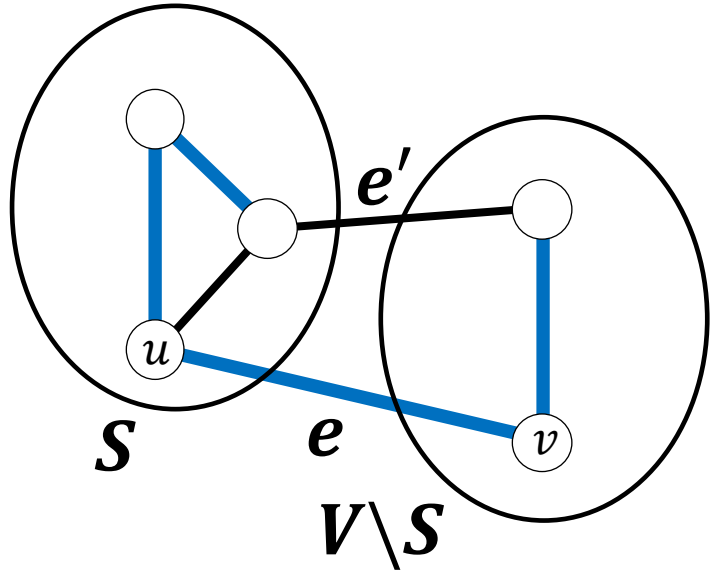
Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let e be the cheapest edge between S and $V \backslash S$.

Suppose T is the MST that does not include e. $T \cup \{e\}$ must have a cycle and that cycle must have another edge e' between S and $V \backslash S$.

Swap e and e' to form $\boldsymbol{T' = T \cup \{e\} \backslash \{e'\}}$.

$\boldsymbol{T'}$ is a cheaper spanning tree because:



MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

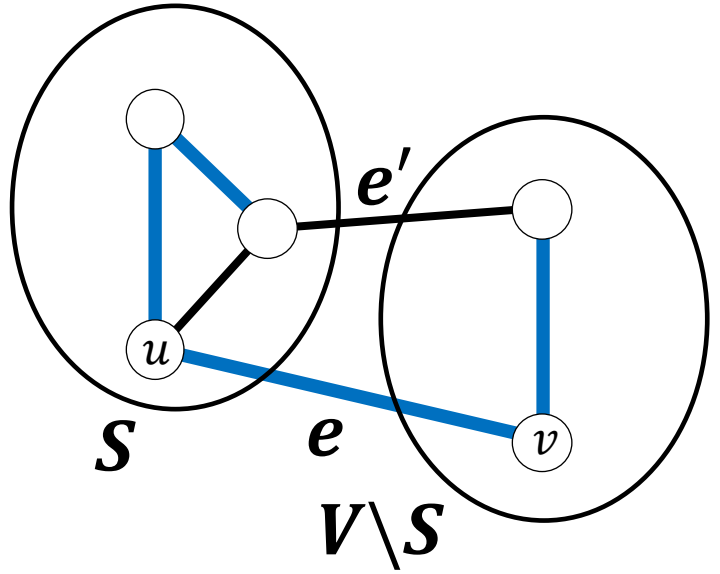
Let e be the cheapest edge between S and $V \backslash S$.

Suppose T is the MST that does not include e. $T \cup \{e\}$ must have a cycle and that cycle must have another edge e' between S and $V \backslash S$.

Swap e and e' to form $\boldsymbol{T' = T \cup \{e\} \backslash \{e'\}}$.

$\boldsymbol{T'}$ is a cheaper spanning tree because:

- $\boldsymbol{T'}$ is a tree (breaking cycle doesn't disconnect graph)



MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

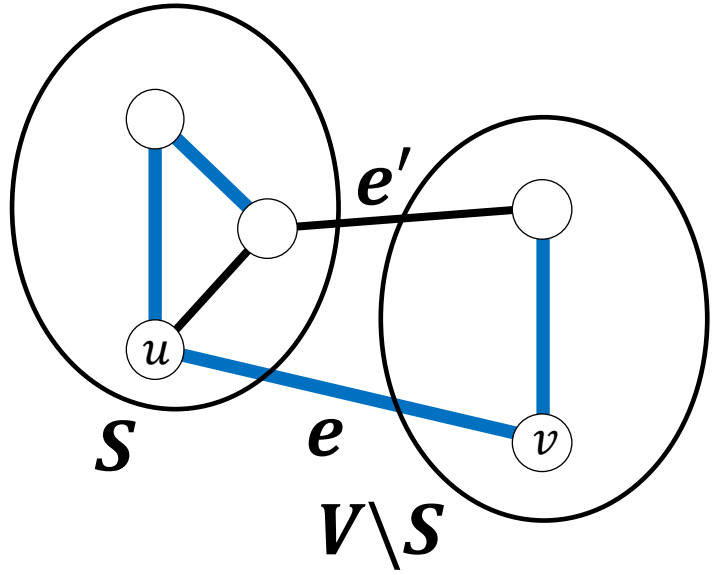
Let e be the cheapest edge between S and $V \backslash S$.

Suppose T is the MST that does not include e. $T \cup \{e\}$ must have a cycle and that cycle must have another edge e' between S and $V \backslash S$.

Swap e and e' to form $\boldsymbol{T' = T \cup \{e\} \backslash \{e'\}}$.

$\boldsymbol{T'}$ is a cheaper spanning tree because:

- $\boldsymbol{T'}$ is a tree (breaking cycle doesn't disconnect graph)
- $\boldsymbol{T'}$ spans V (same number of edges as spanning tree T)



MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

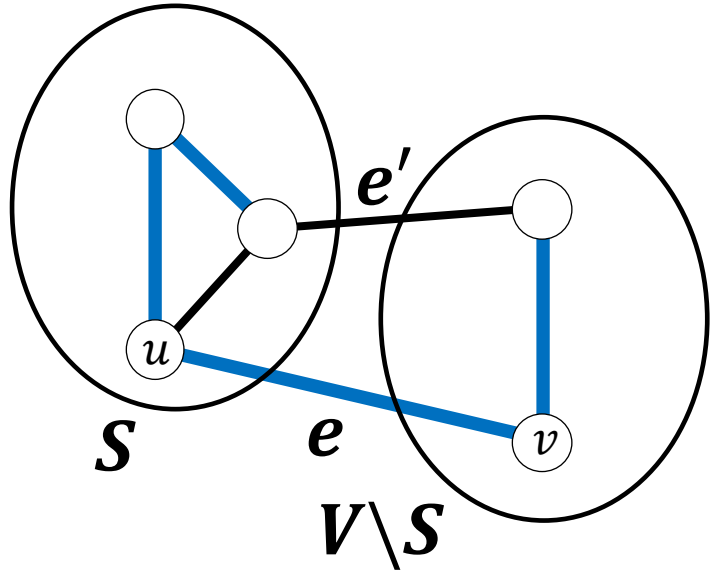
Let e be the cheapest edge between S and $V \backslash S$.

Suppose T is the MST that does not include e. $T \cup \{e\}$ must have a cycle and that cycle must have another edge e' between S and $V \backslash S$.

Swap e and e' to form $\boldsymbol{T' = T \cup \{e\} \backslash \{e'\}}$.

$\boldsymbol{T'}$ is a cheaper spanning tree because:

- $\boldsymbol{T'}$ is a tree (breaking cycle doesn't disconnect graph)
- $\boldsymbol{T'}$ spans V (same number of edges as spanning tree T)
- cost($\boldsymbol{T'}$) < cost(T) since e' was replaced by the cheaper e.



MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let e be the cheapest edge between S and $V \backslash S$.

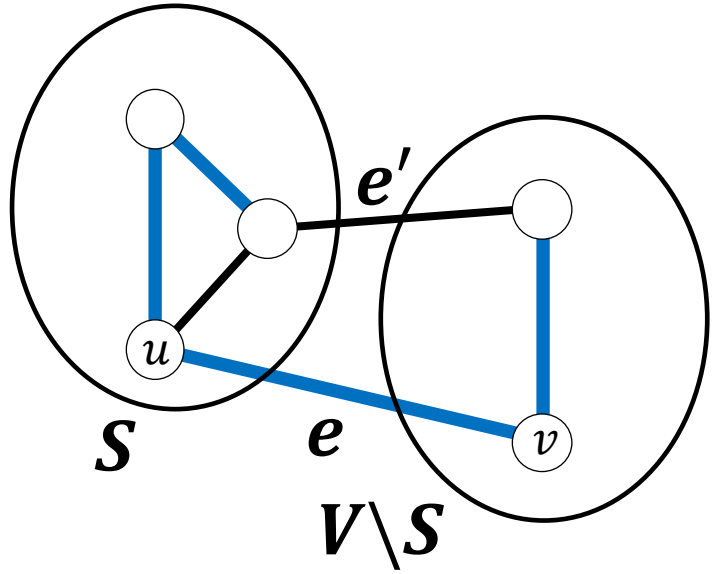
Suppose T is the MST that does not include e. $T \cup \{e\}$ must have a cycle and that cycle must have another edge e' between S and $V \backslash S$.

Swap e and e' to form $\boldsymbol{T' = T \cup \{e\} \backslash \{e'\}}$.

$\boldsymbol{T'}$ is a cheaper spanning tree because:

- $\boldsymbol{T'}$ is a tree (breaking cycle doesn't disconnect graph)
- $\boldsymbol{T'}$ spans V (same number of edges as spanning tree T)
- cost($\boldsymbol{T'}$) < cost(T) since e' was replaced by the cheaper e.

Thus, $\boldsymbol{T'}$ is a cheaper spanning tree



MST Cut Property

Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

Let e be the cheapest edge between S and $V \backslash S$.

Suppose T is the MST that does not include e. $T \cup \{e\}$ must have a cycle and that cycle must have another edge e' between S and $V \backslash S$.

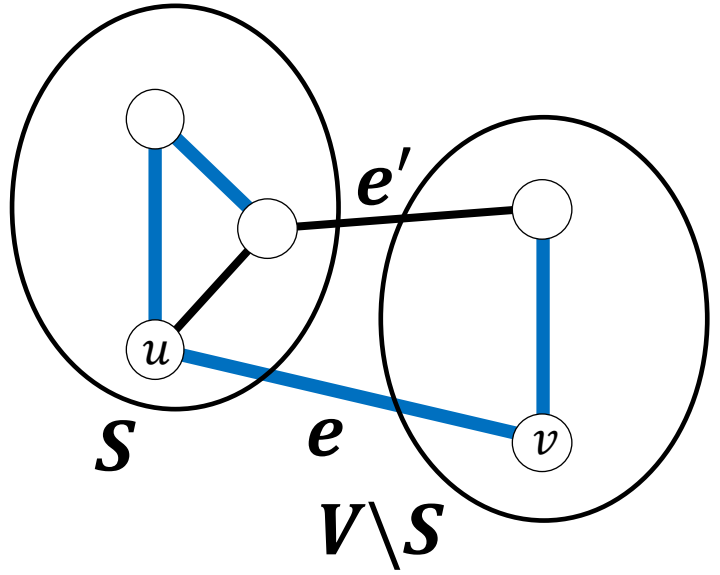
Swap e and e' to form $\boldsymbol{T' = T \cup \{e\} \backslash \{e'\}}$.

$\boldsymbol{T'}$ is a cheaper spanning tree because:

- $\boldsymbol{T'}$ is a tree (breaking cycle doesn't disconnect graph)
- $\boldsymbol{T'}$ spans V (same number of edges as spanning tree T)
- cost($\boldsymbol{T'}$) < cost(T) since e' was replaced by the cheaper e.

Thus, $\boldsymbol{T'}$ is a cheaper spanning tree

⇒ The MST must include e.



Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Proof of optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Kruskal's algorithm.

How do we use the Cut Property to show that Kruskal's is optimal?

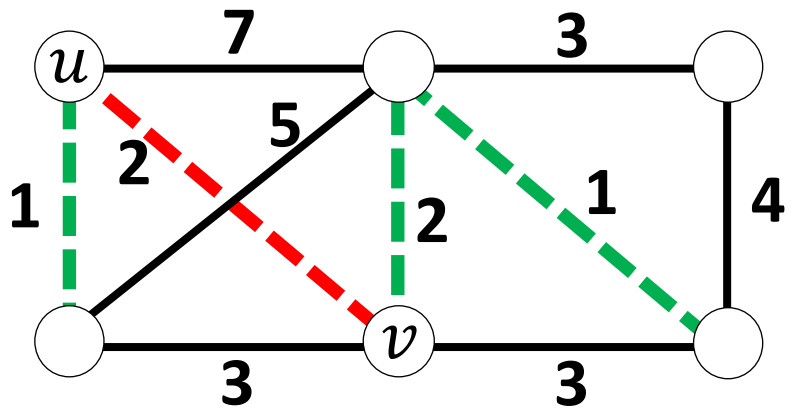
Lemma: The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Proof of optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Kruskal's algorithm.

Consider the iteration that some edge $\boldsymbol{e} = (u, v)$ is added by Kruskal's algorithm.



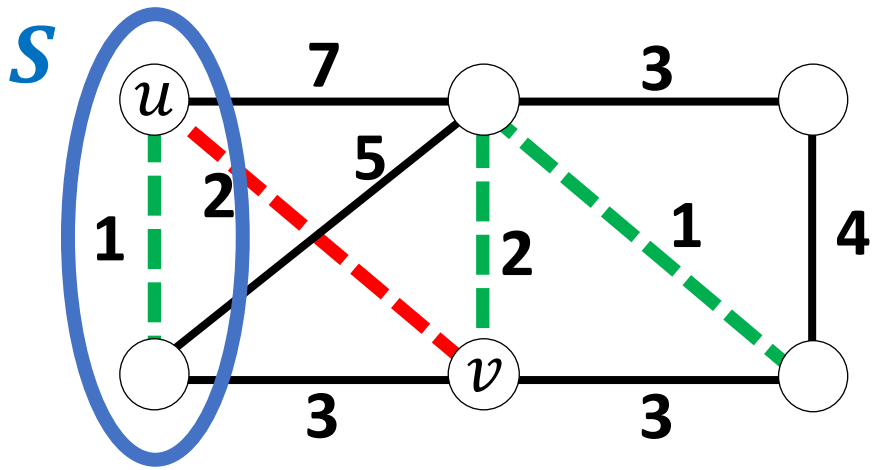
Lemma: The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Proof of optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Kruskal's algorithm.

Consider the iteration that some edge $\boldsymbol{e} = (u, v)$ is added by Kruskal's algorithm. Let $\boldsymbol{S}$ be the set of u and all nodes already connected to u. (or v and all nodes connected to v)



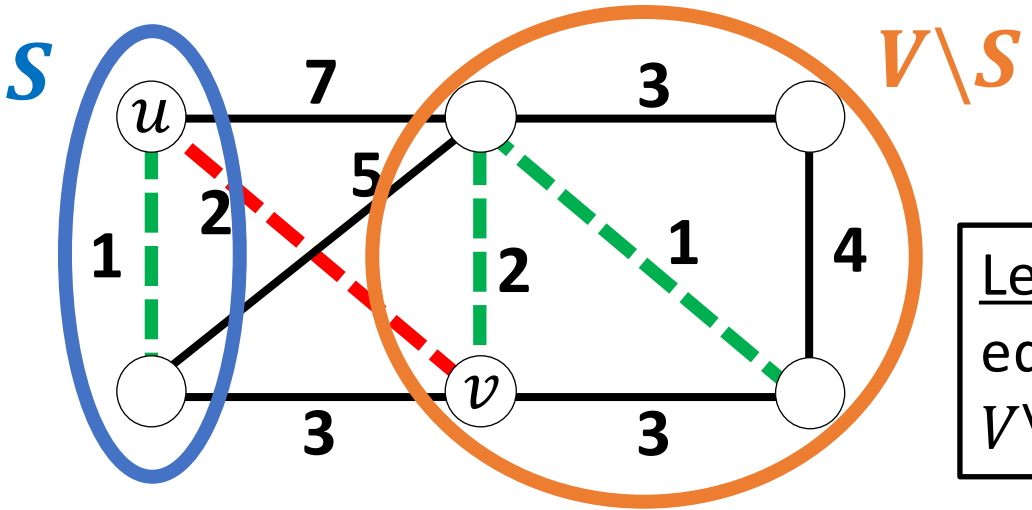
Lemma: The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Proof of optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Kruskal's algorithm.

Consider the iteration that some edge $\boldsymbol{e} = (u, v)$ is added by Kruskal's algorithm. Let $\boldsymbol{S}$ be the set of u and all nodes already connected to u. Clearly $u \in \boldsymbol{S}$ and $v \in \boldsymbol{V \backslash S}$ (otherwise adding $\boldsymbol{e}$ would have created a cycle).



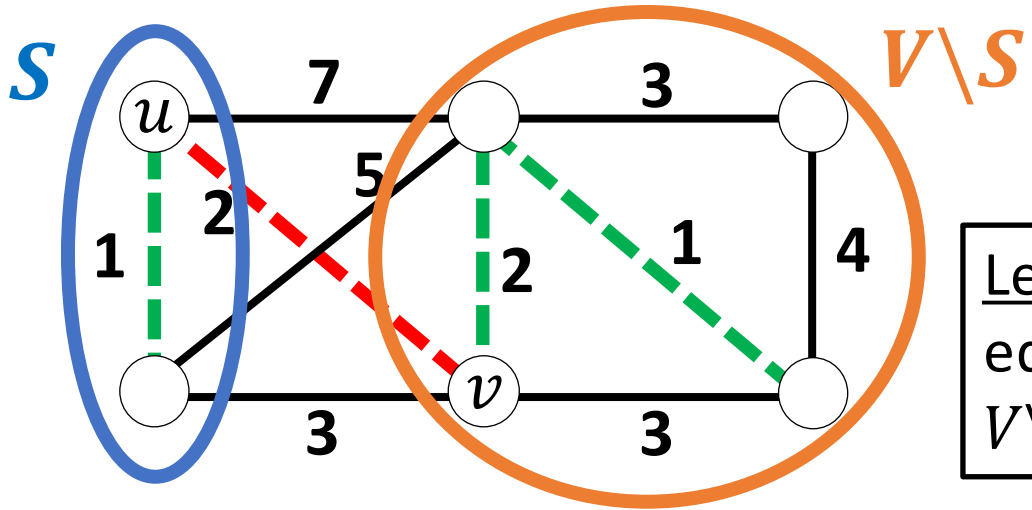
Lemma: The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Proof of optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Kruskal's algorithm.

Consider the iteration that some edge $\boldsymbol{e} = (u, v)$ is added by Kruskal's algorithm. Let $\boldsymbol{S}$ be the set of u and all nodes already connected to u. Clearly $u \in \boldsymbol{S}$ and $v \in \boldsymbol{V \backslash S}$ (otherwise adding $\boldsymbol{e}$ would have created a cycle). We are picking the cheapest edge that crosses the cut (otherwise the cheaper edge would have been selected since it would not have created a cycle either).



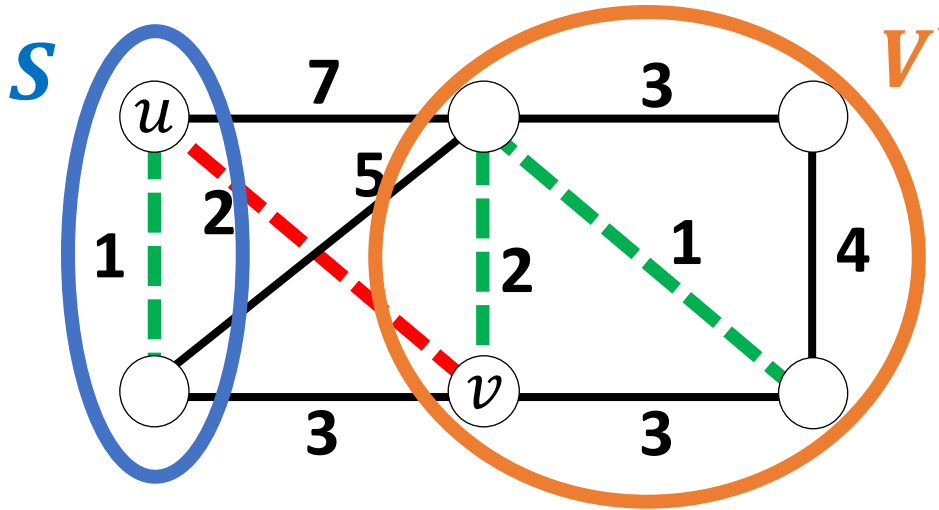
Lemma: The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Proof of optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Kruskal's algorithm.

Consider the iteration that some edge $\boldsymbol{e} = (u, v)$ is added by Kruskal's algorithm. Let $\boldsymbol{S}$ be the set of u and all nodes already connected to u. Clearly $u \in \boldsymbol{S}$ and $v \in \boldsymbol{V \backslash S}$ (otherwise adding $\boldsymbol{e}$ would have created a cycle). We are picking the cheapest edge that crosses the cut (otherwise the cheaper edge would have been selected since it would not have created a cycle either). By the cut property lemma, this edge must be part of the MST.



Lemma: The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

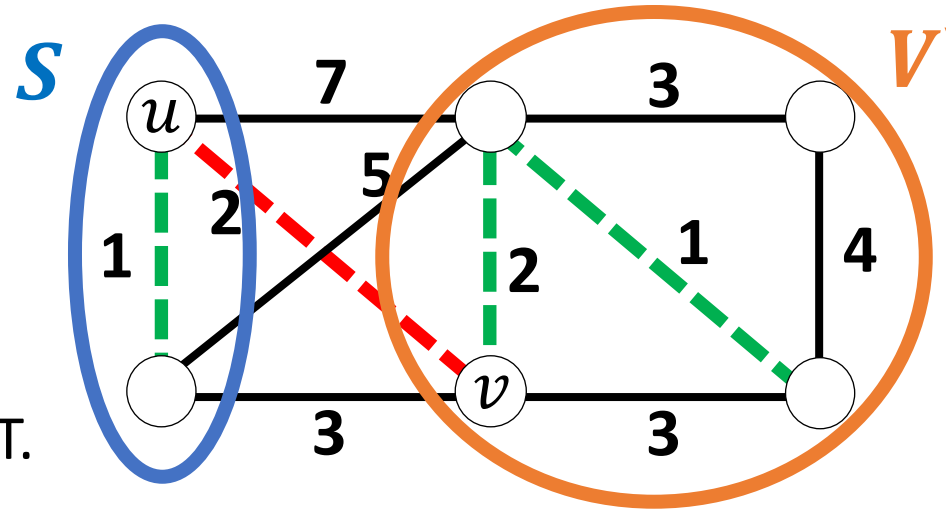
Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Proof of optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Kruskal's algorithm.

Consider the iteration that some edge $\boldsymbol{e} = (u, v)$ is added by Kruskal's algorithm. Let $\boldsymbol{S}$ be the set of u and all nodes already connected to u. Clearly $u \in \boldsymbol{S}$ and $v \in \boldsymbol{V \backslash S}$ (otherwise adding $\boldsymbol{e}$ would have created a cycle). We are picking the cheapest edge that crosses the cut (otherwise the cheaper edge would have been selected since it would not have created a cycle either). By the cut property lemma, this edge must be part of the MST.

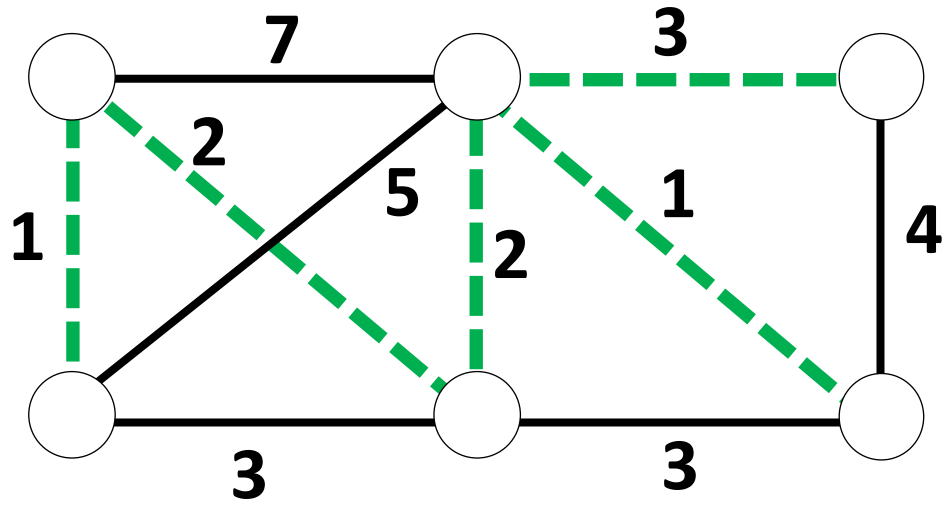
Thus, every edge found by Kruskal's algorithm is part of the MST, and since the edges found form a spanning tree, it is the MST.



Lemma: The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.



Basic algorithm analysis questions:

1. ~~Is the solution valid? (Does it actually find a spanning tree?)~~
2. ~~Is the solution optimal? (Does it find the best spanning tree?)~~
3. What is the running time?

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Running Time:

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Running Time:

```
findMST(G=(V,E)) {
  T = ∅
  sort(E) //smallest to largest weight
  for (e in E) {
    if (T U {e} is acyclic) {
      T = T U {e}
    }
  }
  return T
}
```

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Running Time:

```
findMST(G=(V,E)) {
  T = ∅
  sort(E) //smallest to largest weight
  for (e in E) {
    if (T U {e} is acyclic) {
      T = T U {e}
    }
  }
  return T
}
```

$\leftarrow O(|E| \log(|E|))$

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Running Time:

```
findMST(G=(V,E)) {
  T = ∅
  sort(E) //smallest to largest weight ← O(|E| log(|E|))
  for (e in E) { ← O(|E|)
    if (T U {e} is acyclic) {
      T = T U {e}
    }
  }
  return T
}
```

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Running Time:

```
findMST(G=(V,E)) {
  T = ∅
  sort(E) //smallest to largest weight ← O(|E| log(|E|))
  for (e in E) { ← O(|E|)
    if (T U {e} is acyclic) { ← O(|V| + |E|) using BFS
      T = T U {e}
    }
  }
  return T
}
```

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Running Time:

```
findMST(G=(V,E)) {
  T = ∅
  sort(E) //smallest to largest weight
  for (e in E) {
    if (T U {e} is acyclic) {
      T = T U {e}
    }
  }
  return T
}
```

$\leftarrow O(|E| \log(|E|))$

$\leftarrow O(|E|)$

$\leftarrow O(|V| + |E|)$ **using BFS**

Running time

$$\in O\big(|E| \log(|E|) + |E|(|V| + |E|)\big)$$

$$\in O\big(|E|^2 + |E||V|\big)$$

Kruskal's MST Algorithm

Algorithm: Add the edge with smallest weight, that does not create a cycle.

Running Time:

```
findMST(G=(V,E)) {
  T = ∅
  sort(E) //smallest to largest weight
  for (e in E) {
    if (T U {e} is acyclic) {
      T = T U {e}
    }
  }
  return T
}
```

Can be improved to $O(1)$, thus $O(|E| \log(|E|))$ overall

$O(|E| \log(|E|))$ ← sort(E) //smallest to largest weight

$O(|E|)$ ← for (e in E) {

$O(|V| + |E|)$ **using BFS** ← if (T U {e} is acyclic) {

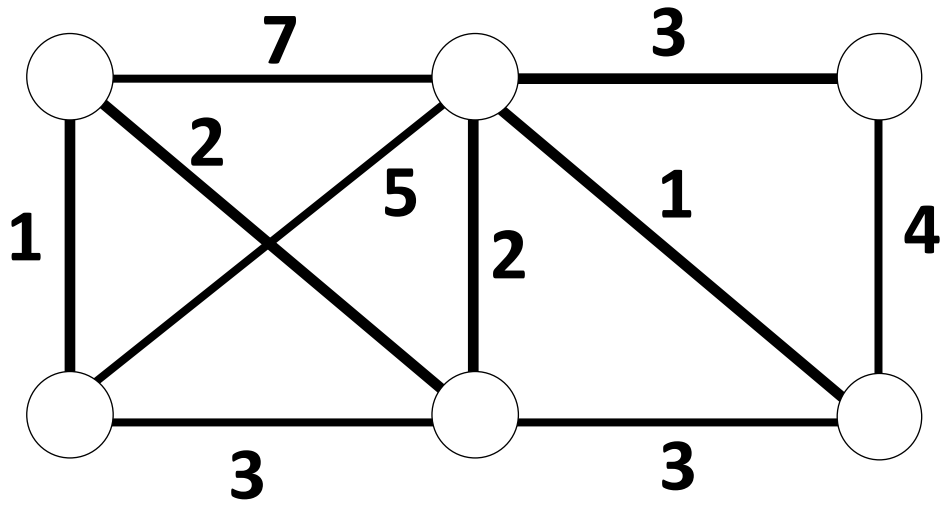
Running time

$$\in O\big(|E| \log(|E|) + |E|(|V| + |E|)\big)$$

$$\in O\big(|E|^2 + |E||V|\big)$$

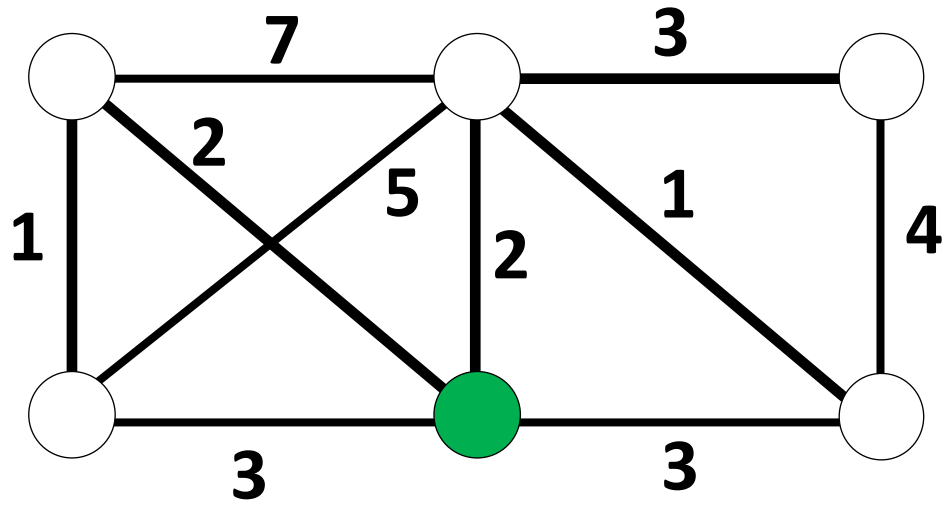
Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.



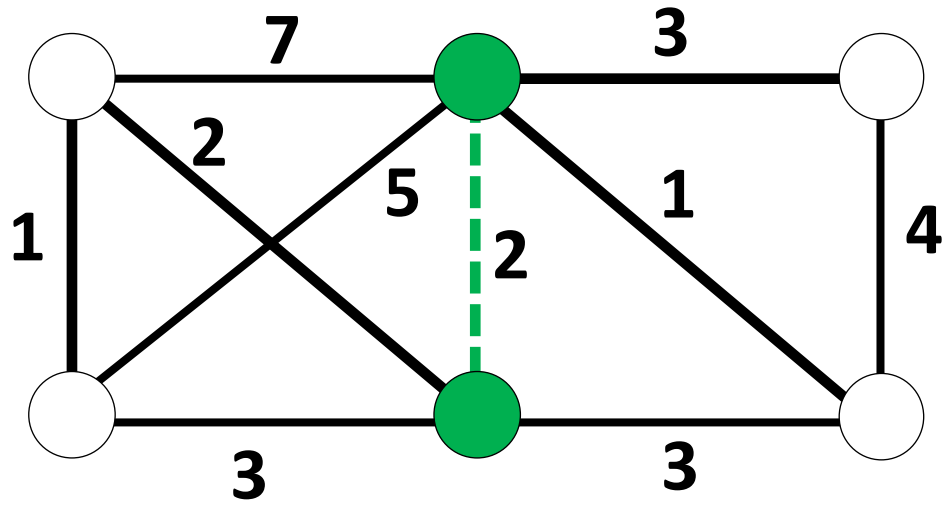
Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.



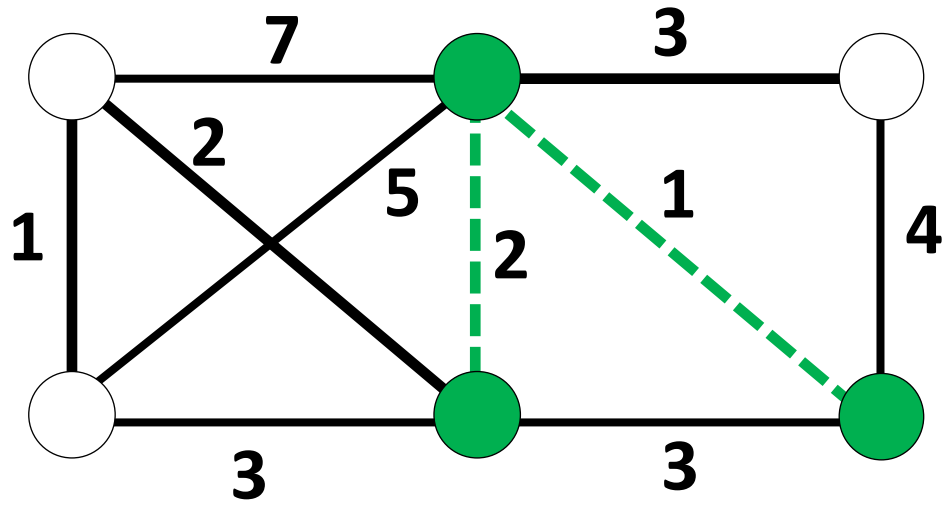
Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.



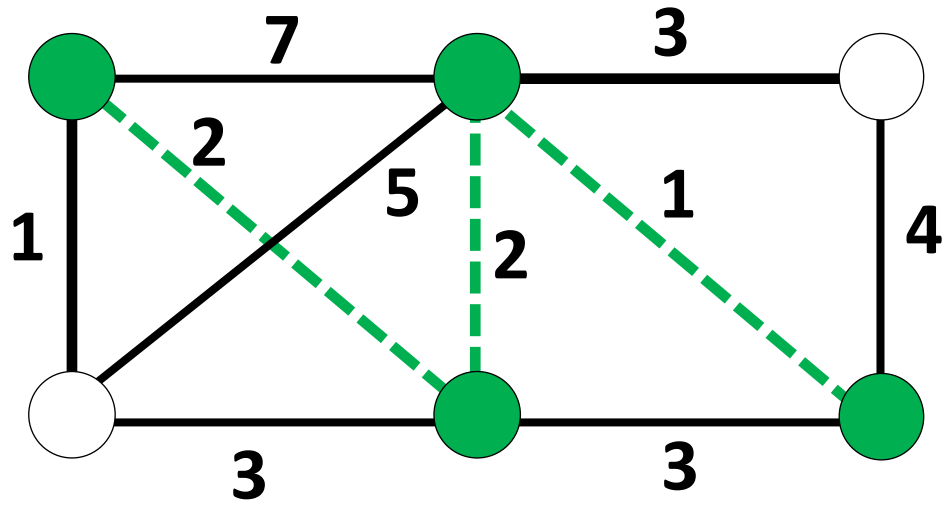
Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.



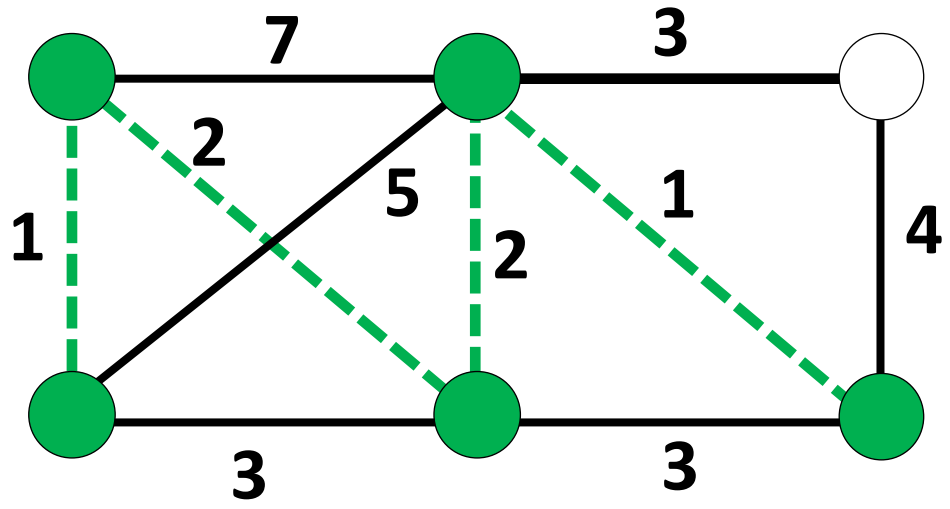
Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.



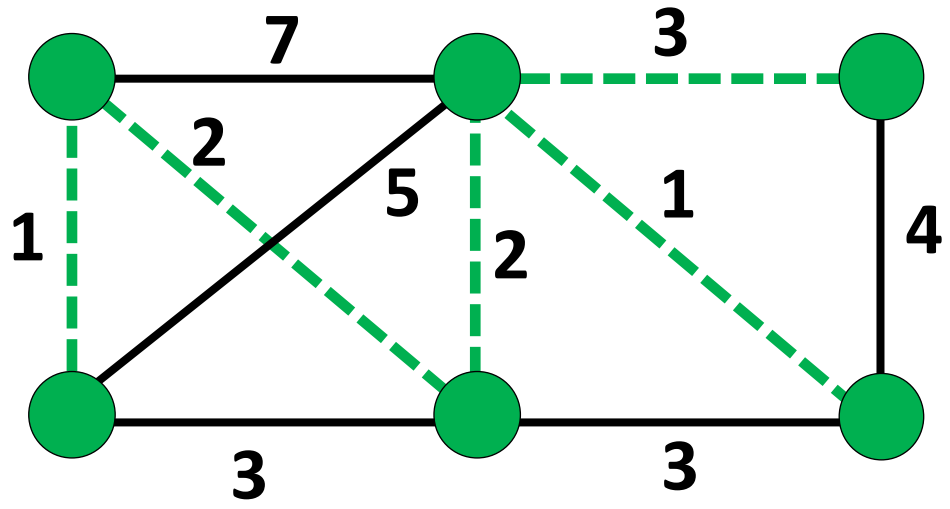
Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.



Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.

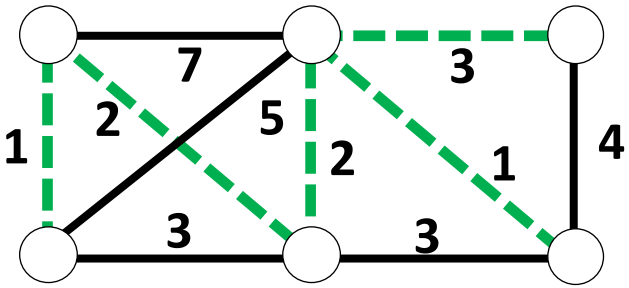


Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.

Proof of validity & optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Prim's algorithm.

T is a tree because ??



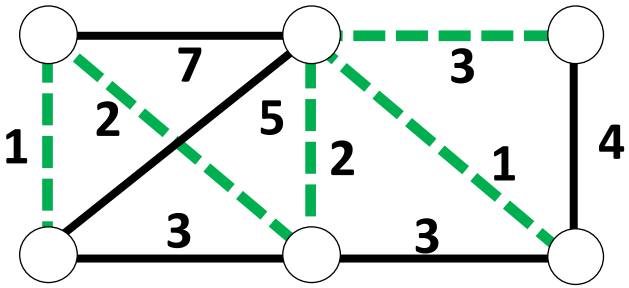
Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.

Proof of validity & optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Prim's algorithm.

T is a tree because it consists of a single connected component and we never add an additional edge to an already connected node (i.e. no cycles).

T is a spanning tree because ??



Prim's MST Algorithm

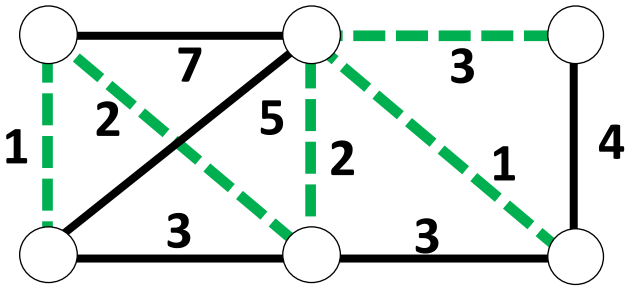
Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.

Proof of validity & optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Prim's algorithm.

T is a tree because it consists of a single connected component and we never add an additional edge to an already connected node (i.e. no cycles).

T is a spanning tree because we added nodes (edges) until there were no nodes left to add.

T is an MST because ??



Prim's MST Algorithm

Algorithm: Beginning at any node, add the node that can be connected as cheaply as possible to the tree we are building.

Proof of validity & optimality: Let $G = (V, E)$, and $T \subseteq E$ be the set of edges resulting from Prim's algorithm.

T is a tree because it consists of a single connected component and we never add an additional edge to an already connected node (i.e. no cycles).

T is a spanning tree because we added nodes (edges) until there were no nodes left to add.

T is an MST because at each step we add the cheapest edge between the current tree and the remaining nodes (cut property).