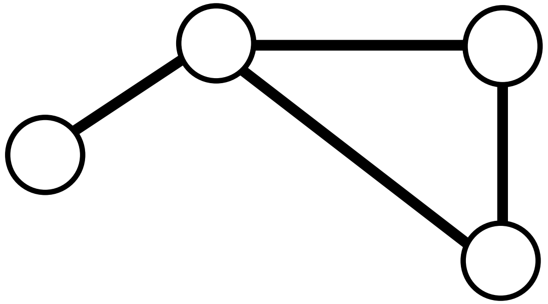
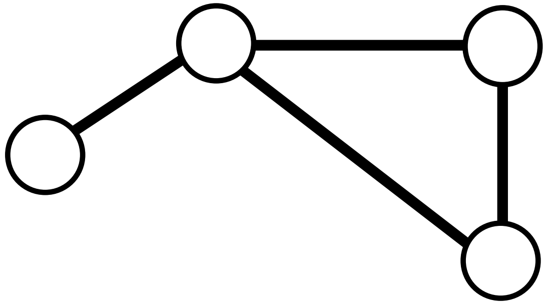
Minimum Spanning Trees
CSCI 532

Minimum Spanning Tree (MST)



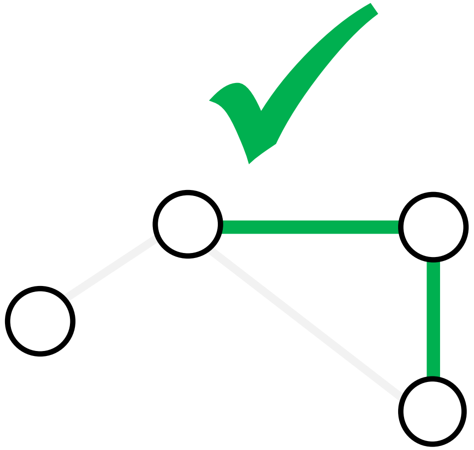
Given a connected graph, a subset of edges is a…

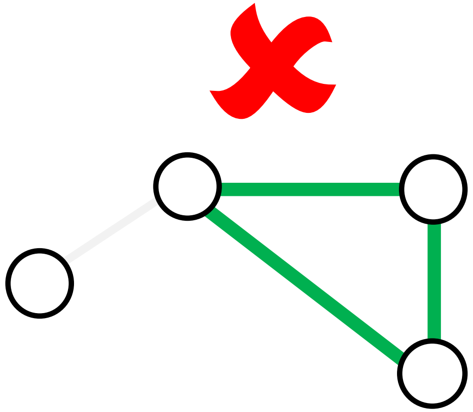
Minimum Spanning Tree (MST)



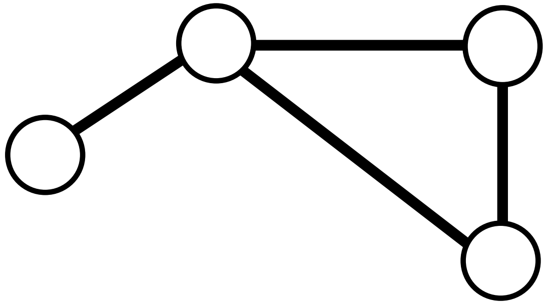
Given a connected graph, a subset of edges is a…

Tree if it is connected and acyclic.

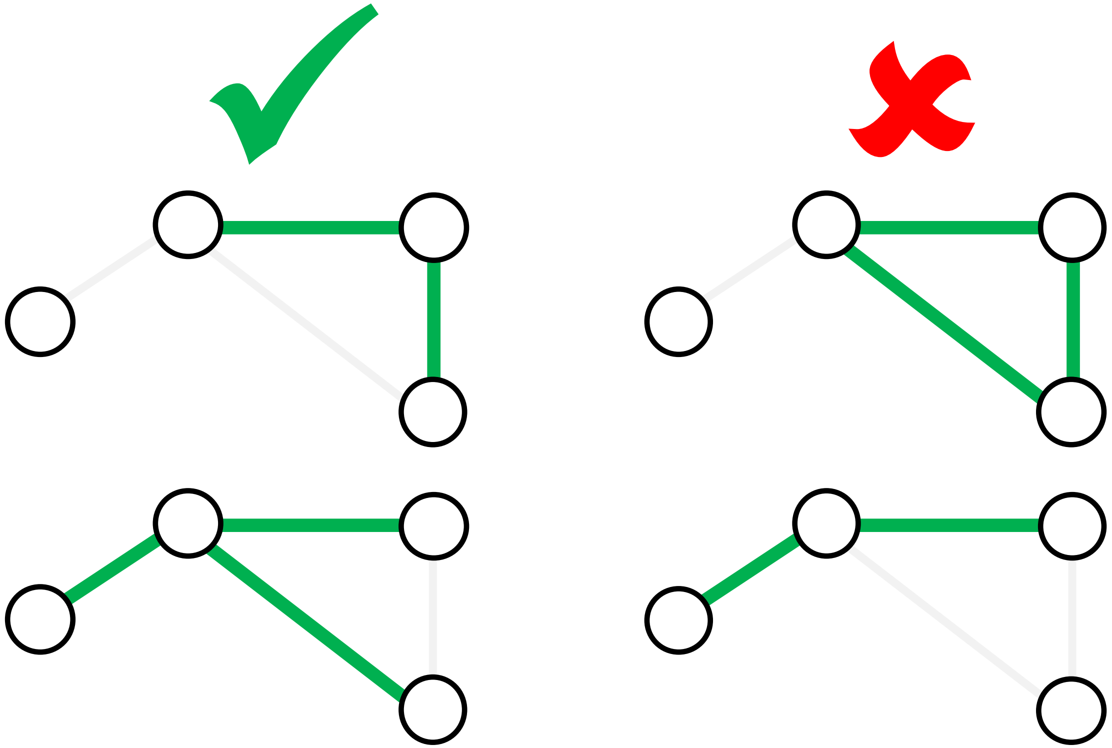




Minimum Spanning Tree (MST)



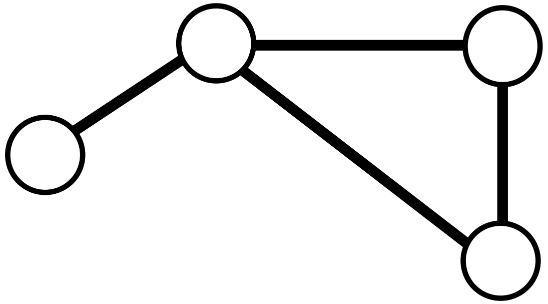
Given a connected graph, a subset of edges is a…



Tree if it is connected and acyclic.

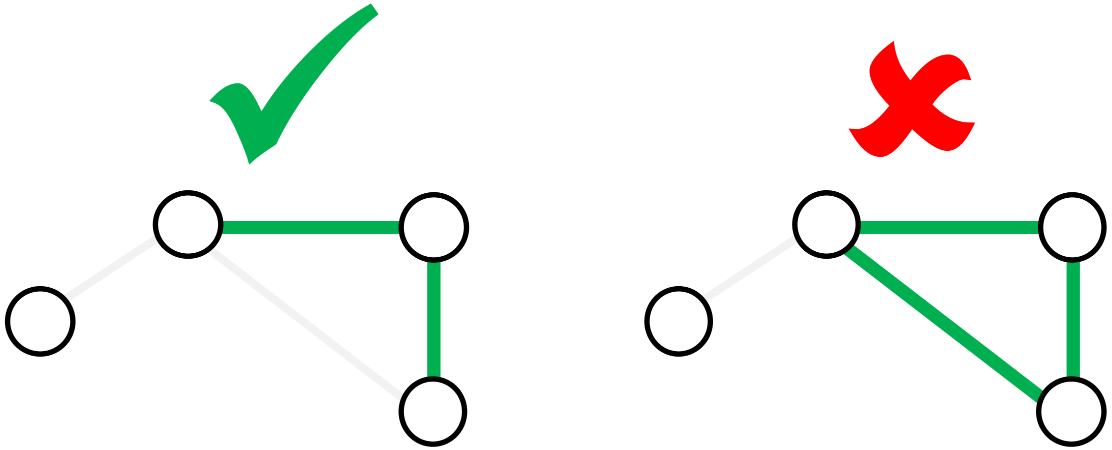
Spanning tree if it is a tree and includes all vertices in the graph.

Minimum Spanning Tree (MST)

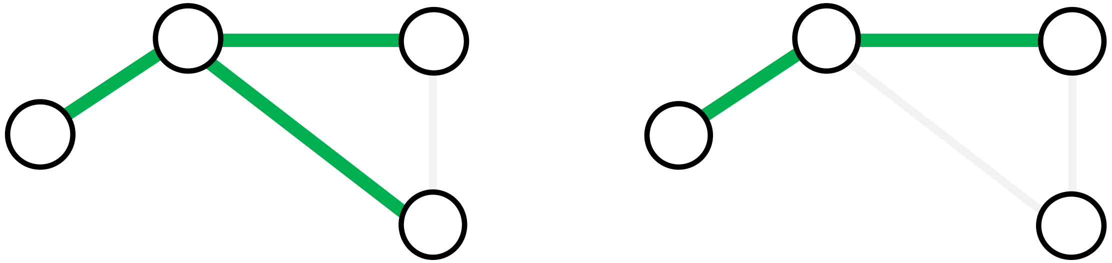


Given a connected graph, a subset of edges is a…

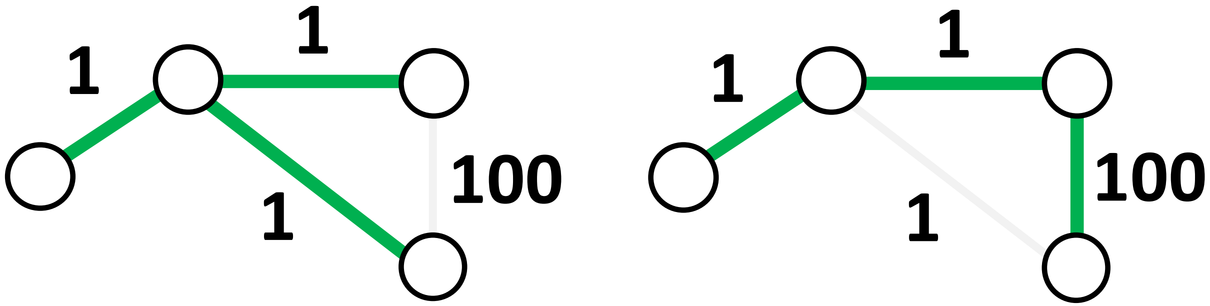
Tree if it is connected and acyclic.



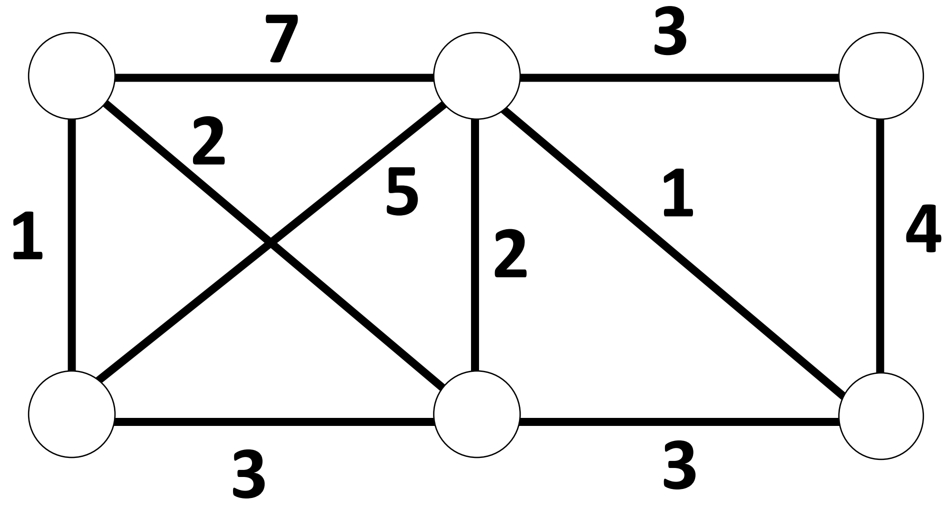
Spanning tree if it is a tree and includes all vertices in the graph.



Minimum spanning tree if it is a spanning tree whose sum of edge costs is the minimum possible value.

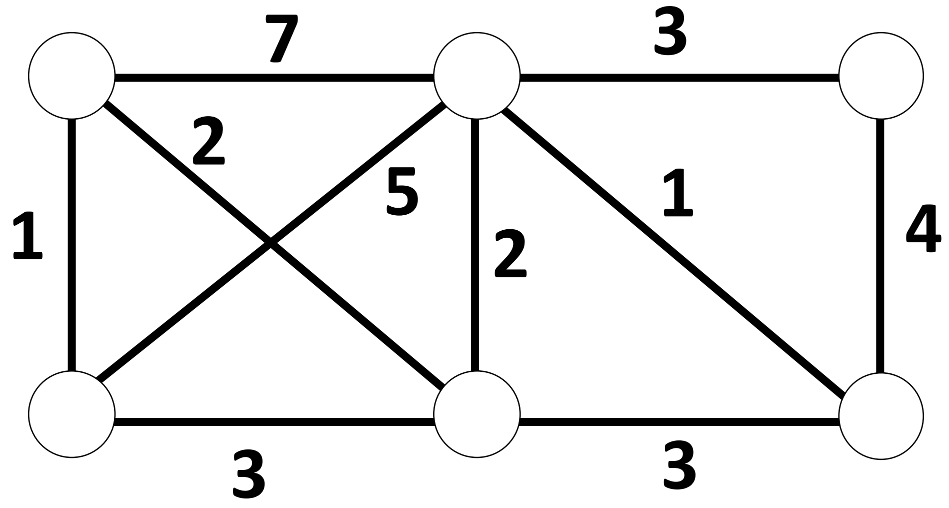


MST Problem



Goal: Given a connected, edge weighted graph, find its Minimum Spanning Tree.

MST Problem

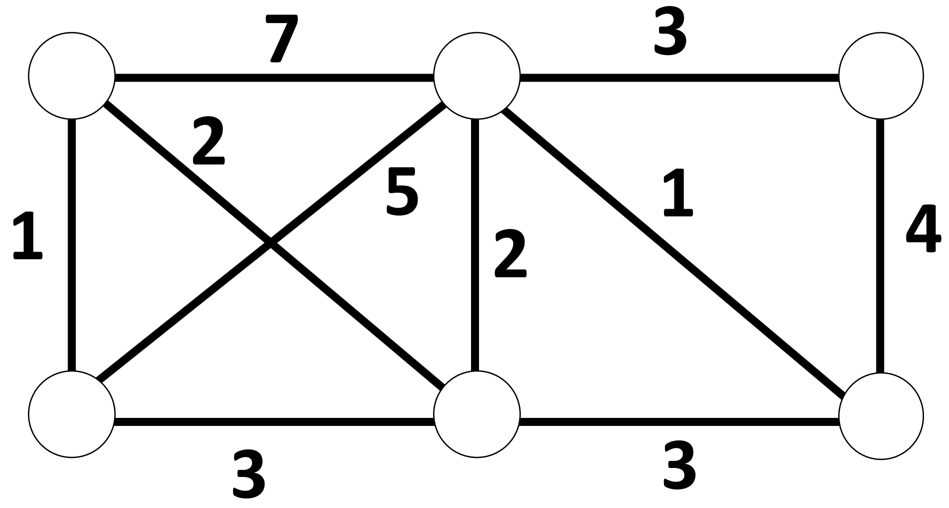


Greedy Algorithms:

- Make the choice that best helps some objective.
- Do not look ahead, plan, or revisit past decisions.
- Hope that optimal local choices lead to optimal global solutions.

Kruskal's MST Algorithm

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

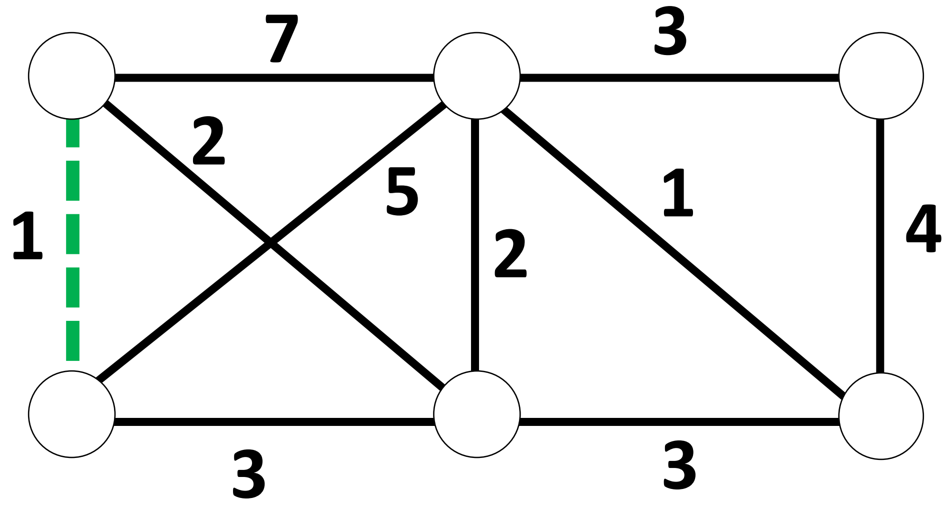


Greedy Algorithms:

- Make the choice that best helps some objective.
- Do not look ahead, plan, or revisit past decisions.
- Hope that optimal local choices lead to optimal global solutions.

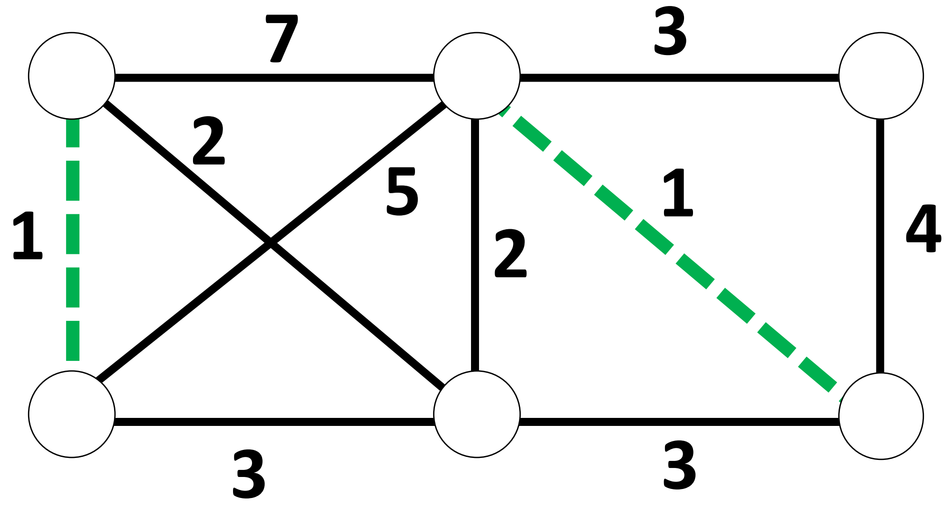
Kruskal’s MST Algorithm

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



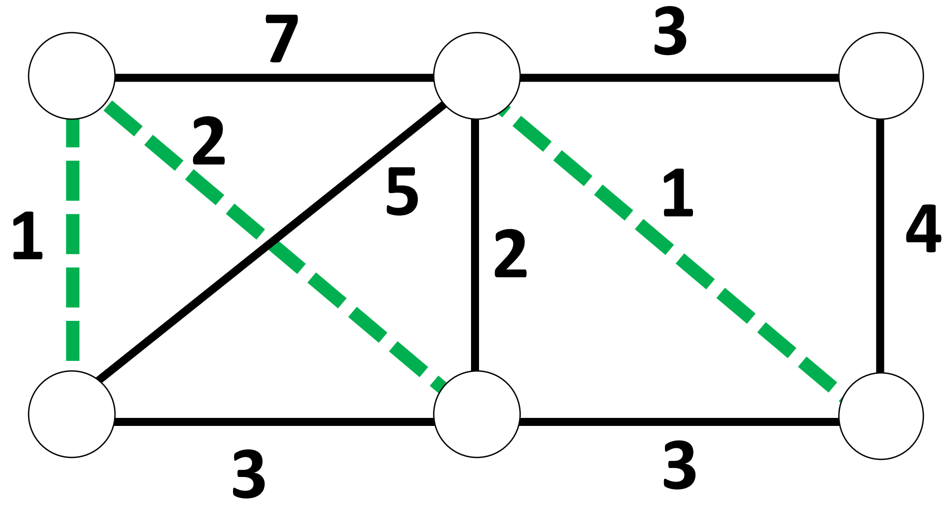
Kruskal’s MST Algorithm

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



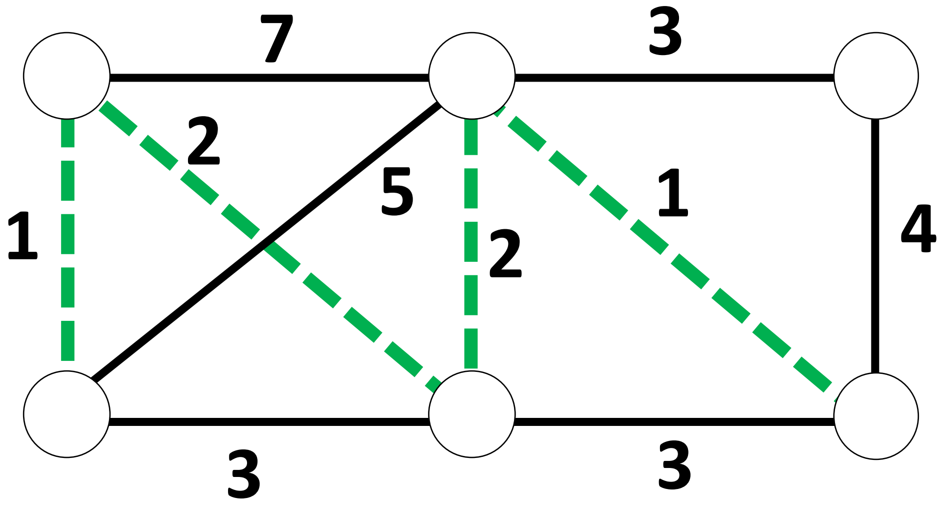
Kruskal's MST Algorithm

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



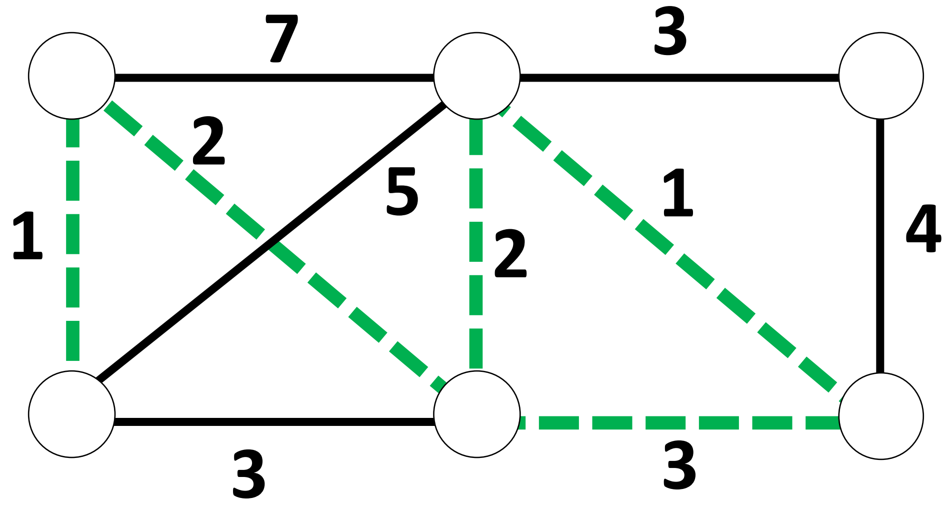
Kruskal's MST Algorithm

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



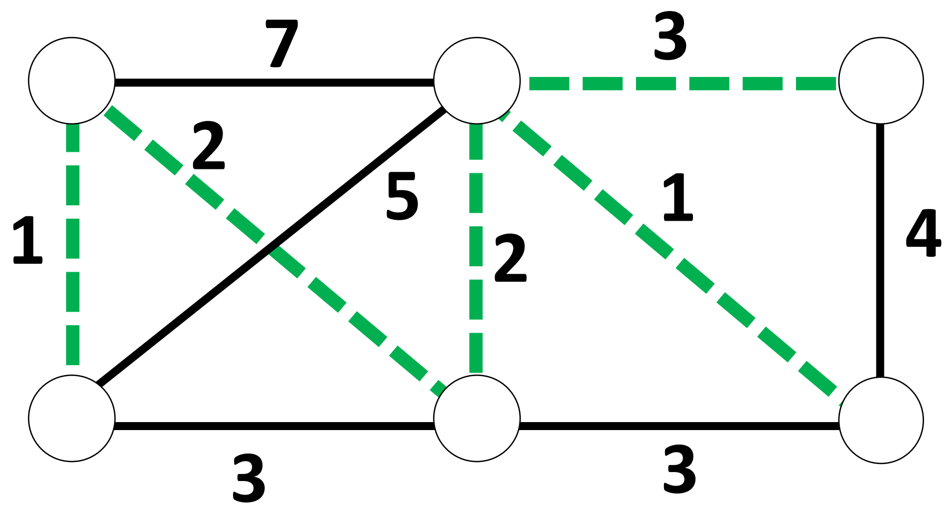
Kruskal's MST Algorithm

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



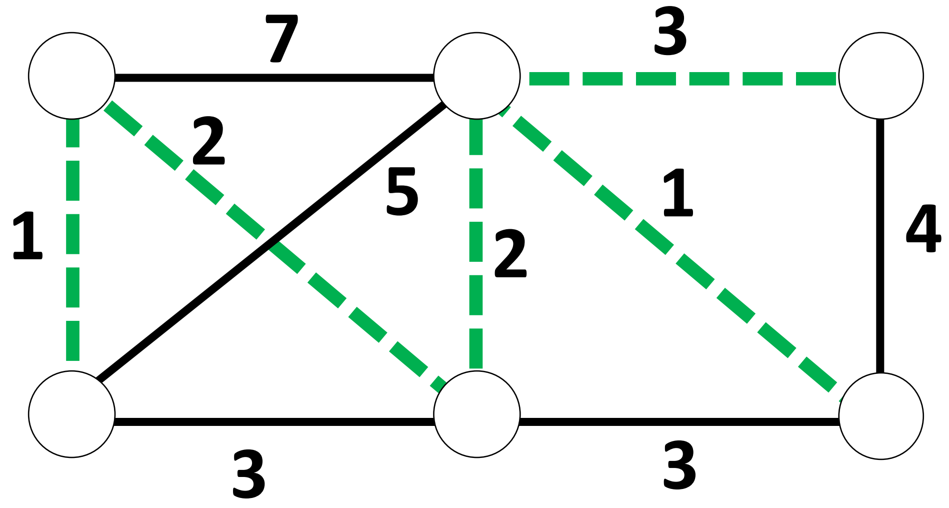
Kruskal’s MST Algorithm

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



Kruskal's MST Algorithm

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



What are some questions we may have about the algorithm?

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

Kruskal’s MST Algorithm

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

Proof of validity: ?

Kruskal's MST Algorithm

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

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

Kruskal's MST Algorithm

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

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

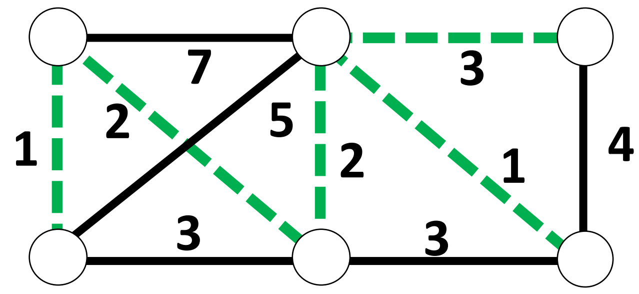
What do we need to show?

Kruskal’s MST Algorithm

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

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

T is a tree because it is connected (otherwise we could have added more edges without creating cycles) and there are no cycles.



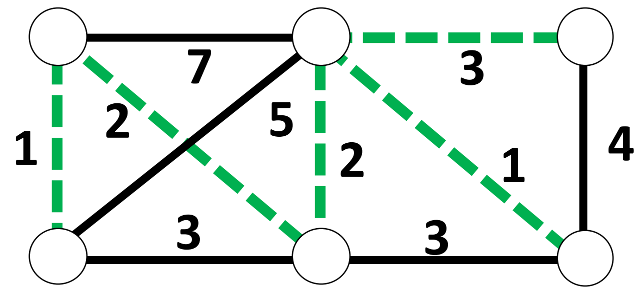
Kruskal's MST Algorithm

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

<u>Proof of validity:</u> Let $G = (V, E)$ be the connected graph, and $T \subseteq E$ be the set of edges resulting from Kruskal's algorithm.

T is a tree because it is connected (otherwise we could have added more edges without creating cycles) and there are no cycles.

T spans G because if it did not, we could have added more edges to connected unreached nodes without creating cycles.



Kruskal's MST Algorithm

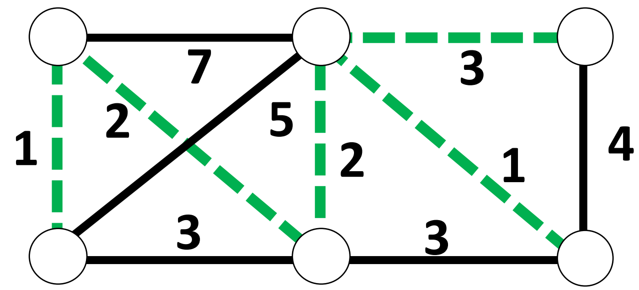
Algorithm: Add the edge with smallest weight, that does not create a cycle.

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

T is a tree because it is connected (otherwise we could have added more edges without creating cycles) and there are no cycles.

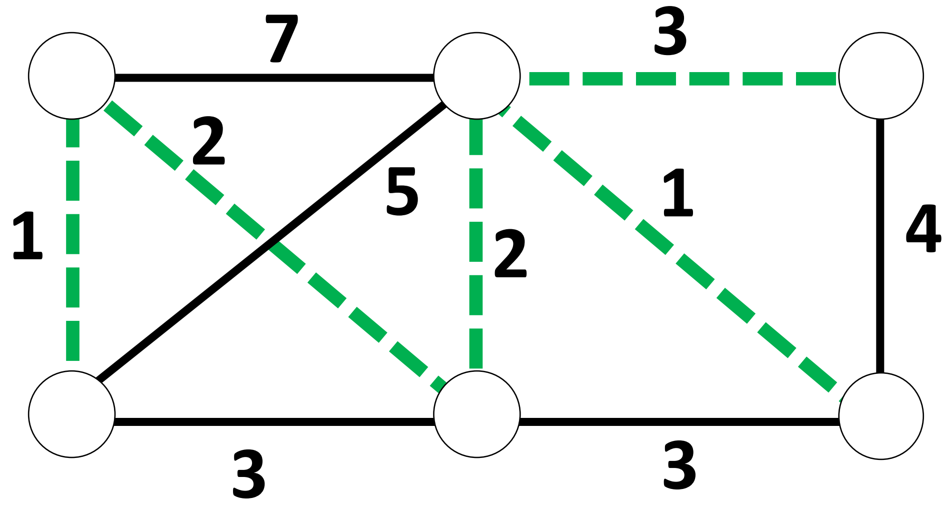
T spans G because if it did not, we could have added more edges to connected unreached nodes without creating cycles.

$\therefore$ T is a spanning tree of G



Kruskal's MST Algorithm

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



What are some questions we may have about the algorithm?

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

Kruskal’s MST Algorithm

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

Running Time:

Kruskal's MST Algorithm

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

<u>Running Time:</u>

```
findMST(G=(V,E)) {
  T = Ø
  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 ∪ {e} is acyclic) {
      T = T ∪ {e}
    }
  }
  return T
}
```

← $O(|E| \log(|E|))$

Kruskal’s MST Algorithm

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

Running Time:

```
findMST(G=(V,E)) {
   T = Ø
   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 ∪ {e} is acyclic) {   ← O(|V| + |E|) using BFS
      T = T ∪ {e}
    }
  }
  return T
}
```

Kruskal's MST Algorithm

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

<u>Running Time:</u>

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

sort(E) ← $O(|E|\log(|E|))$

for (e in E) ← $O(|E|)$

if (T ∪ {e} is acyclic) ← $O(|V| + |E|)$ **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:

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

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

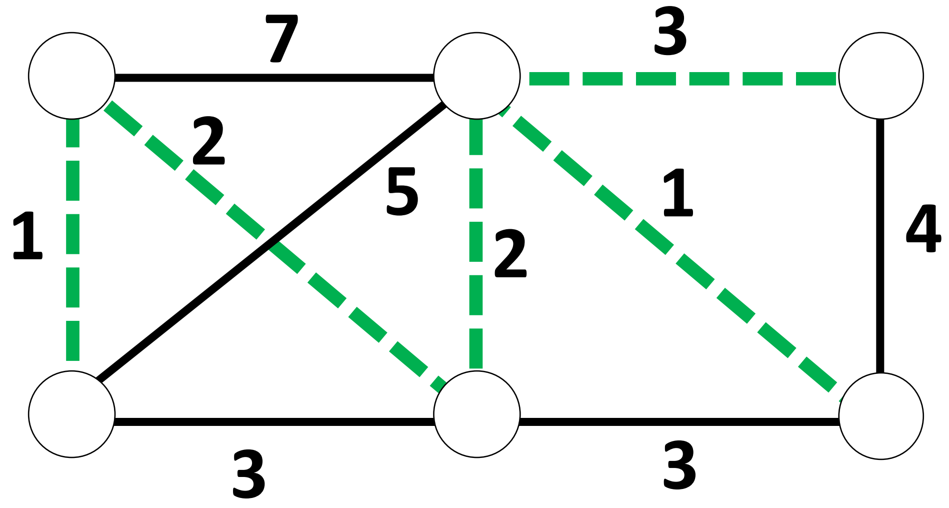
- sort(E): $O(|E| \log(|E|))$
- for (e in E): $O(|E|)$
- if (T ∪ {e} is acyclic): $O(|V| + |E|)$ using BFS

Running time

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

Kruskal's MST Algorithm

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



What are some questions we may have about the algorithm?

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

Kruskal's MST Algorithm

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

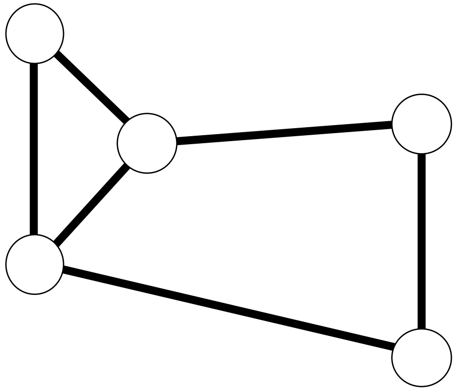
Proof of optimality: T is an MST, because???

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.

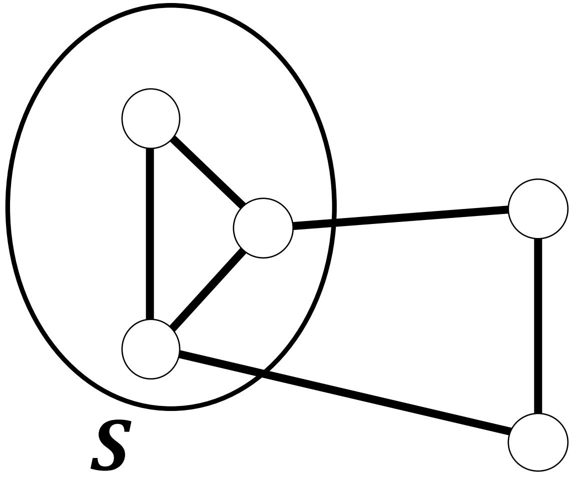
Proof:



MST Cut Property

<u>Lemma:</u> 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.

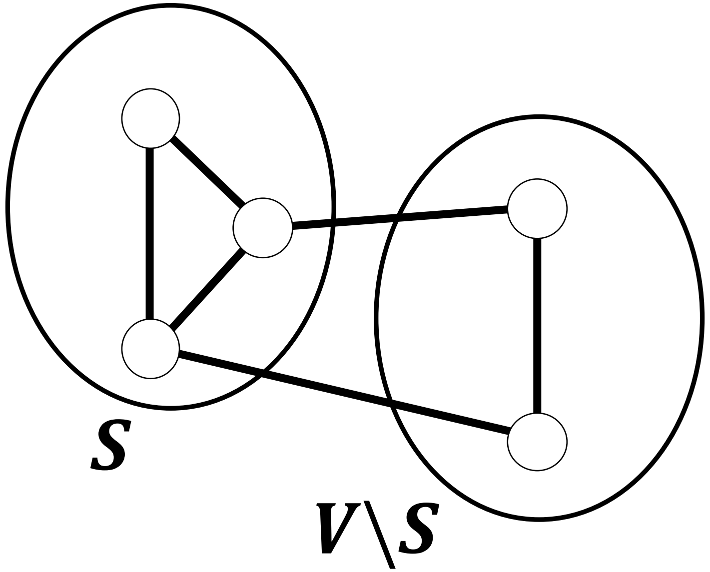
<u>Proof:</u>



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.

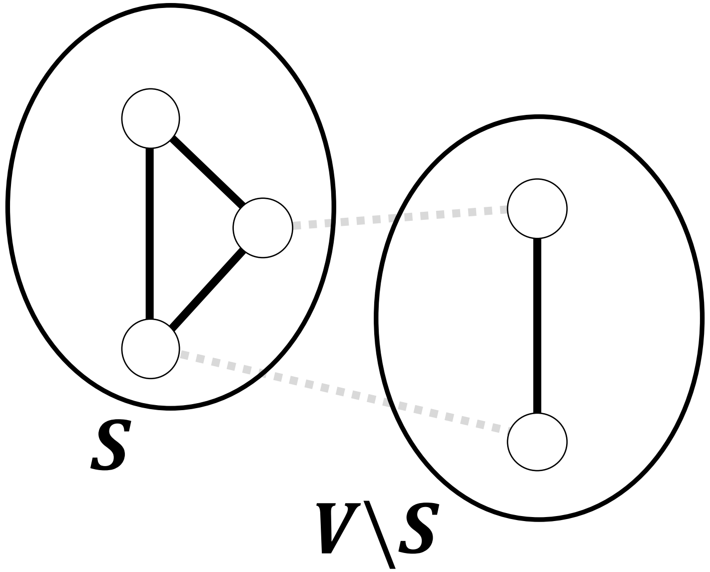
Proof:



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).

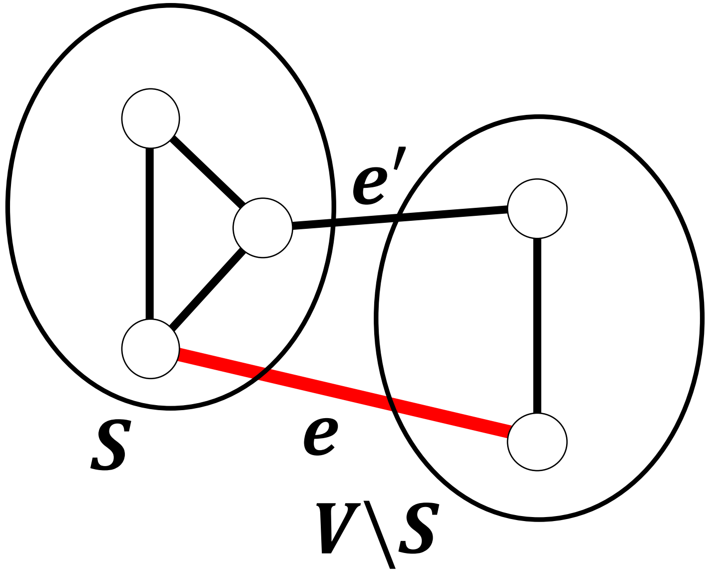


MST Cut Property

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 $V \backslash S$.



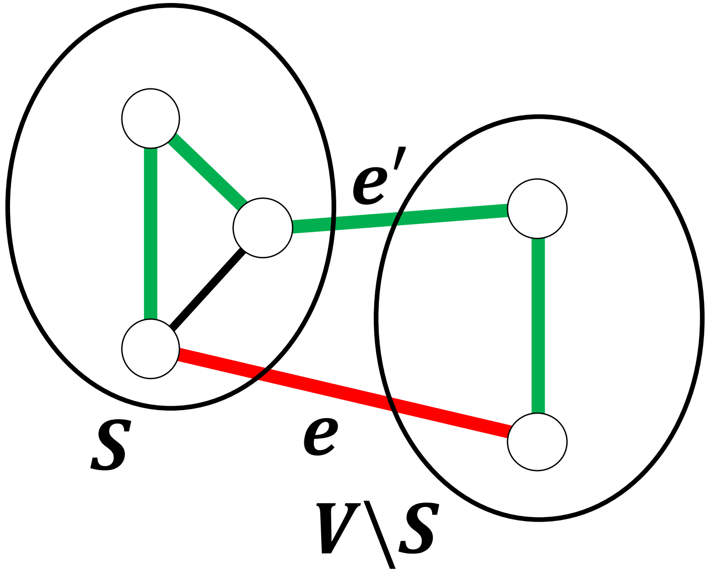
MST Cut Property

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

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

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

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



MST Cut Property

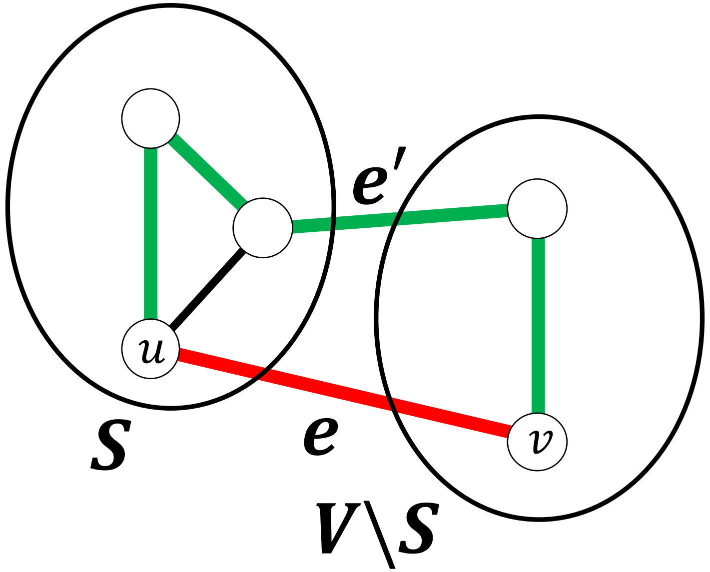
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

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

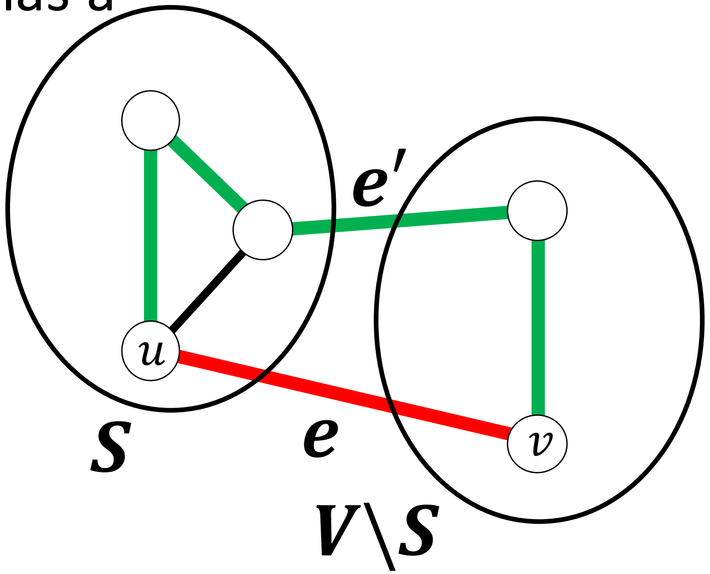
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

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

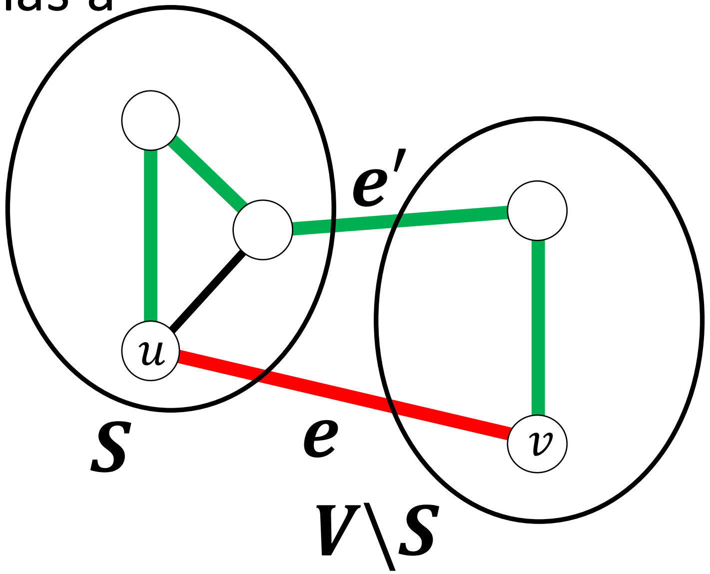
<u>Lemma:</u> Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

<u>Proof:</u> 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.

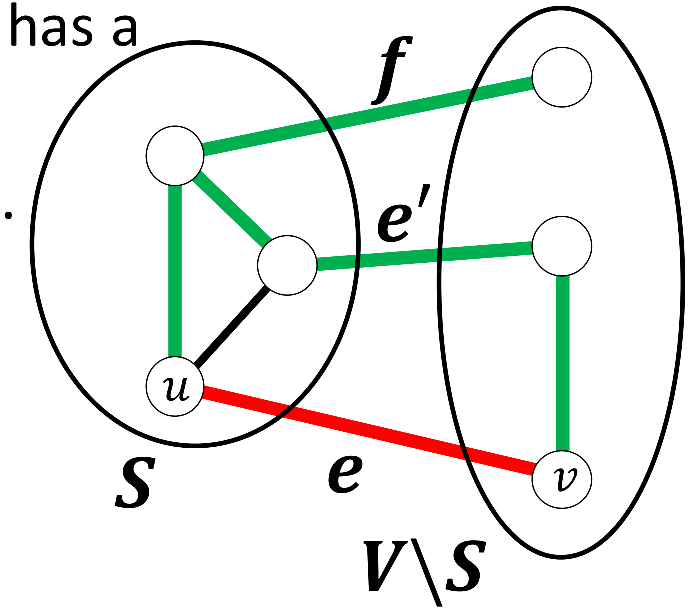
Proof: Any MST of G must include some edge between S and $V\backslash S$ (otherwise it would not be a spanning tree).

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

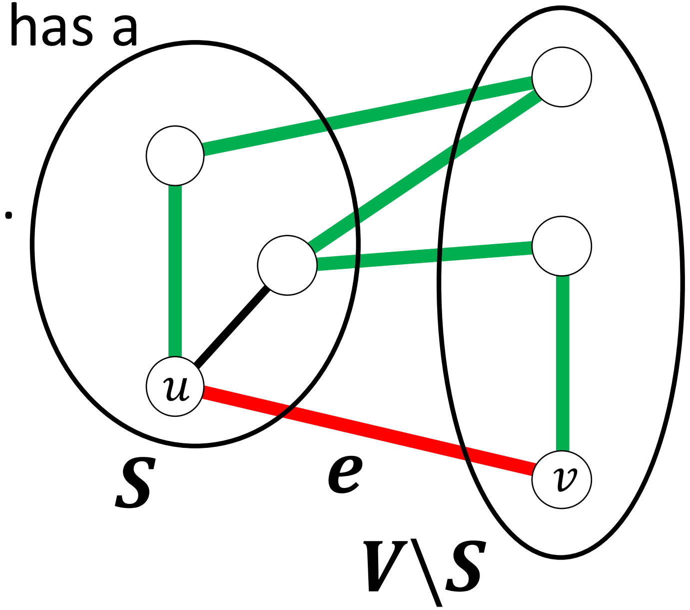
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

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

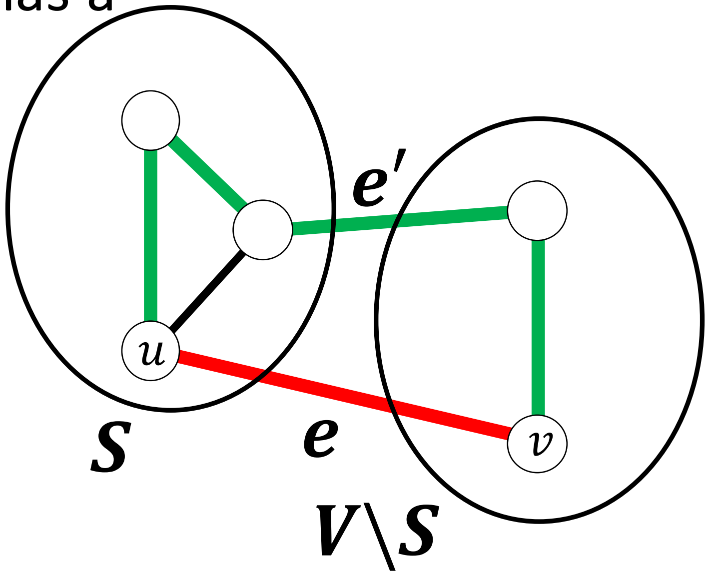
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V\backslash S$ is part of the MST.

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 $\boldsymbol{T}$)



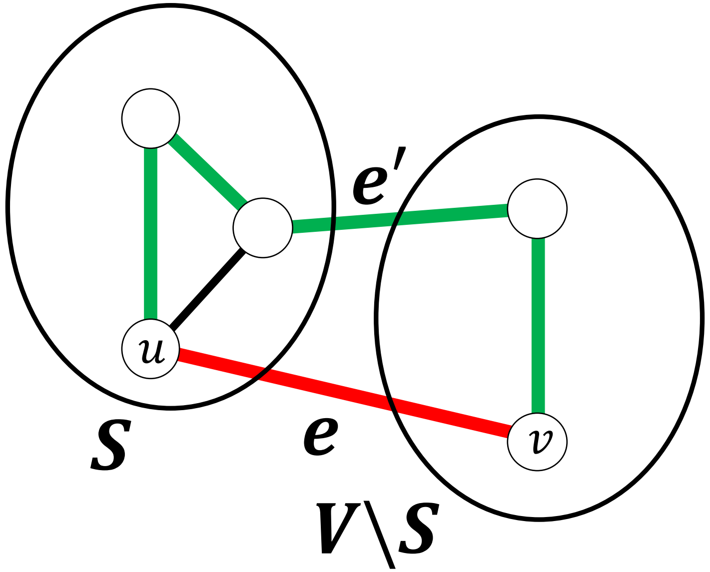
MST Cut Property

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

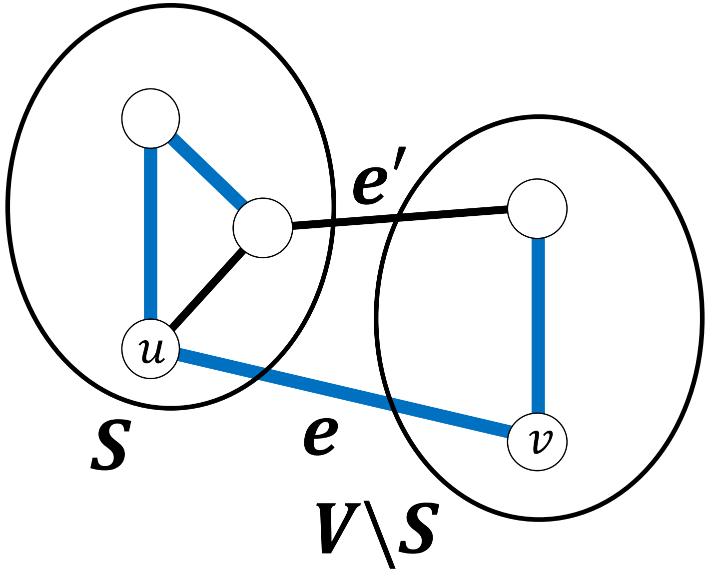
Lemma: Suppose that S is a subset of nodes from $G = (V, E)$. Then, the cheapest edge e between S and $V \backslash S$ is part of the MST.

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 $V \backslash S$.

Remove 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.

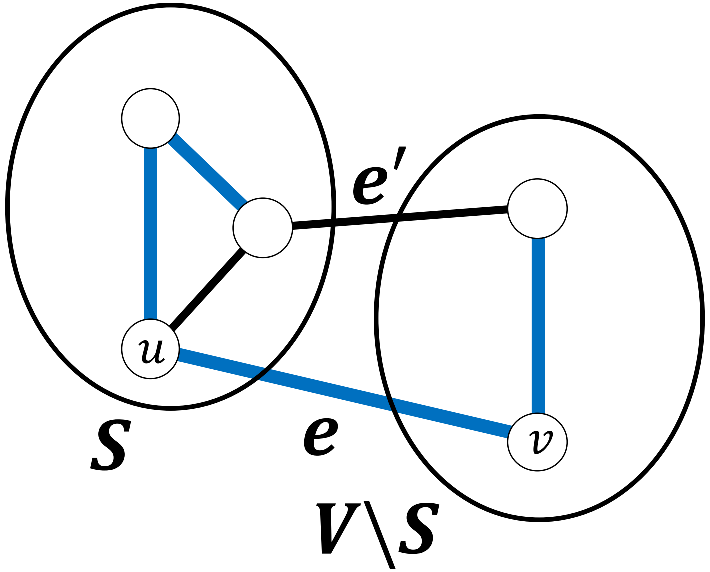
Proof: Any MST of G must include some edge between S and $V \backslash S$ (otherwise it would not be a spanning tree).

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 $V \backslash S$.

Remove 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).

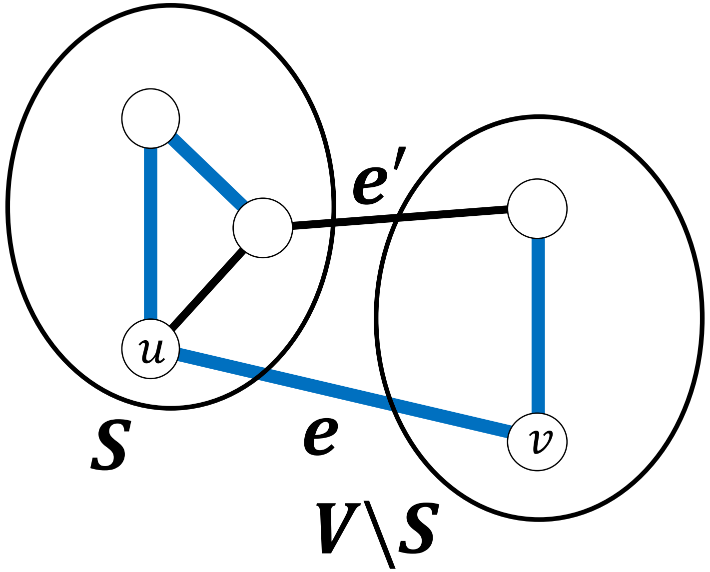
Let e be the cheapest edge between S and $V \backslash S$.

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 $V \backslash S$.

Remove 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

<u>Lemma:</u> 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.

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

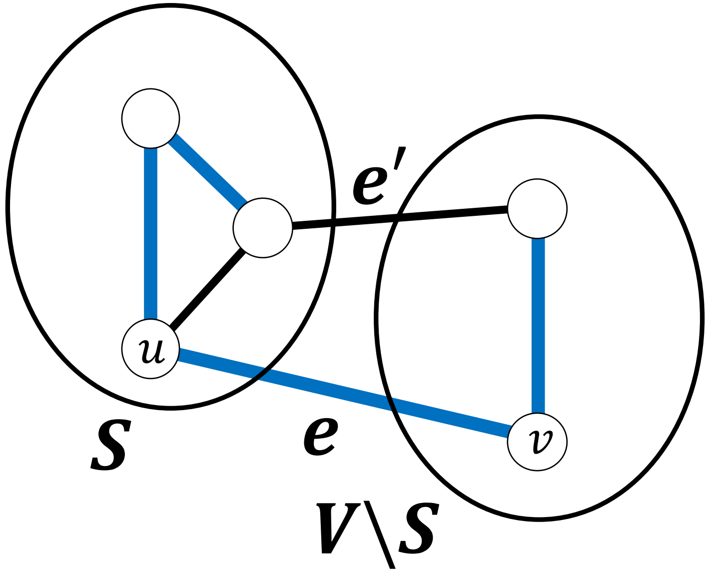
Let e be the cheapest edge between S and $V \backslash S$.

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 $V \backslash S$.

Remove 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).

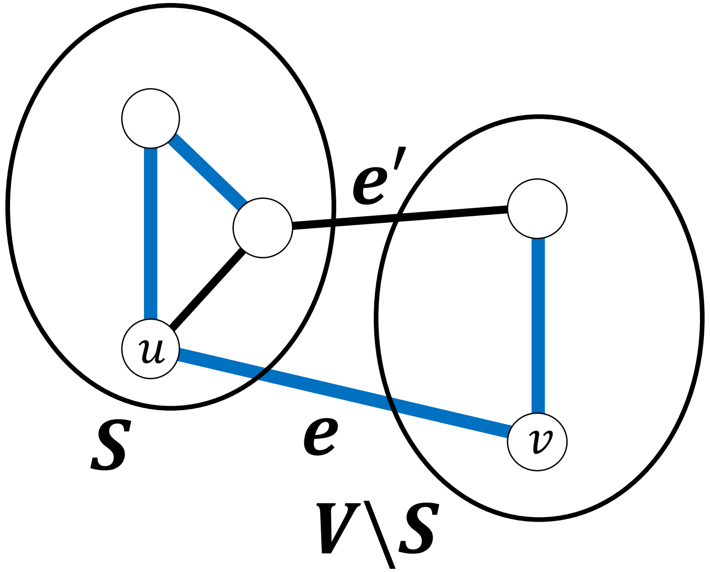
Let e be the cheapest edge between S and $V\backslash S$.

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 $V\backslash S$.

Remove 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 $V\backslash S$.

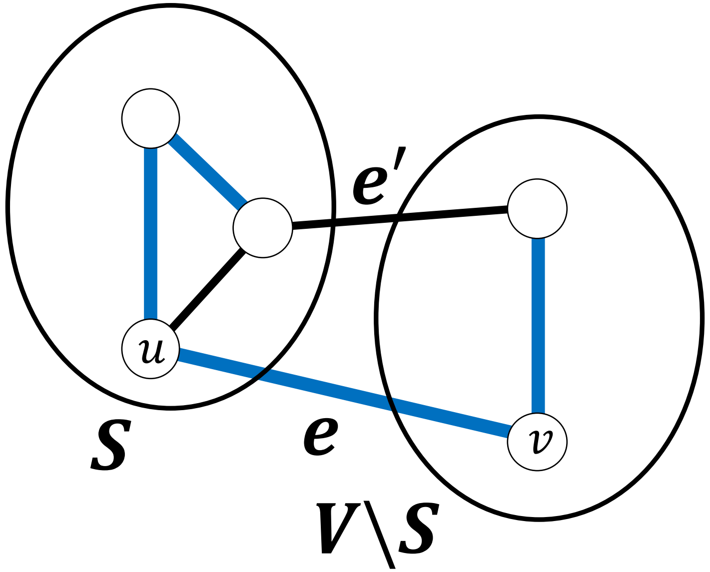
Suppose T is the MST that does not include e. $T \cup \{e\}$ must have a cycle and that cycle must have another edge e' between S and $V\backslash S$.

Remove 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 $V \backslash S$.

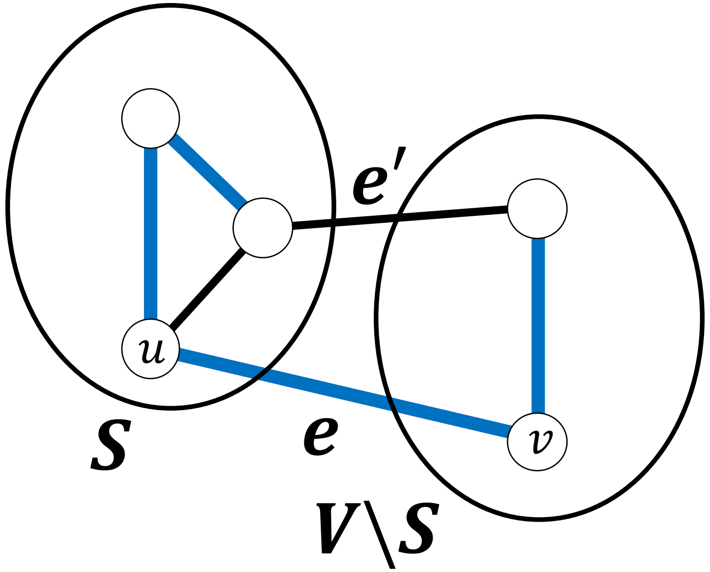
Remove e' to form $\boldsymbol{T'} = T \cup \{e\} \backslash \{e'\}$.

$\boldsymbol{T'}$ 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?

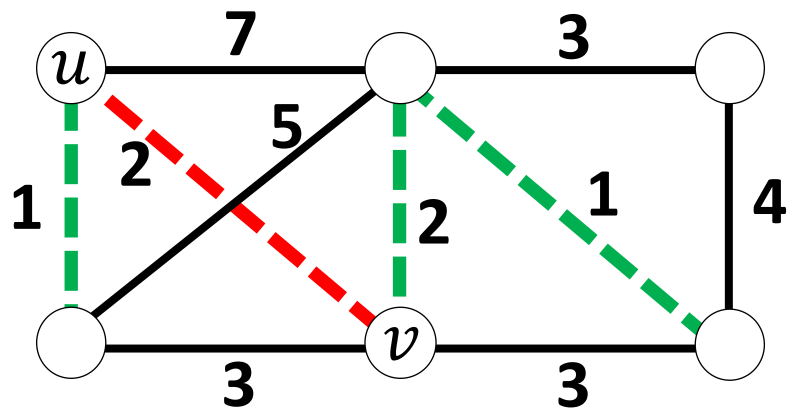
Lemma: The cheapest edge between $S \subseteq V$ and $V \setminus S$ is part of the MST.

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



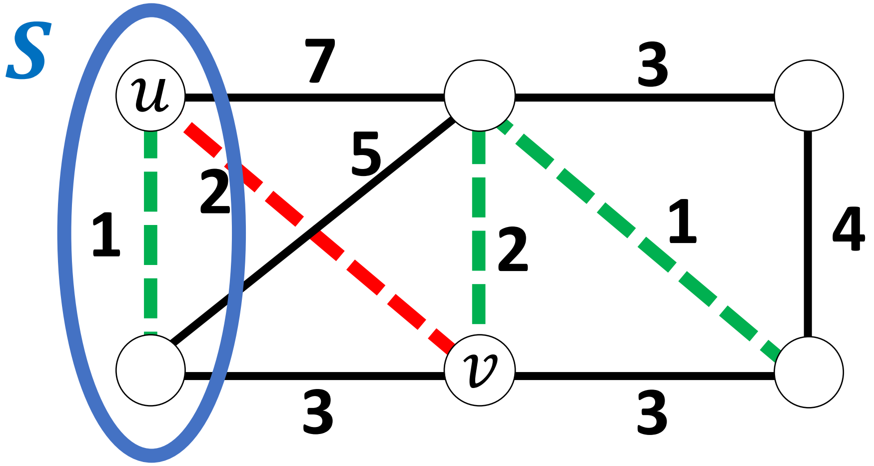
Lemma: The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

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 interation 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)



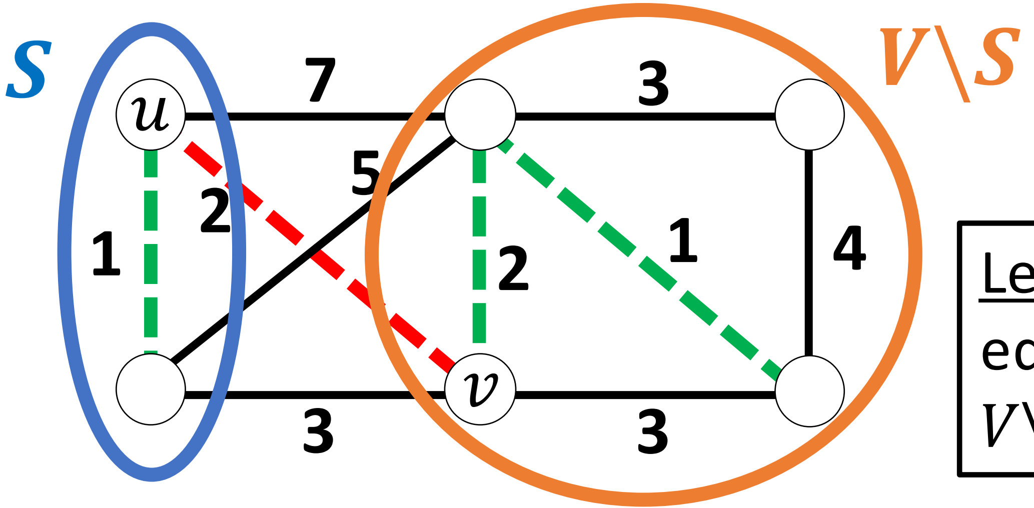
Lemma: The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

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 interation 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 $v \in \boldsymbol{V \setminus S}$ (otherwise adding $\boldsymbol{e}$ would have created a cycle).



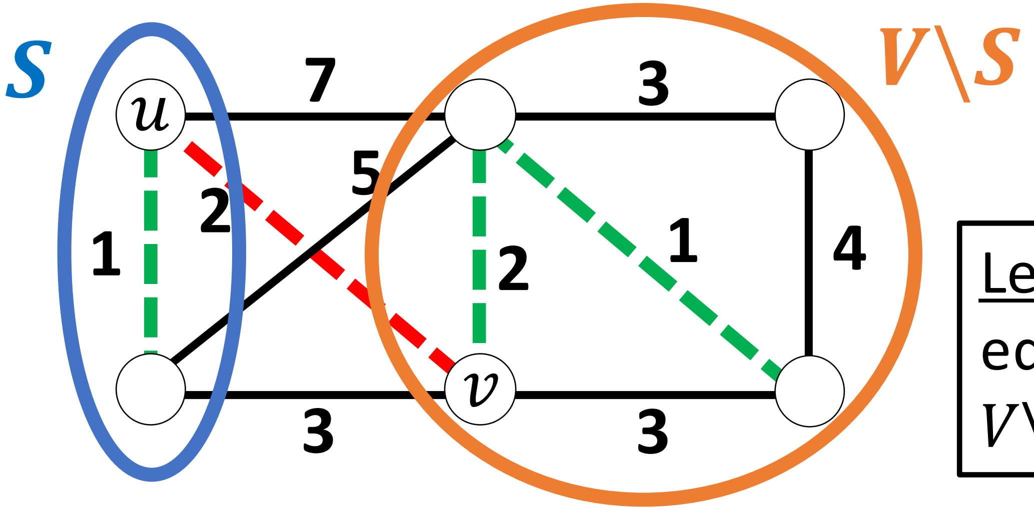
Lemma: The cheapest edge between $S \subseteq V$ and $V \setminus S$ is part of the MST.

Kruskal's MST Algorithm

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

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

Consider the interation 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).



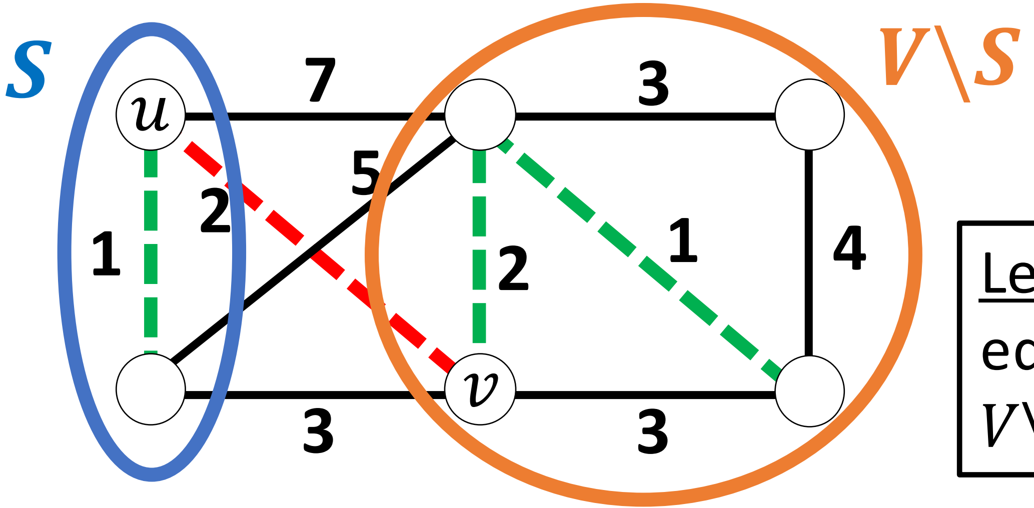
<u>Lemma:</u> The cheapest edge between $S \subseteq V$ and $V \backslash S$ is part of the MST.

Kruskal's MST Algorithm

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

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

Consider the interation 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.



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

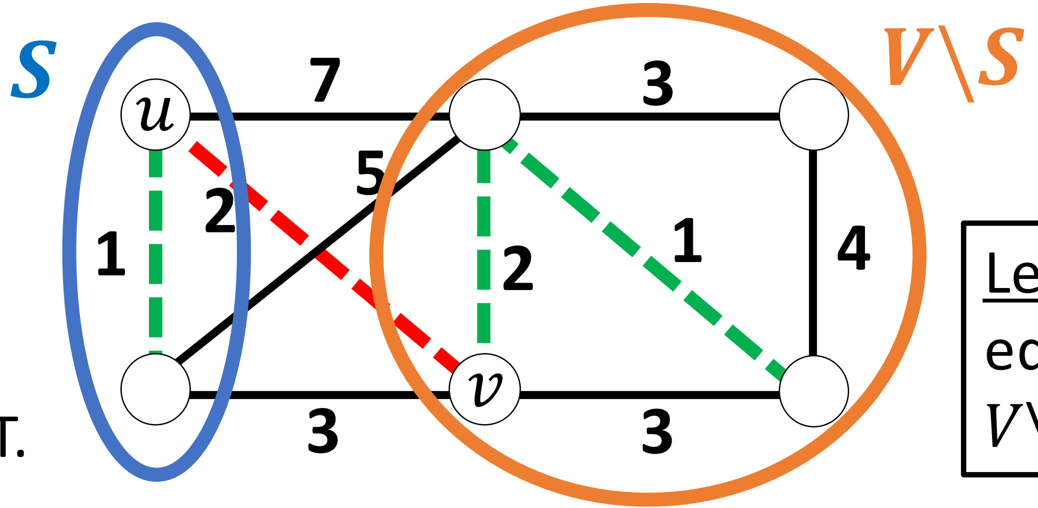
Kruskal's MST Algorithm

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 interation 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.

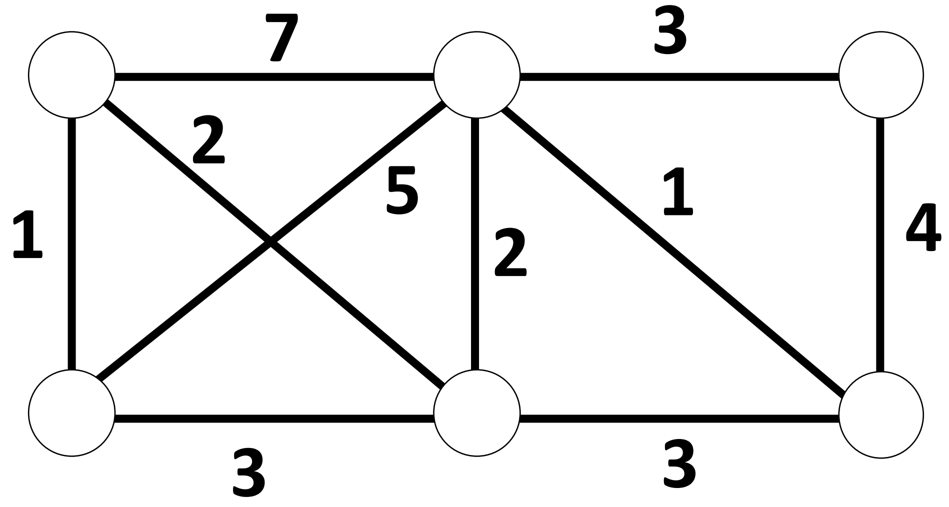
Thus, every edge found by Kruskal's algorithm is part of the MST, and since the edges found form a spanning tree, it is the MST.



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

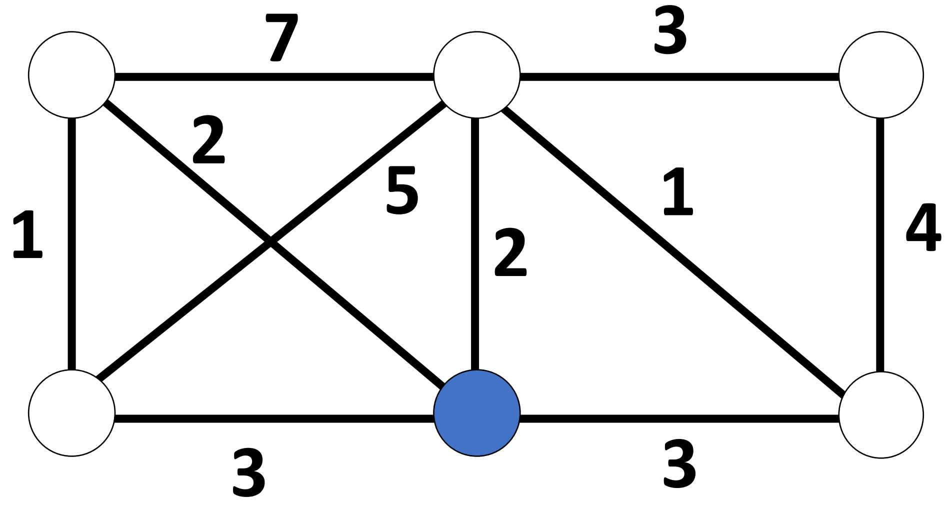
Prim's MST Algorithm

Algorithm: Mark a random node as ***connected***. Find the **edge** with smallest weight between a ***connected*** node and one that is not. Mark both endpoints as ***connected***.



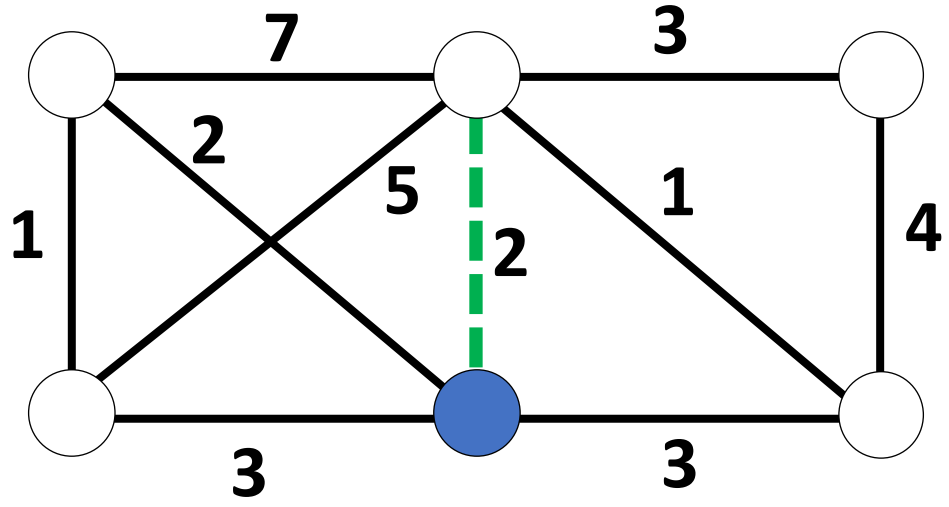
Prim's MST Algorithm

<u>Algorithm:</u> Mark a random node as ***connected***. Find the **edge** with smallest weight between a ***connected*** node and one that is not. Mark both endpoints as ***connected***.



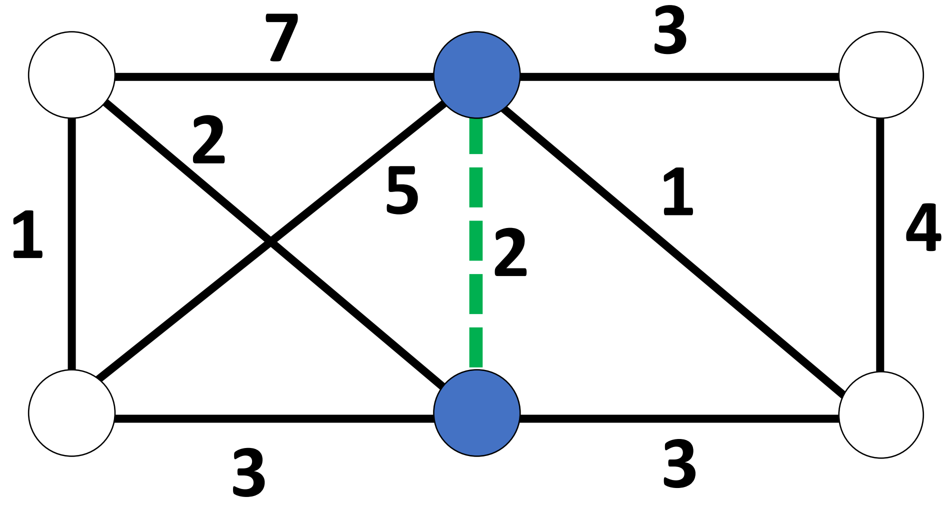
Prim’s MST Algorithm

<u>Algorithm:</u> Mark a random node as ***connected***. Find the **edge** with smallest weight between a ***connected*** node and one that is not. Mark both endpoints as ***connected***.



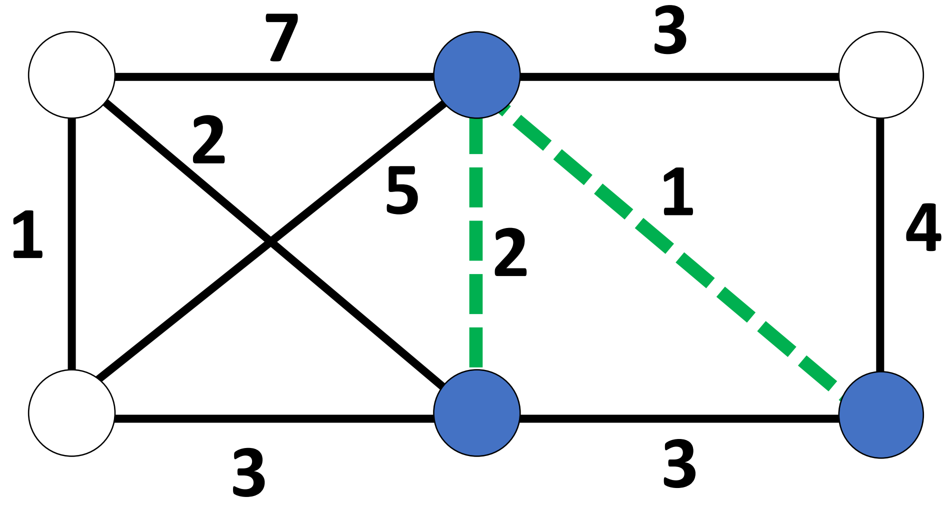
Prim's MST Algorithm

Algorithm: Mark a random node as ***connected***. Find the **edge** with smallest weight between a ***connected*** node and one that is not. Mark both endpoints as ***connected***.



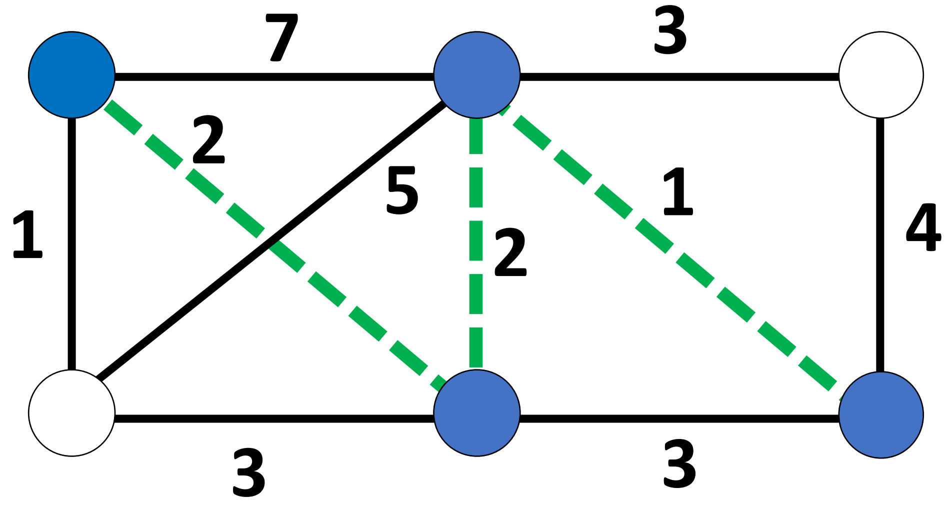
Prim's MST Algorithm

<u>Algorithm:</u> Mark a random node as ***connected***. Find the **edge** with smallest weight between a ***connected*** node and one that is not. Mark both endpoints as ***connected***.



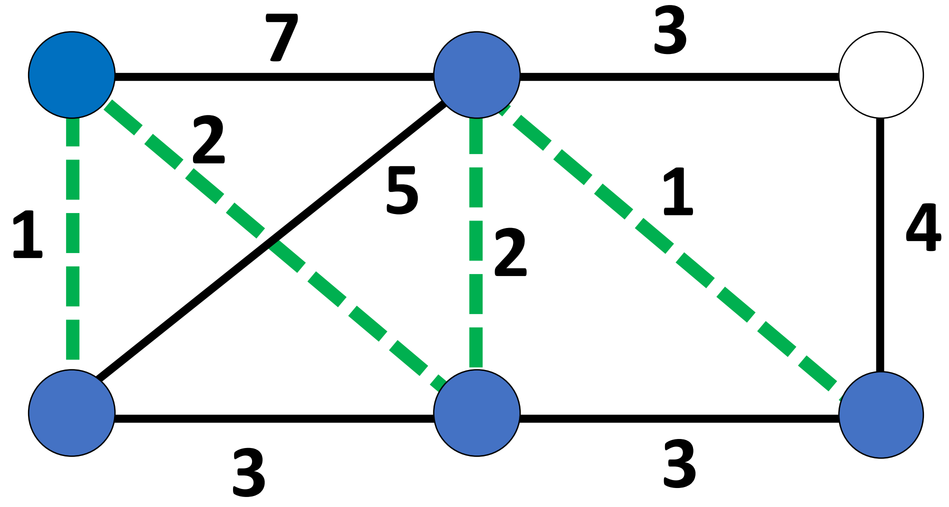
Prim's MST Algorithm

<u>Algorithm:</u> Mark a random node as ***connected***. Find the **edge** with smallest weight between a ***connected*** node and one that is not. Mark both endpoints as ***connected***.



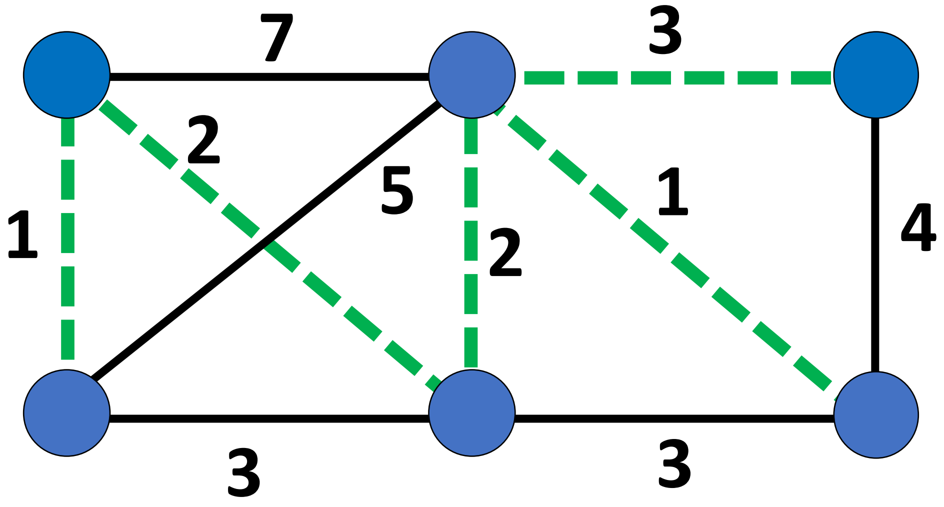
Prim’s MST Algorithm

Algorithm: Mark a random node as ***connected***. Find the **edge** with smallest weight between a ***connected*** node and one that is not. Mark both endpoints as ***connected***.



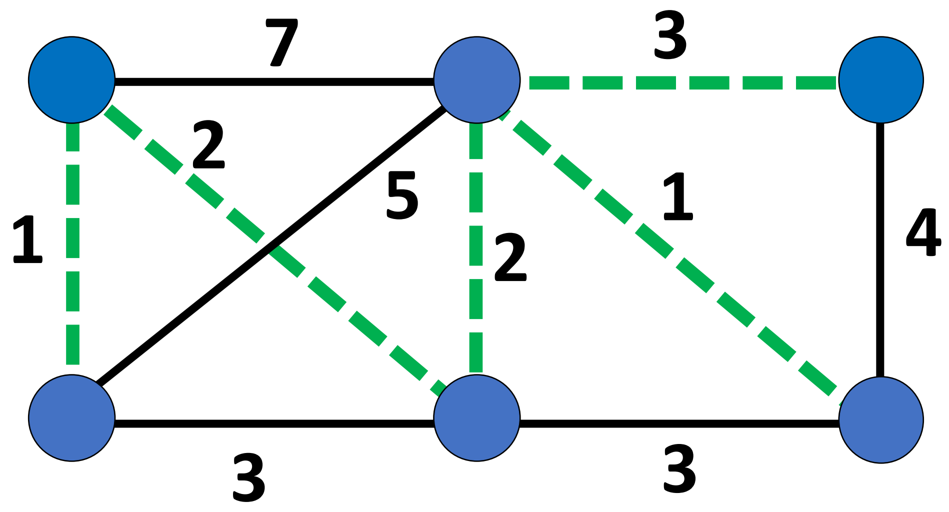
Prim’s MST Algorithm

<u>Algorithm:</u> Mark a random node as ***connected***. Find the **edge** with smallest weight between a ***connected*** node and one that is not. Mark both endpoints as ***connected***.



Prim's MST Algorithm

Algorithm: Mark a random node as ***connected***. Find the **edge** with smallest weight between a ***connected*** node and one that is not. Mark both endpoints as ***connected***.



Homework Questions:

1. Is the solution valid? (Does it actually find a spanning tree?)
2. Is the solution optimal?

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 every MST.