Minimum Cost Spanning Trees: Kruskal's Algorithm

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Mathematics for Data Science 1
Week 12

Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V

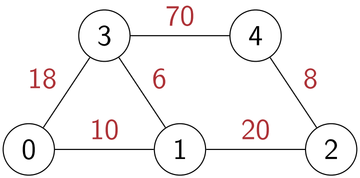
Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

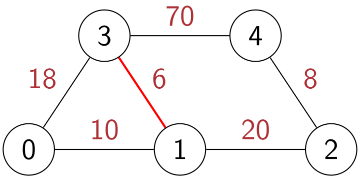
Example



Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

Example

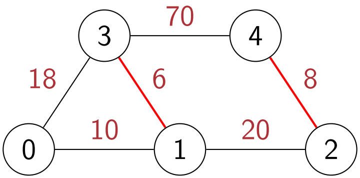


- Start with smallest edge, $(1, 3)$

Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

Example

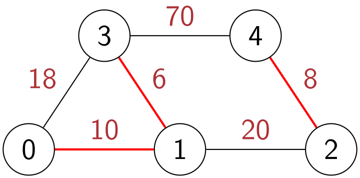


- Start with smallest edge, $(1, 3)$
- Add next smallest edge, $(2, 4)$

Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

Example

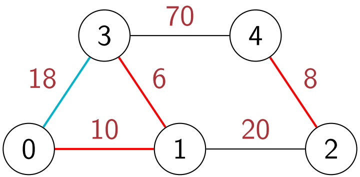


- Start with smallest edge, $(1, 3)$
- Add next smallest edge, $(2, 4)$
- Add next smallest edge, $(0, 1)$

Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

Example

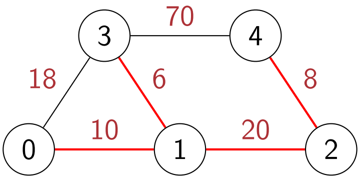


- Start with smallest edge, $(1, 3)$
- Add next smallest edge, $(2, 4)$
- Add next smallest edge, $(0, 1)$
- Can't add $(0, 3)$, forms a cycle

Minimum cost spanning tree (MCST)

- Weighted undirected graph, $G = (V, E), W : E \to \mathbb{R}$
  - G assumed to be connected
- Find a minimum cost spanning tree
  - Tree connecting all vertices in V
- Strategy 2
  - Start with n components, each a single vertex
  - Process edges in ascending order of cost
  - Include edge if it does not create a cycle

Example



- Start with smallest edge, $(1, 3)$
- Add next smallest edge, $(2, 4)$
- Add next smallest edge, $(0, 1)$
- Can't add $(0, 3)$, forms a cycle
- Add next smallest edge, $(1, 2)$

Kruskal's algorithm

- $G = (V, E)$, $W : E \rightarrow \mathbb{R}$

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$

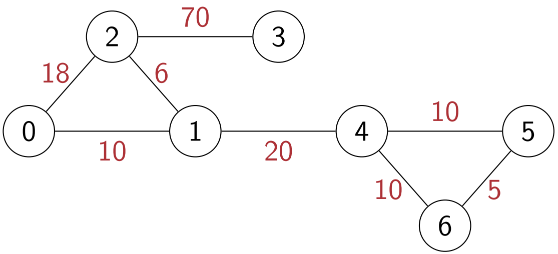
Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Kruskal's algorithm

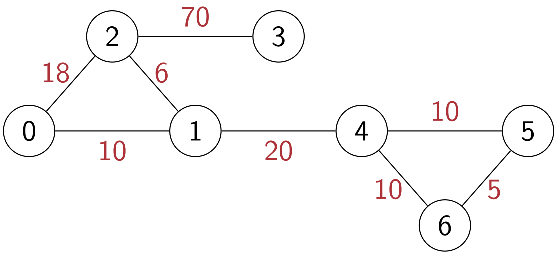
- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



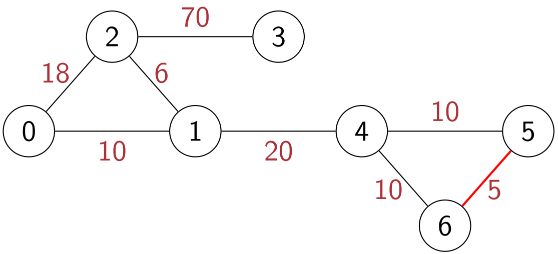
Sort E as
$\{(5,6), (1,2), (0,1), (4,5), (4,6), (0,2), (1,4), (2,3)\}$

Set $TE = \emptyset$

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



Sort E as
$\{(5,6),(1,2),(0,1),(4,5),(4,6),(0,2),(1,4),(2,3)\}$

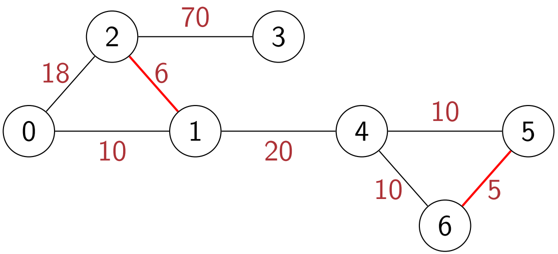
Add $(5,6)$

Set $TE = \{(5,6)\}$

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



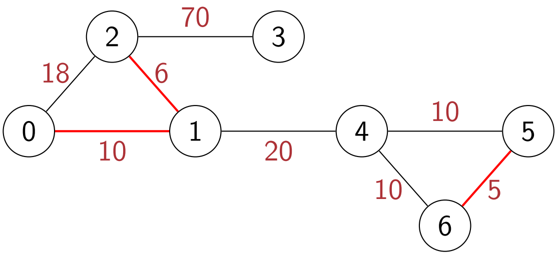
Sort E as
$\{(5,6),(1,2),(0,1),(4,5),(4,6),(0,2),(1,4),(2,3)\}$

Add $(1,2)$

Set $TE = \{(5,6),(1,2)\}$

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



Sort E as
$\{(5,6),(1,2),(0,1),(4,5),(4,6),(0,2),(1,4),(2,3)\}$

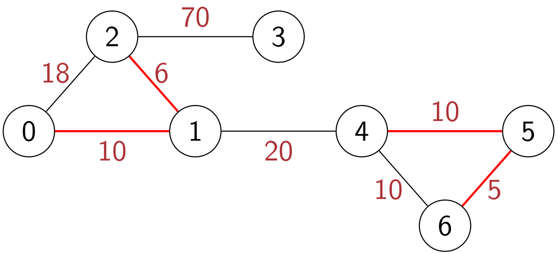
Add $(0,1)$

Set $TE = \{(5,6),(1,2),(0,1)\}$

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



Sort E as

$\{(5,6),(1,2),(0,1),(4,5),(4,6),(0,2),(1,4),(2,3)\}$

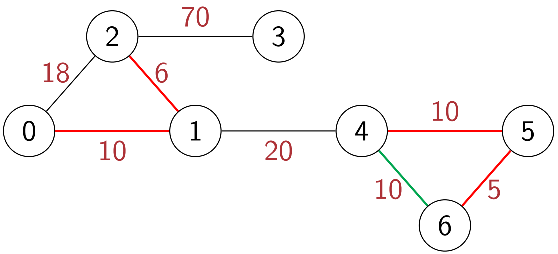
Add $(4,5)$

Set $TE = \{(5,6),(1,2),(0,1),(4,5)\}$

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



Sort E as

$\{(5,6),(1,2),(0,1),(4,5),(4,6),(0,2),(1,4),(2,3)\}$

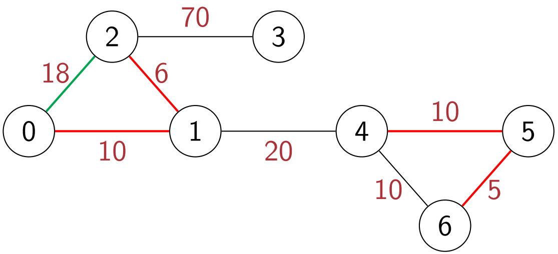
Skip $(4,6)$

Set $TE = \{(5,6),(1,2),(0,1),(4,5)\}$

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



Sort E as

$\{(5,6),(1,2),(0,1),(4,5),(4,6),(0,2),(1,4),(2,3)\}$

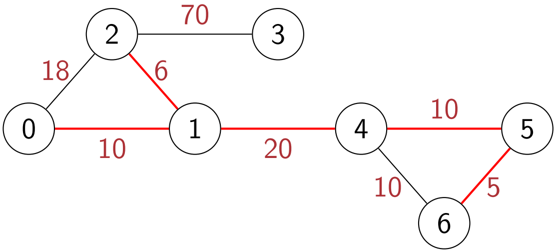
Skip $(0,2)$

Set $TE = \{(5,6),(1,2),(0,1),(4,5)\}$

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



Sort E as

$\{(5,6),(1,2),(0,1),(4,5),(4,6),(0,2),(1,4),(2,3)\}$

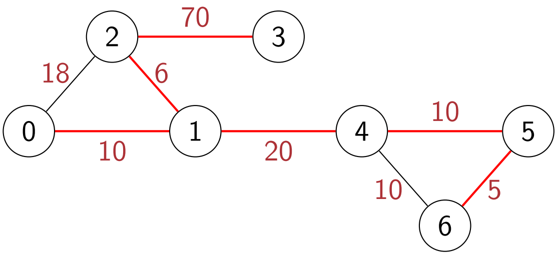
Add $(1,4)$

Set $TE = \{(5,6),(1,2),(0,1),(4,5),(1,4)\}$

Kruskal's algorithm

- $G = (V, E)$, $W : E \to \mathbb{R}$
- Let $E = \{e_0, e_1, \ldots, e_{m-1}\}$ be edges sorted in ascending order by weight
- Let $TE \subseteq E$ be the set of tree edges already added to MCST
- Initially, $TE = \emptyset$
- Scan E from e_0 to e_{m-1}
  - If adding e_i to TE creates a loop, skip it
  - Otherwise, add e_i to TE

Example



Sort E as
$\{(5,6),(1,2),(0,1),(4,5),(4,6),(0,2),(1,4),(2,3)\}$

Add $(2,3)$

Set $TE = \{(5,6),(1,2),(0,1),(4,5),(1,4),(2,3)\}$

Correctness of Krusksal's algorithm

Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

Correctness of Krusksal's algorithm

Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

- Edges in TE partition vertices into connected components

Correctness of Krusksal's algorithm

Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

- Edges in TE partition vertices into connected components
- Initially each vertex is a separate component

Correctness of Krusksal's algorithm

Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

- Edges in TE partition vertices into connected components
- Initially each vertex is a separate component
- Adding $e = (u, w)$ merges components of u and w

Correctness of Krusksal's algorithm

Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

- Edges in TE partition vertices into connected components
- Initially each vertex is a separate component
- Adding $e = (u, w)$ merges components of u and w
  - If u and w are in the same component, e forms a cycle and is discarded

Correctness of Krusksal's algorithm

Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

- Edges in TE partition vertices into connected components
- Initially each vertex is a separate component
- Adding $e = (u, w)$ merges components of u and w
  - If u and w are in the same component, e forms a cycle and is discarded
  - Otherwise, let U be component of u, W be $V \setminus U$

Correctness of Krusksal's algorithm

Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

- Edges in TE partition vertices into connected components
- Initially each vertex is a separate component
- Adding $e = (u, w)$ merges components of u and w
  - If u and w are in the same component, e forms a cycle and is discarded
  - Otherwise, let U be component of u, W be $V \setminus U$
  - U, W form a partition of V with $u \in U$ and $w \in W$

Correctness of Krusksal's algorithm

Minimum Separator Lemma

- Let V be partitioned into two non-empty sets U and $W = V \setminus U$
- Let $e = (u, w)$ be the minimum cost edge with $u \in U$, $w \in W$
- Every MCST must include e

- Edges in TE partition vertices into connected components
- Initially each vertex is a separate component
- Adding $e = (u, w)$ merges components of u and w
  - If u and w are in the same component, e forms a cycle and is discarded
  - Otherwise, let U be component of u, W be $V \setminus U$
  - U, W form a partition of V with $u \in U$ and $w \in W$
  - Since we are scanning edges in ascending order of cost, e is minimum cost edge connecting U and W, so it must be part of any MCST

Summary

- Kruskal's algorithm builds an MCST bottom up
  - Start with n components, each an isolated vertex
  - Scan edges in ascending order of cost
  - Whenever an edge merges disjoint components, add it to the MCST
- Correctness follows from Minimum Separator Lemma

Summary

- Kruskal's algorithm builds an MCST bottom up
  - Start with n components, each an isolated vertex
  - Scan edges in ascending order of cost
  - Whenever an edge merges disjoint components, add it to the MCST
- Correctness follows from Minimum Separator Lemma
- If edge weights repeat, MCST is not unique

Summary

- Kruskal's algorithm builds an MCST bottom up
  - Start with n components, each an isolated vertex
  - Scan edges in ascending order of cost
  - Whenever an edge merges disjoint components, add it to the MCST
- Correctness follows from Minimum Separator Lemma
- If edge weights repeat, MCST is not unique
- "Choose minimum cost edge" will allow choices
  - Consider a triangle on 3 vertices with all edges equal

Summary

- Kruskal's algorithm builds an MCST bottom up
  - Start with n components, each an isolated vertex
  - Scan edges in ascending order of cost
  - Whenever an edge merges disjoint components, add it to the MCST
- Correctness follows from Minimum Separator Lemma
- If edge weights repeat, MCST is not unique
- "Choose minimum cost edge" will allow choices
  - Consider a triangle on 3 vertices with all edges equal
- Different choices lead to different spanning trees

Summary

- Kruskal's algorithm builds an MCST bottom up
  - Start with n components, each an isolated vertex
  - Scan edges in ascending order of cost
  - Whenever an edge merges disjoint components, add it to the MCST
- Correctness follows from Minimum Separator Lemma
- If edge weights repeat, MCST is not unique
- "Choose minimum cost edge" will allow choices
  - Consider a triangle on 3 vertices with all edges equal
- Different choices lead to different spanning trees
- In general, there may be a very large number of minimum cost spanning trees