# Lesson 5: Minimum Spanning Tree

## Notes

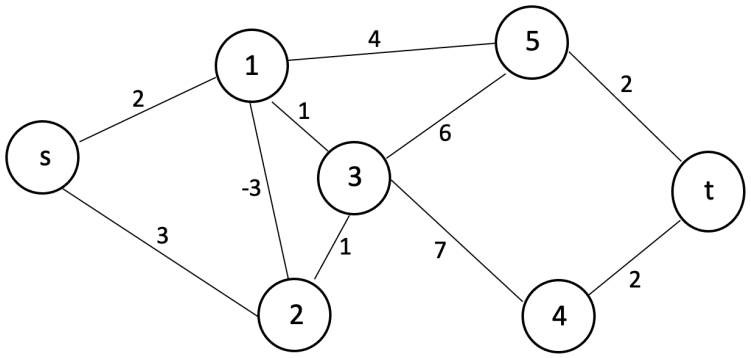
Book acknowledgment:

## Goals

- Minimum Spanning Tree

## 1 Try it on your own

For the network below, find the set of edges that (a) have the minimum cumulative cost and (b) ensure that all nodes are connected. In other words, determine the minimum cumulative weight spanning tree.




What is your solution? How did you determine it? Can you transform your strategy into an algorithm?

## 2 Problem Definition

Given a graph $G = (V, E)$, in which edge $(i, j) \in E$ has a weight $w_{ij}$. Determine the subset of edges of a connected subgraph that connects all the vertices without any cycles and with the minimum cumulative edge weights. In other words, determine a spanning tree that minimizes the cumulative edge weights.

***Reminder:*** How many edges will a minimum spanning tree have?

How would you go about solving this problem?

## 3 Kruskal's Algorithm

Kruskal's Algorithm is a greedy algorithm that chooses the smallest weight edge to include in the MST that does not create a cycle in the current MST.

***Written Steps:***

1. Sort all the edges in $E$ in non-decreasing order of their weight. Set $T = \emptyset$.
2. If $|T| < |V|$, then proceed to Step 3. Otherwise, proceed to Step 4.
3. Pick the smallest weight edge $(i, j) \in E$. If $(i, j)$ does not form a cycle among the edges in $T$, then add $(i, j)$ to $T$. Remove $(i, j)$ from $E$. Return to Step 2.
4. Terminate with a MST among those edges in $T$.

***Pseudocode:***

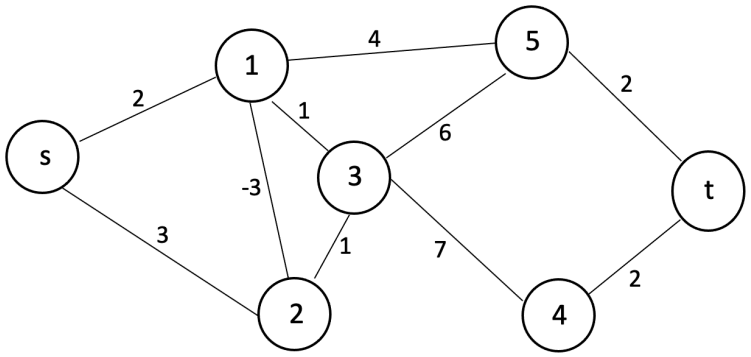
**Algorithm 1** Kruskal's Algorithm

```
Sort all the edges in E in non-decreasing order of their weight.
Set T = ∅.
while |T| < |V| do
  Pick the smallest weight edge (i, j) ∈ E.
  if (i, j) does not form a cycle among the edges in T then
    Add (i, j) to T.
  end if
  Remove (i, j) from E
end while
```

How would you sort the edges in $E$ according to their weight?

How would you check to see if adding $(i, j)$ to $T$ would create a cycle?

For the network below, determine the MST using Kruskal's Algorithm.



# 4 Prim's Algorithm

Like Kruskal's Algorithm, Prim's Algorithm is a greedy approach. It starts with an empty spanning tree that maintains two sets of nodes. The former set contains the nodes included in the MST, and the latter set contains the nodes not yet included in the MST. At each step, Prim's considers all the edges that connect the two sets, and picks the minimum weight edges from among these edges. After picking the edge, Prim's moves one node from the latter set to the former set. The group of edges that connects the two sets is referred to as a *cut*.

***Written Steps:***

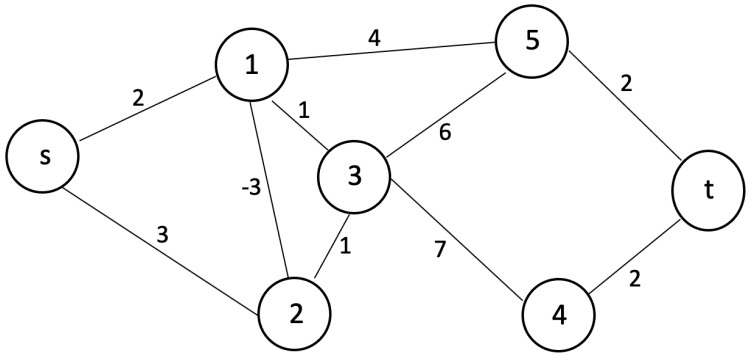
1. Initialize $T = \emptyset$.
2. While $|T| < |V|$:
   (a) Pick a edge $(u, v) \in E,\ u \in T$, and $v \in V \setminus T$ for which $c_{uv} = \min\{c_{iv} : i \in T\}$.
   (b) Add $v$ to $T$ and set $parent(v) = u$.

***Pseudocode:***

**Algorithm 2** Prim's Algorithm

```
Initialize T = ∅.
while |T| < |V| do
  Pick edge (u, v) ∈ E, u ∈ T, and v ∈ V \ T for which c_uv = min{c_iv : i ∈ T}
  Add v to T.
  Set parent(v) = u.
end while
```

For the network below, determine the MST using Prim's Algorithm.



## 5 Kruskal's vs. Prim's

- Kruskal's Algorithm tends to run faster in sparse graph, while Prim's Algorithm tends to run faster in dense graphs.
- Prim's starts to build the MST from any node in the graph, while Kruskal's builds a MST from the vertex carruying minimum weight in the graph.
- Kruskal's Algorithm traverses one node only once, while Prim's Algorithm traverse one node more than once time to the get the minimum distance.
- Prim's Algorithm gives connected components and works only on connected graphs, while Kruskal's Algorithm can generate disconnected components.
- Prim's Algorithm runs in $O(V^2)$ time, and Kruskal's Algorithm runs in $O(E \log(V))$ time.