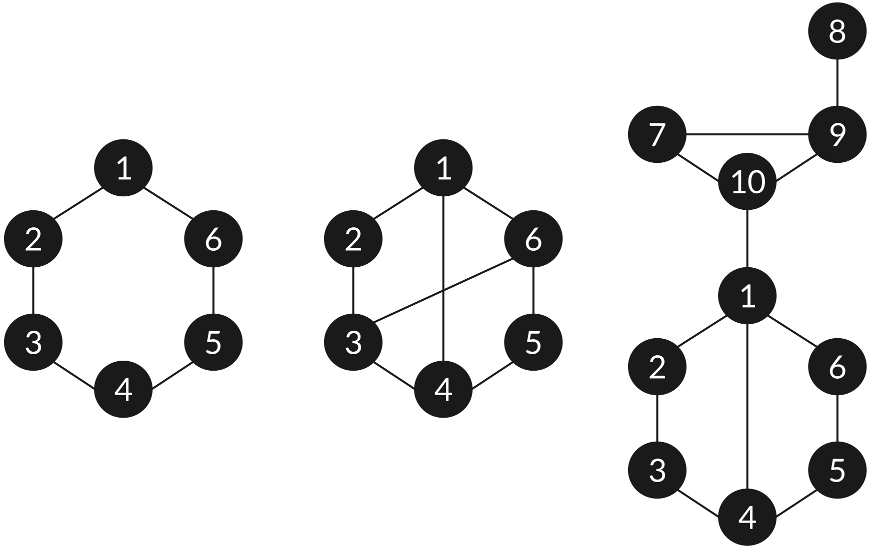
CIS 263 Introduction to Data Structures and Algorithms

Graph Algorithms (Spanning Trees)

Spanning Tree

$G = (V, E)$
$V = \{1, 2, 3, 4, 5, 6\}$
$E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}, \{6, 1\}\}$

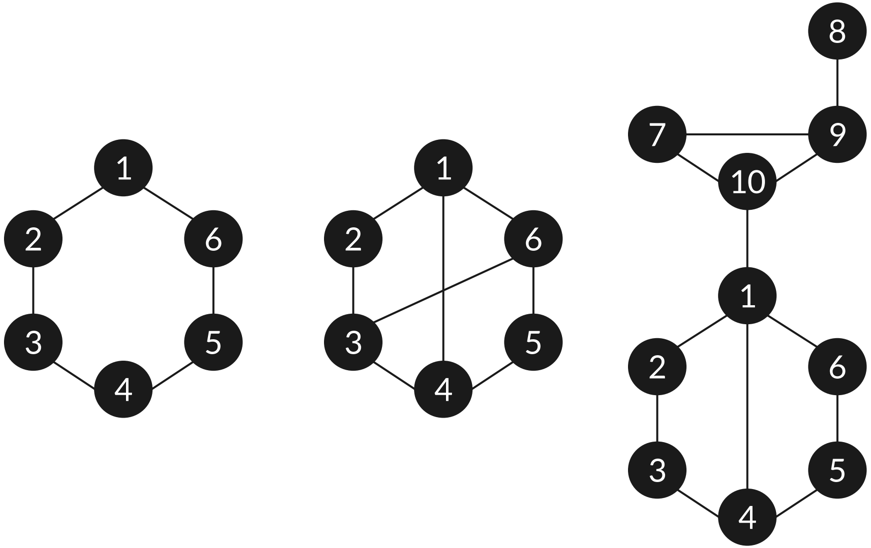


Graph with Cycles

Spanning Tree

G = (V, E)
V = {1, 2, 3, 4, 5, 6}
E = {{1, 2}, {2, 3}, {3, 4}, {4, 5}, {5, 6}, {6, 1}}

- We want to get rid of cycles in a graph
- Many applications:
  - TSP for an example
  - Networking and networking algorithms

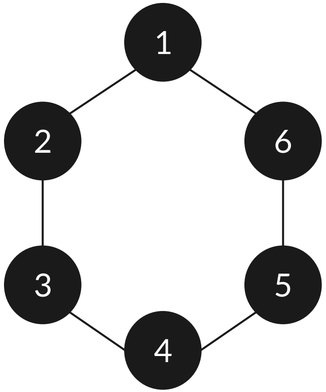


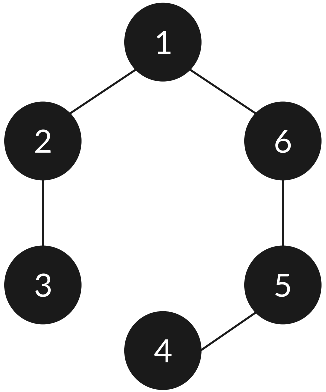
Graph with Cycles

Spanning Tree

G = (V, E)
V = {1, 2, 3, 4, 5, 6}
E = {{1, 2}, {2, 3}, {3, 4}, {4, 5}, {5, 6}, {6, 1}}

- We want to get rid of cycles in a graph
- Many applications:
  - TSP for an example
  - Networking and networking algorithms
- **For this particular example, you can drop any edges**

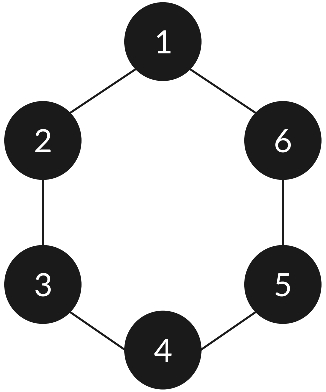


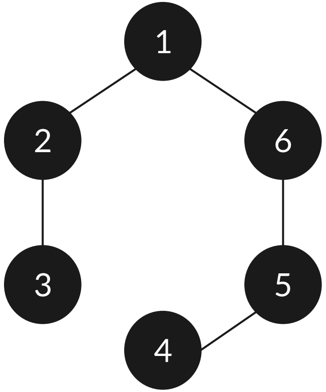


Spanning Tree

$G = (V, E)$
$V = \{1, 2, 3, 4, 5, 6\}$
$E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}, \{6, 1\}\}$

- We want to get rid of cycles in a graph
- Many applications:
  - TSP for an example
  - Networking and networking algorithms
- For this particular example, you can drop any edges
- **A Spanning Tree is a sub graph (of its parent graph)**





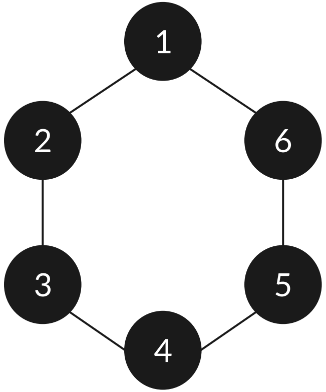
$S = (V', E')$
$V' = V = \{1, 2, 3, 4, 5, 6\}$
$E' = \{\{1, 2\}, \{2, 3\}, \cancel{\{3, 4\}}, \{4, 5\}, \{5, 6\}, \{6, 1\}\}$

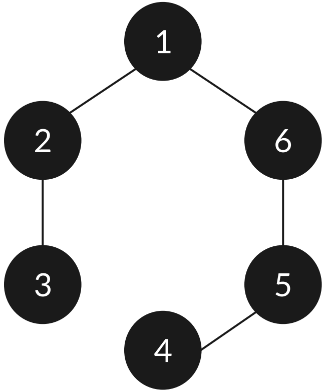
$|V| = 6$
$|E'| = |V| - 1$

Spanning Tree

$G = (V, E)$
$V = \{1, 2, 3, 4, 5, 6\}$
$E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}, \{6, 1\}\}$

- We want to get rid of cycles in a graph
- Many applications:
  - TSP for an example
  - Networking and networking algorithms
- For this particular example, you can drop any edges
- A Spanning Tree is a sub graph (of its parent graph)
- **Nb of possibilities (for this particular example): *6***



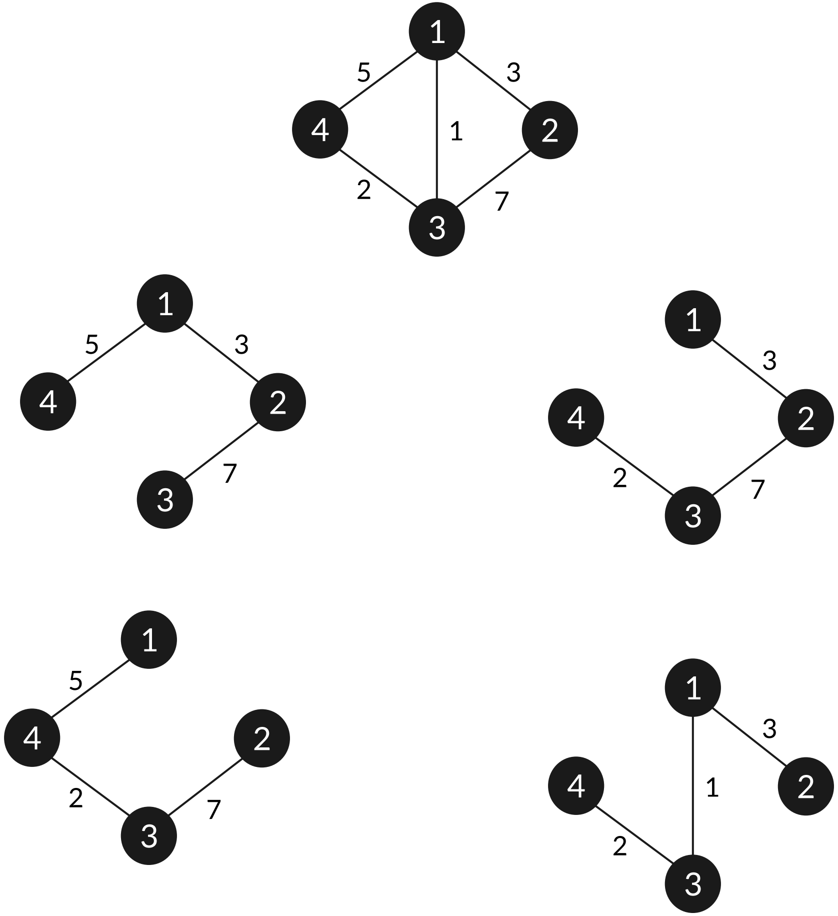


$S = (V', E')$
$V' = V = \{1, 2, 3, 4, 5, 6\}$
$E' = \{\{1, 2\}, \{2, 3\}, \cancel{\{3, 4\}}, \{4, 5\}, \{5, 6\}, \{6, 1\}\}$

$|V| = 6$
$|E'| = |V| - 1$

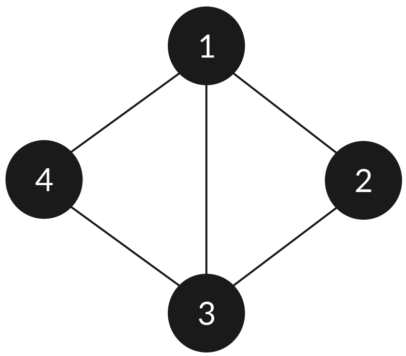
Minimum Spanning Tree

- We have to find spanning Trees with the minimum cost
  - Search all combinations
  - Approximation Algorithms
    - Greedy Algorithms
      - Prim’s Algorithm
      - Kruskal’s Algorithm



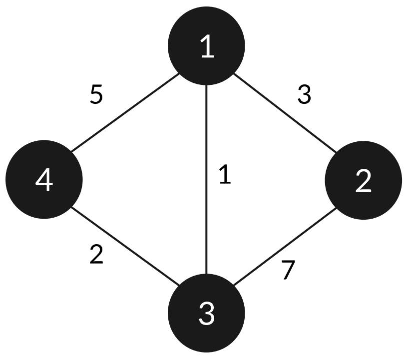
Weighted Graph

- We want to travel from (2) to (4)
- We have different paths
- **Counting the number of edges** can be assumed as an optimization metric.



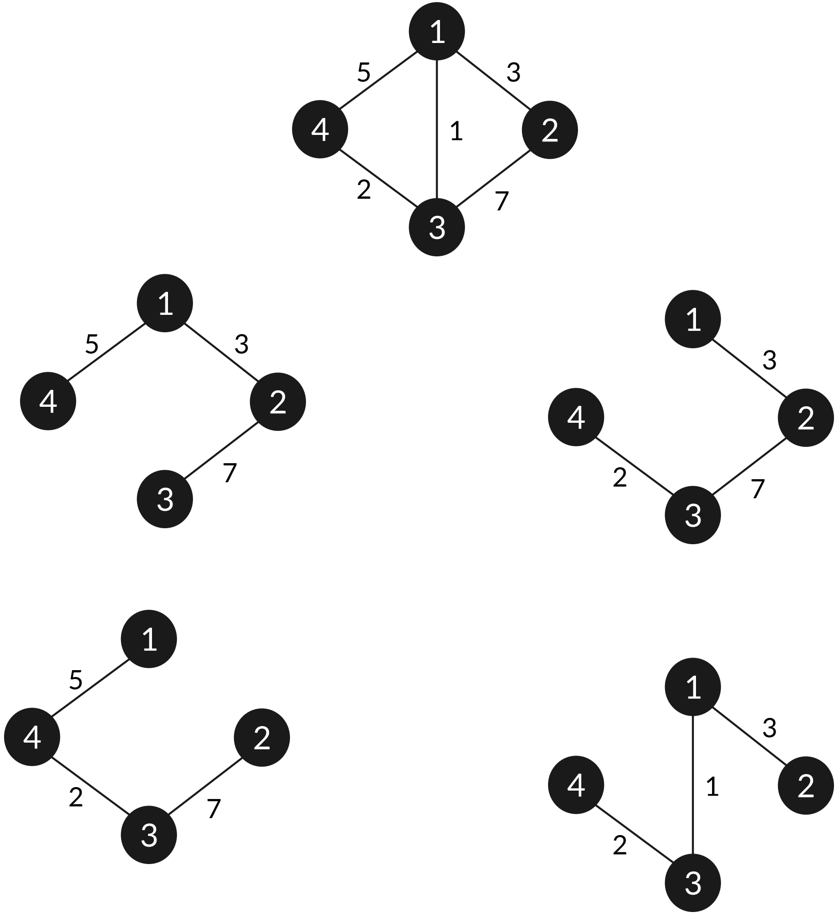
Weighted Graph

- We want to travel from (2) to (4)
- We have different paths
- What if explicit weights are associated
    - (2, 1, 3, 4), 3 edges but cost is 6 (the minimum)



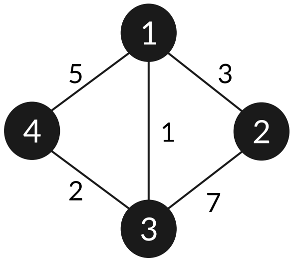
Minimum Spanning Tree

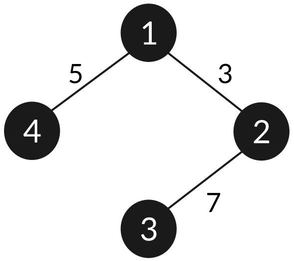
- We have to find spanning Trees with the minimum cost
    - Search all combinations
    - Approximation Algorithms
        - Greedy Algorithms
            - Prim’s Algorithm
            - Kruskal’s Algorithm

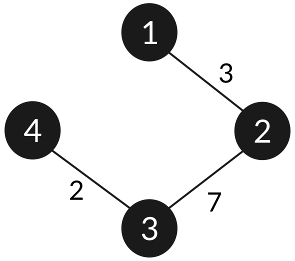


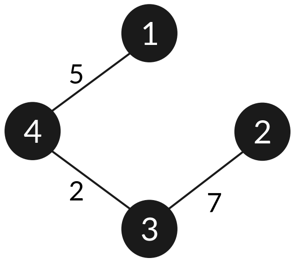
Minimum Spanning Tree

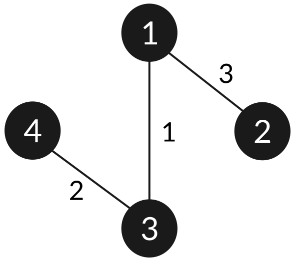
- We have to find spanning Trees with the minimum cost
  - Search all combinations
  - Approximation Algorithms
    - **Greedy Algorithms**
      - Prim's Algorithm
      - Kruskal's Algorithm





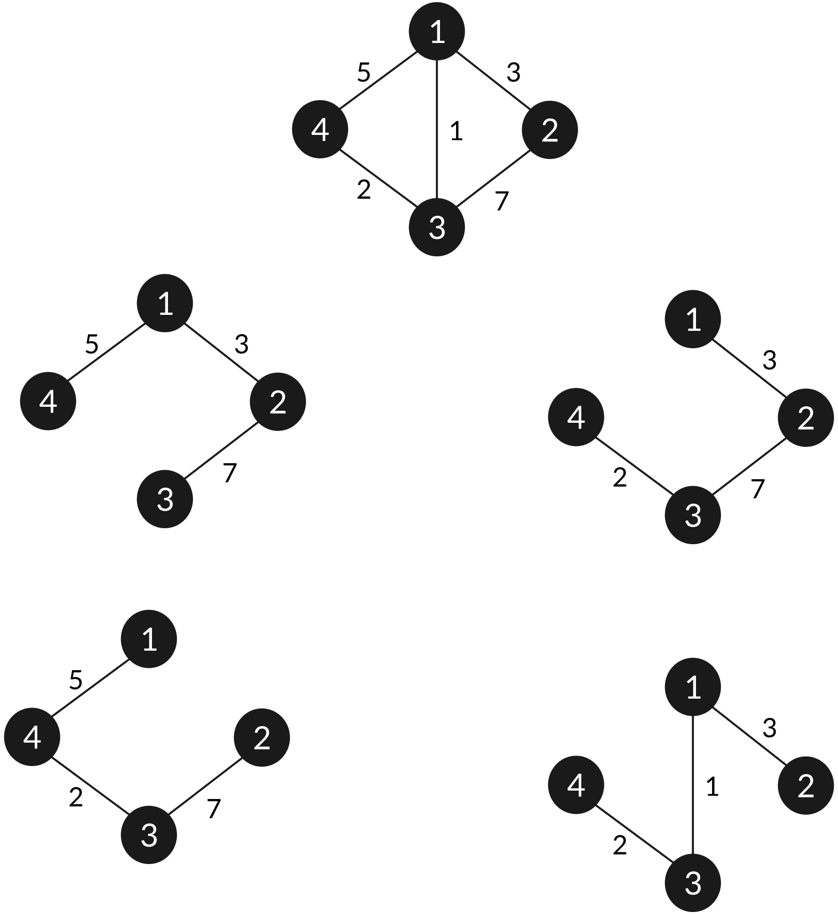






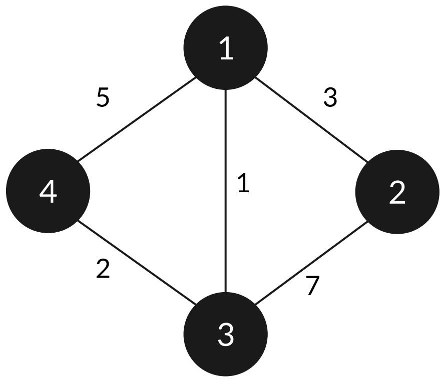
Minimum Spanning Tree

- We have to find spanning Trees with the minimum cost
    - Search all combinations
    - Approximation Algorithms
        - **Greedy Algorithms**
            - Prim's Algorithm
            - Kruskal's Algorithm



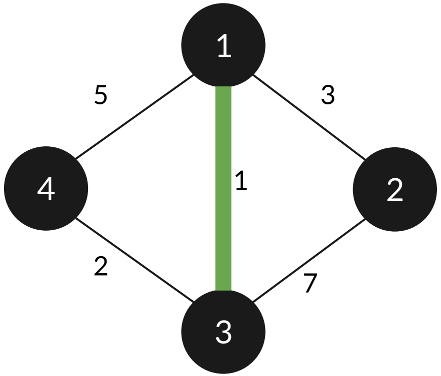
- Uses heuristics
- Solution may not be optimal

Prim’s Algorithm



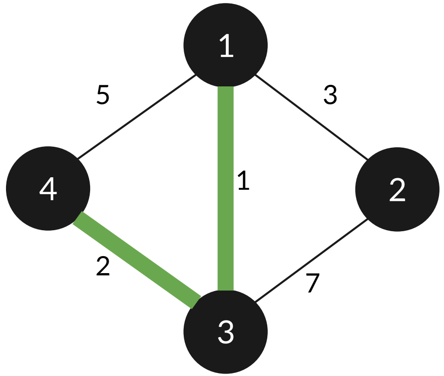
Prim’s Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (but form the connected set)
  - Iterate but avoid loops



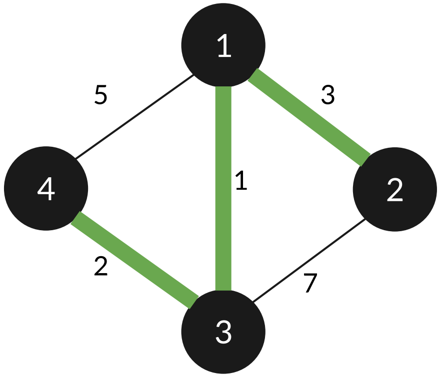
Prim’s Algorithm

- Start with the minimum cost edge
- **Pick the next minimum cost edge (but form the connected set)**
  - Iterate but avoid loops



Prim’s Algorithm

- Start with the minimum cost edge
- **Pick the next minimum cost edge (but form the connected set)**
    - Iterate but avoid loops

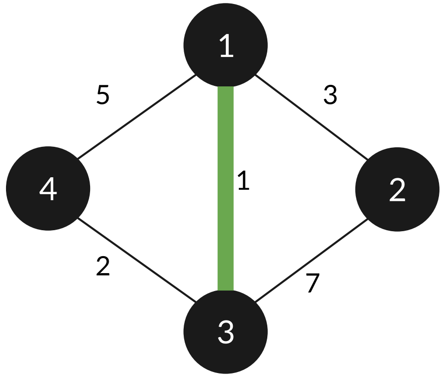


S = (*V*′, *E*′) ⊂ (*V*, *E*)
|*V*| = |*V*′| = *4*
|*E*′| = |V| - 1

Cost: 2 + 1 + 3 = 6

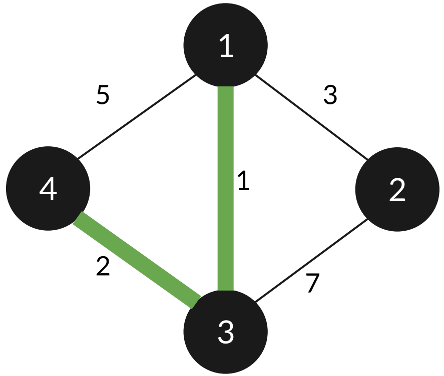
Kruskal's Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
  - Iterate but avoid loops



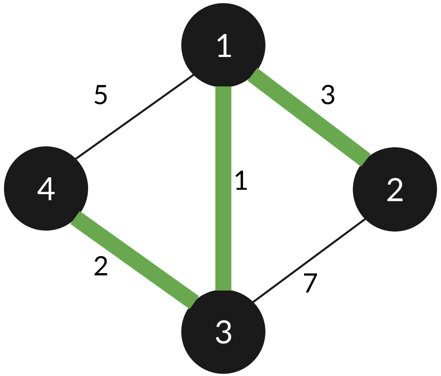
Kruskal's Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
  - Iterate but avoid loops



Kruskal's Algorithm

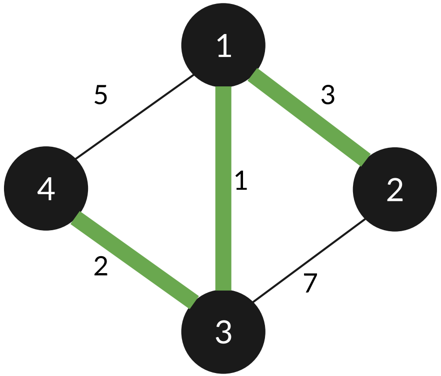
- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
  - Iterate but avoid loops



$S = (V', E') \subset (V, E)$
$|V| = |V'| = 4$
$|E'| = |V| - 1$

Kruskal’s Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
    - Iterate but avoid loops

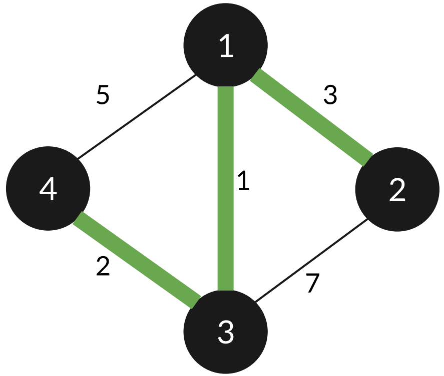


$S = (V', E') \subset (V, E)$
$|V| = |V'| = 4$
$|E'| = |V| - 1$

Cost: 2 + 1 + 3 = 6

Kruskal's Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
    - Iterate but avoid loops



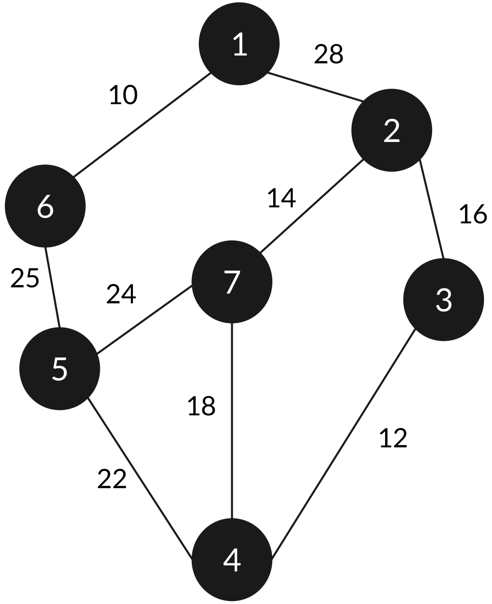
$S = (V', E') \subset (V, E)$
$|V| = |V'| = 4$
$|E'| = |V| - 1$

- Exactly same solution
- Same order

Lets try a little complex one

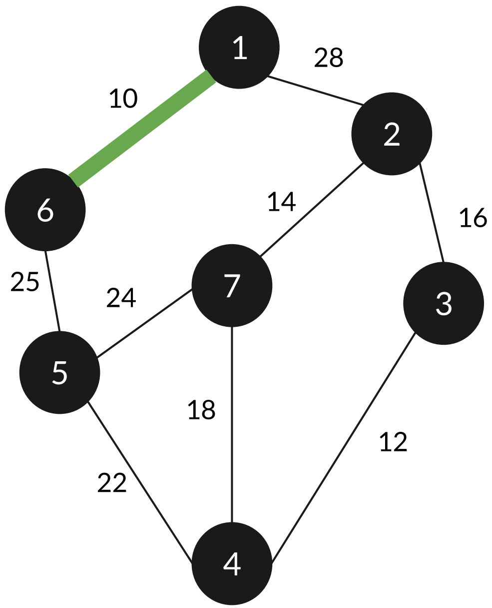
Prim's Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (but form the connected set)
    - Iterate but avoid loops



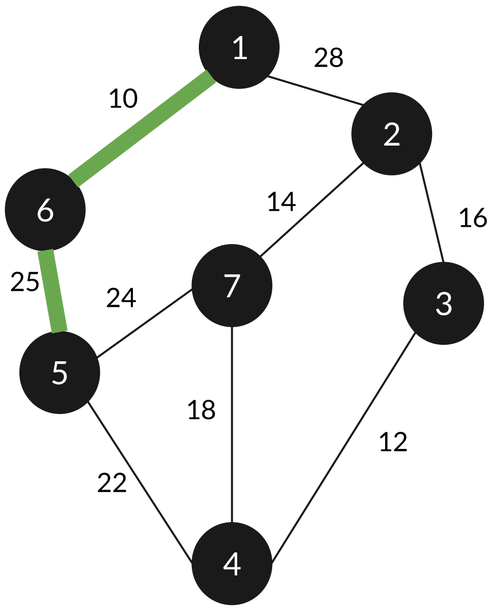
Prim’s Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (but form the connected set)
    - Iterate but avoid loops



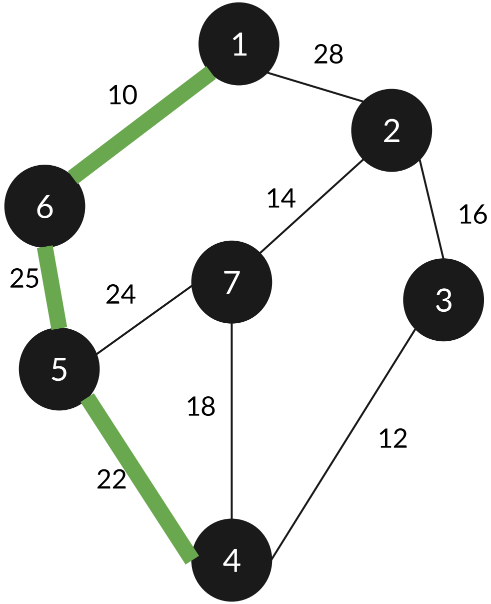
Prim's Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (but form the connected set)
    - Iterate but avoid loops



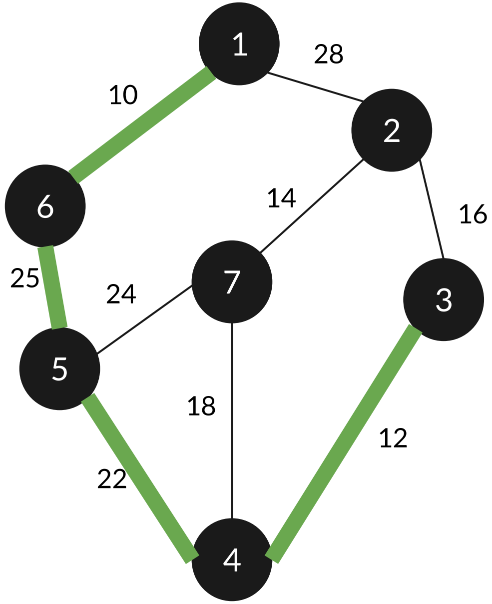
Prim’s Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (but form the connected set)
    - Iterate but avoid loops



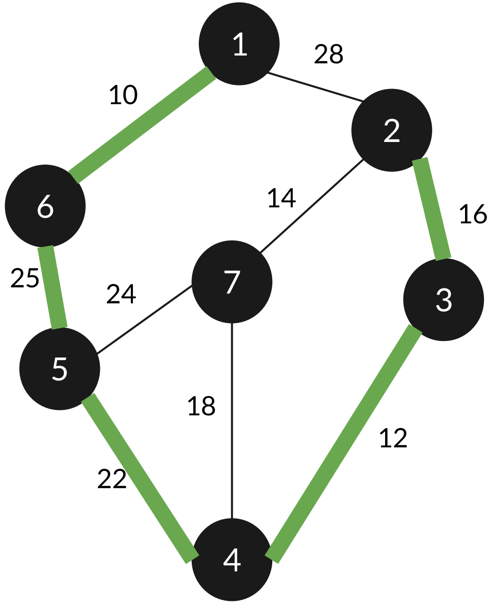
Prim's Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (but form the connected set)
    - Iterate but avoid loops



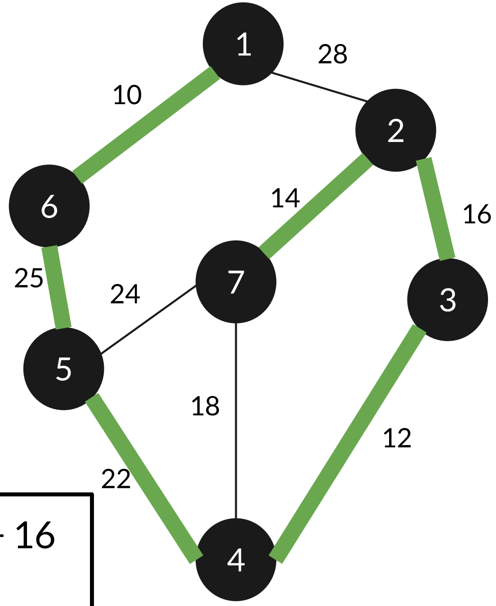
Prim’s Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (but form the connected set)
    - Iterate but avoid loops



Prim’s Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (but form the connected set)
    - Iterate but avoid loops

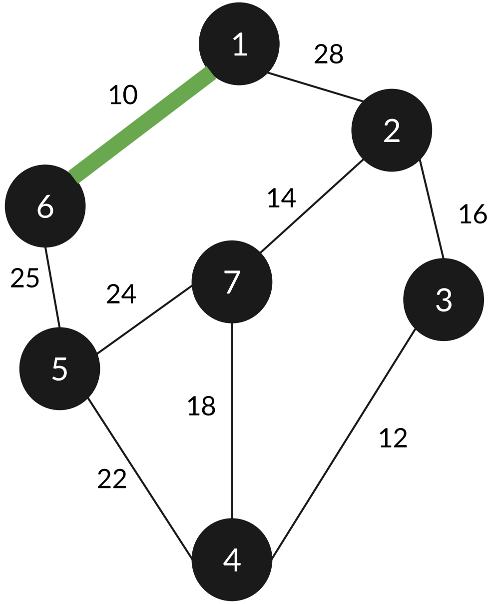


$S = (V', E') \subset (V, E)$
$|V| = |V'| = 7$
$|E'| = |V| - 1$

Cost: 10 + 25 + 22 + 12 + 16 + 14 = 99

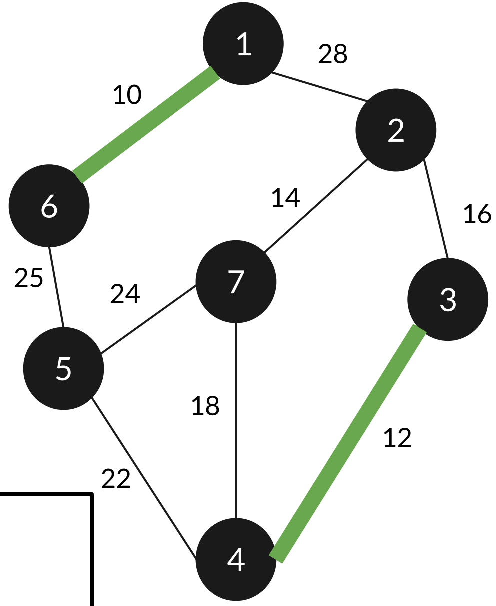
Kruskal's Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
    - Iterate but avoid loops



Kruskal’s Algorithm

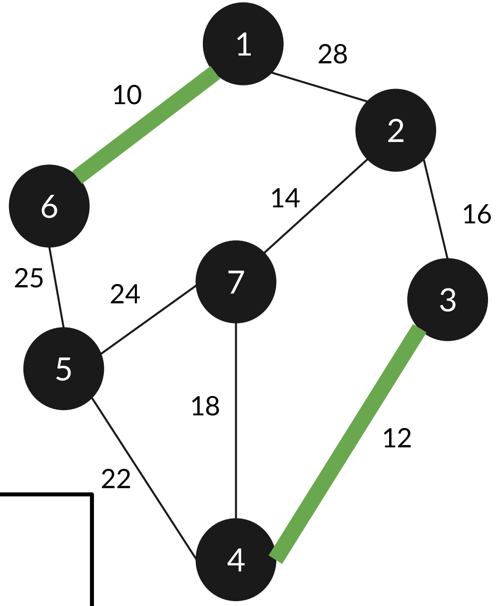
- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
  - Iterate but avoid loops



Cost: 10

Kruskal's Algorithm

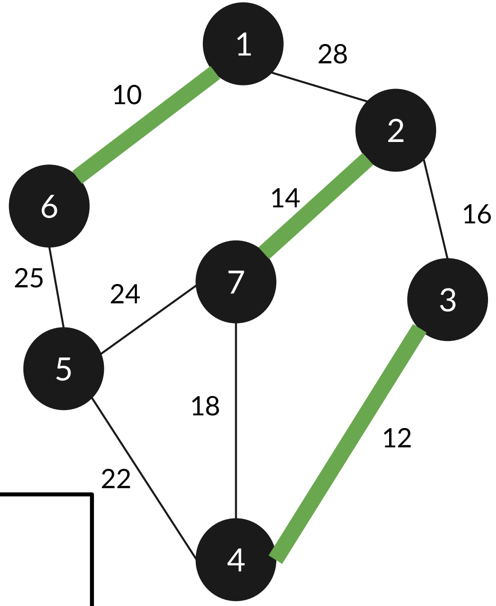
- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
  - Iterate but avoid loops



Cost: 10 + 12

Kruskal's Algorithm

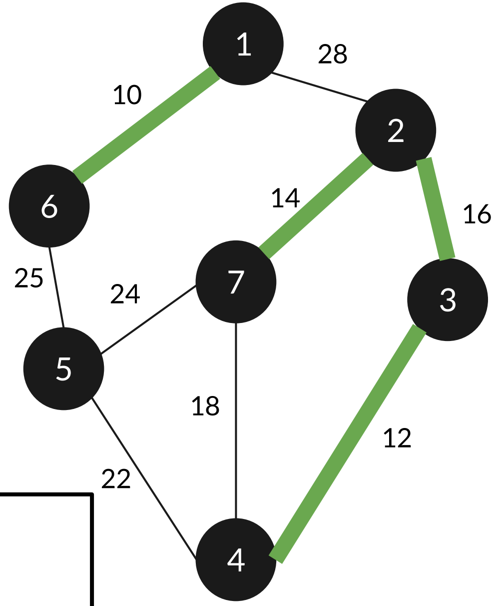
- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
  - Iterate but avoid loops



Cost: 10 + 12 + 14

Kruskal’s Algorithm

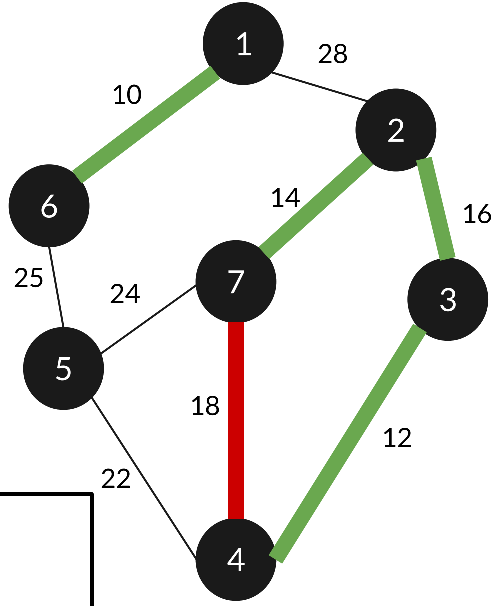
- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
    - Iterate but avoid loops



Cost: 10 + 12 + 14 + 16

Kruskal's Algorithm

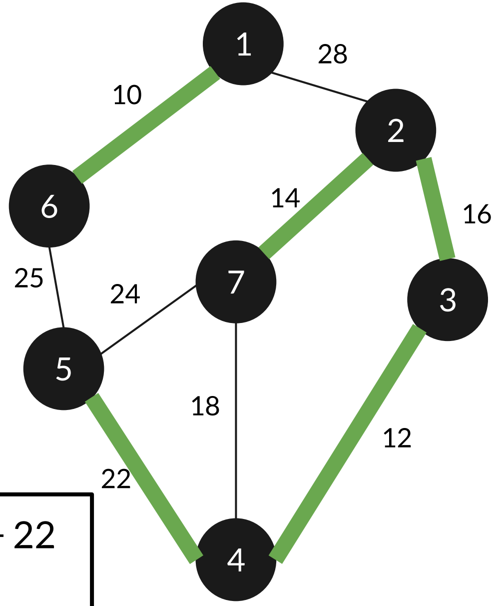
- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
    - Iterate but avoid loops



Cost: 10 + 12 + 14 + 16

Kruskal's Algorithm

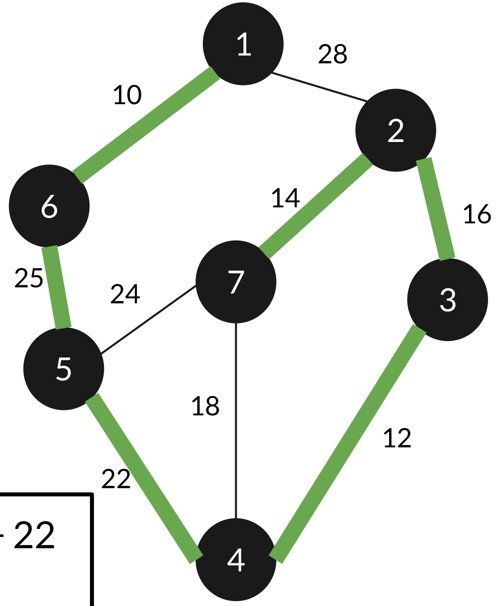
- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
  - Iterate but avoid loops



Cost: 10 + 12 + 14 + 16 + 22

Kruskal's Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
  - Iterate but avoid loops

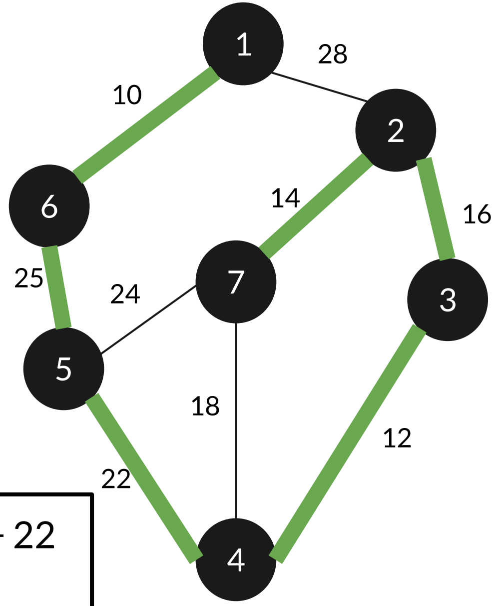


$S = (V', E') \subset (V, E)$
$|V| = |V'| = 4$
$|E'| = |V| - 1$

Cost: 10 + 12 + 14 + 16 + 22 + 25 = 99

Kruskal’s Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
    - Iterate but avoid loops

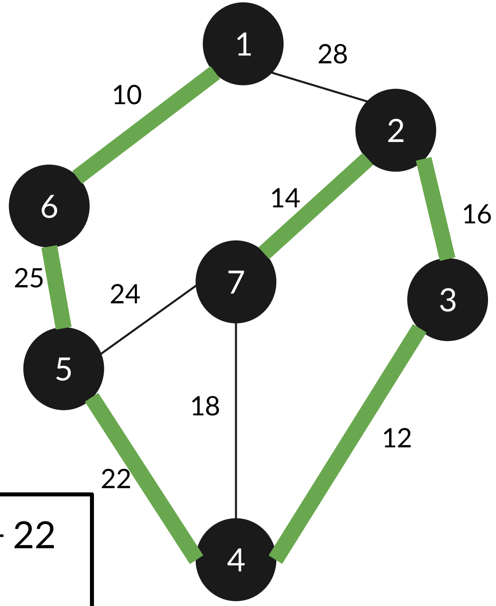


- Exactly same solution
- **Different order**

Cost: 10 + 12 + 14 + 16 + 22 + 25 = 99

Kruskal’s Algorithm

- **Start with the minimum cost edge**
- Pick the next minimum cost edge (not necessarily form the connected set)
    - Iterate but avoid loops



- Exactly same solution
- **Different order**

Cost: 10 + 12 + 14 + 16 + 22 + 25 = 99

QA