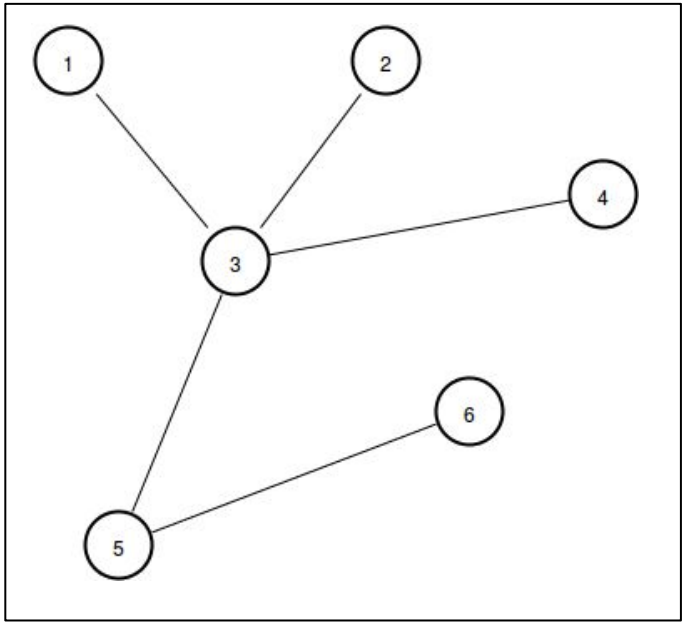
Minimum Spanning Trees

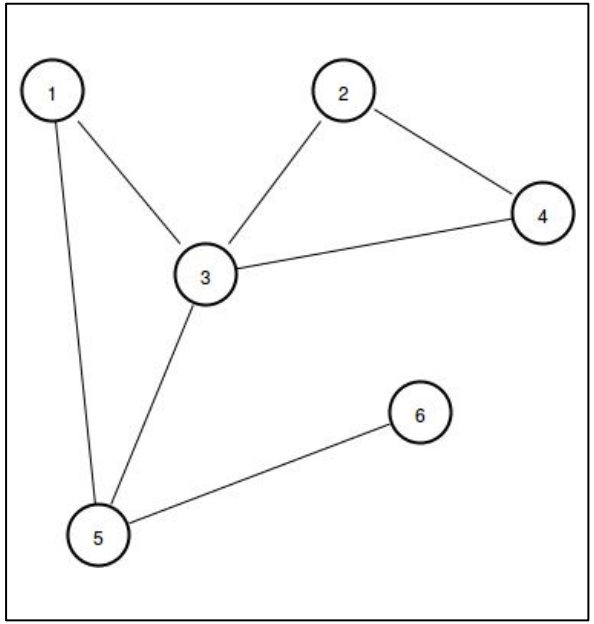
(Unrooted) Trees

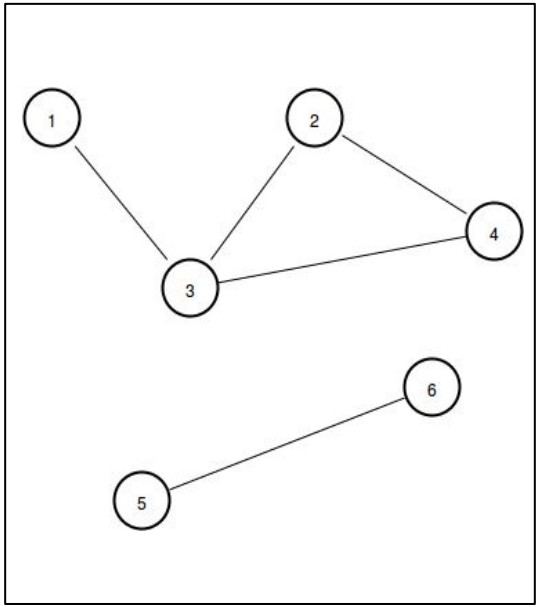
- An undirected graph with N vertices is called a *tree* if it has **N-1 edges** and **is connected**.

(Unrooted) Trees

- An undirected graph with N vertices is called a *tree* if it has **N-1 edges** and **is connected**.

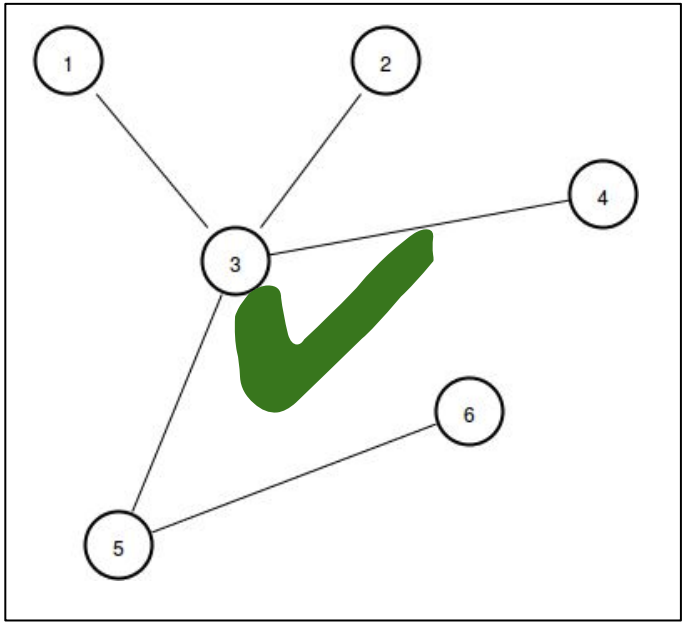


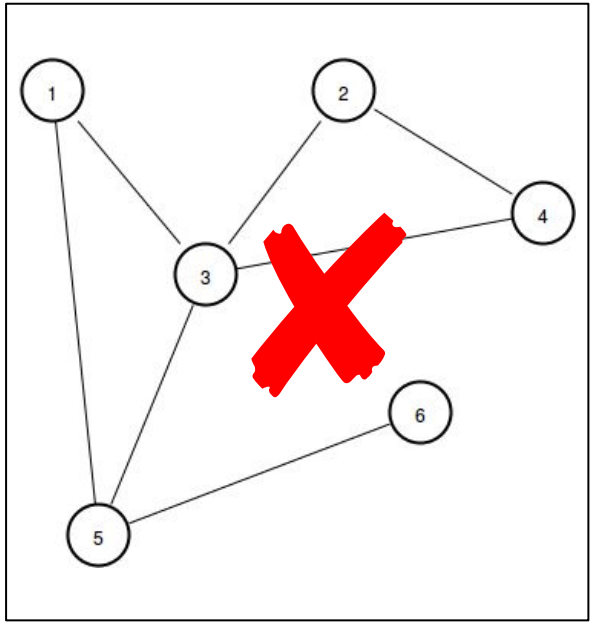


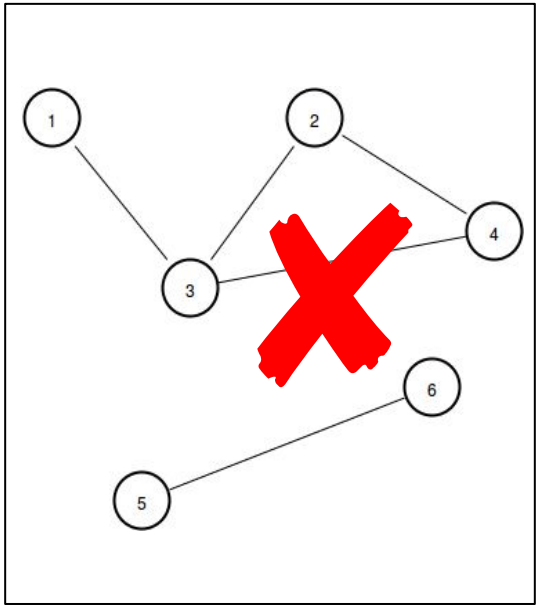


(Unrooted) Trees

- An undirected graph with N vertices is called a *tree* if it has **N-1 edges** and **is connected**.





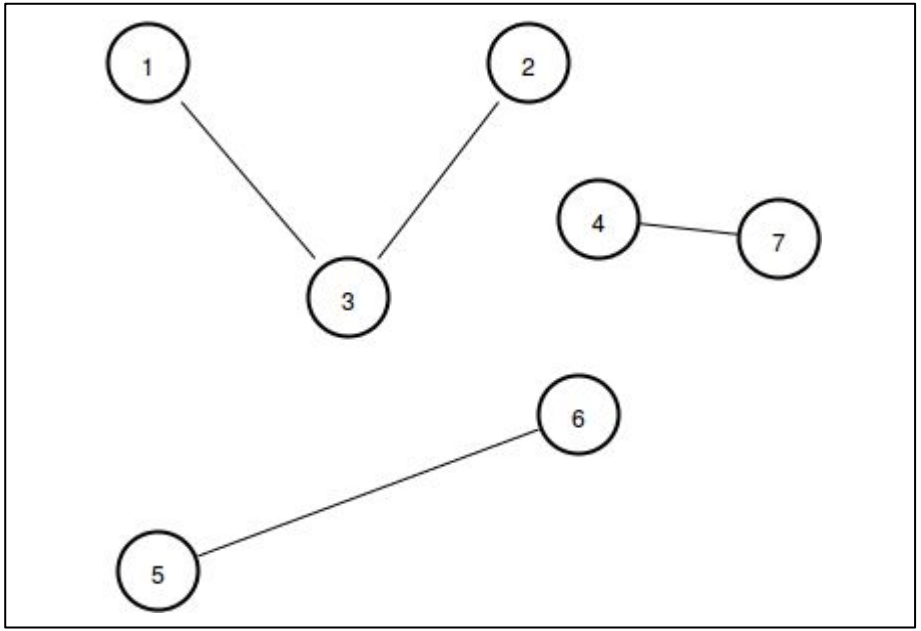


(Unrooted) Trees

- This is different from a *rooted* tree, where some node is designated as the root. With unrooted trees we don't really care about having a root or not.
  - The goal for this definition is to just generalize the idea of a rooted tree.
- Some properties of trees:
  - There is only one path between any pair of nodes
  - There are no cycles (where we are not allowed to re-use edges)

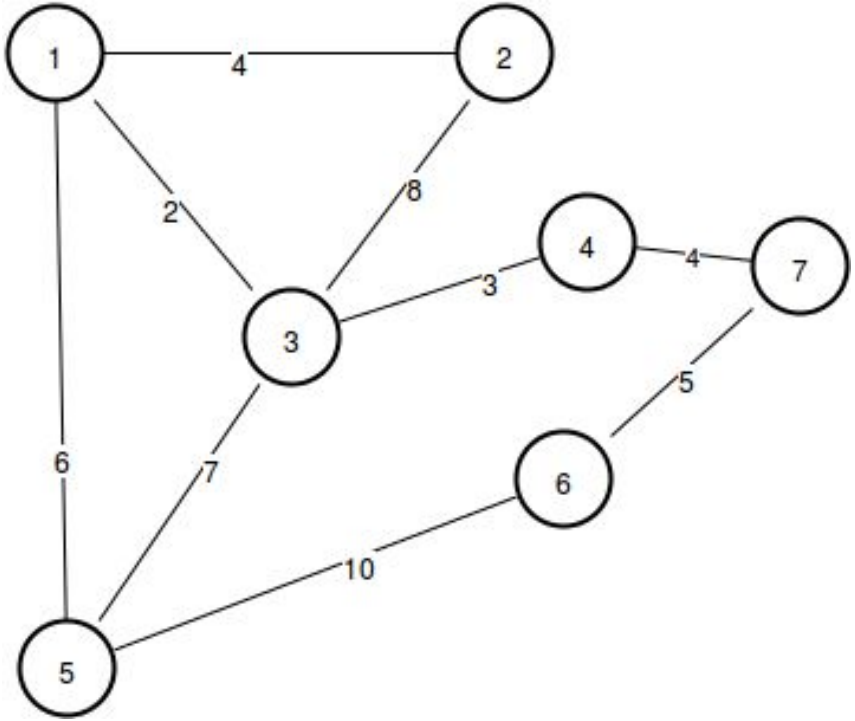
Forest

- A *forest* is a collection of (unrooted) trees.



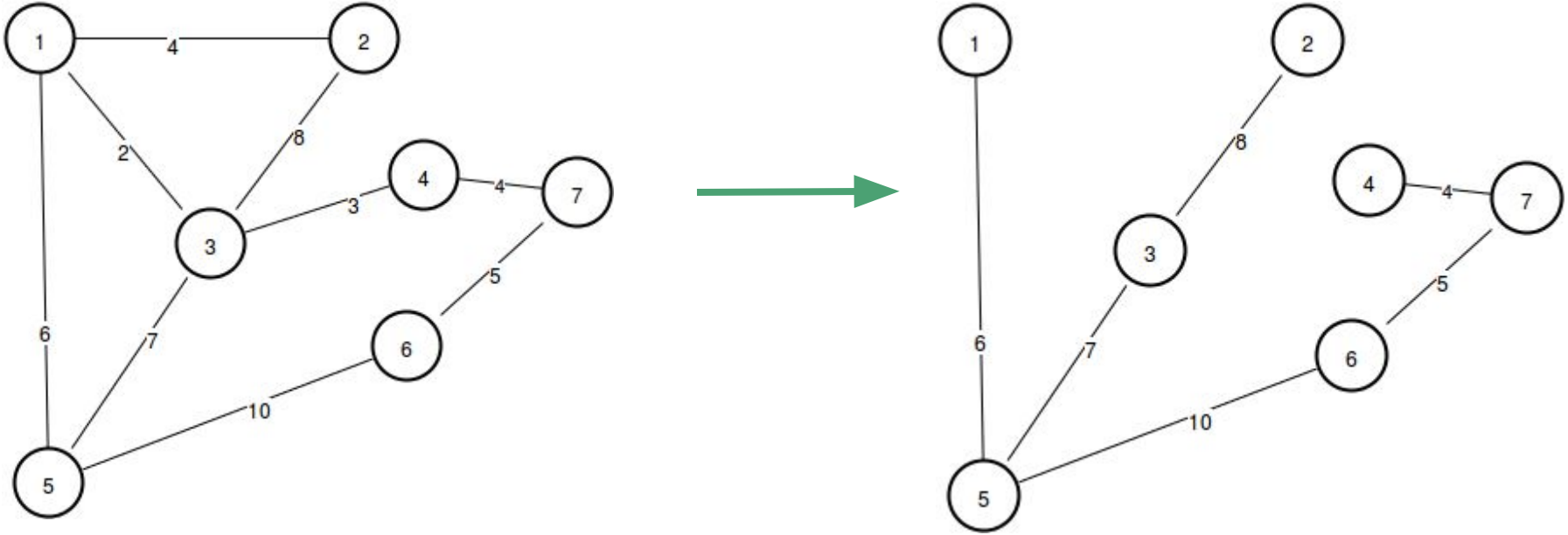
Spanning Tree

- Suppose we are given a weighted, undirected graph.



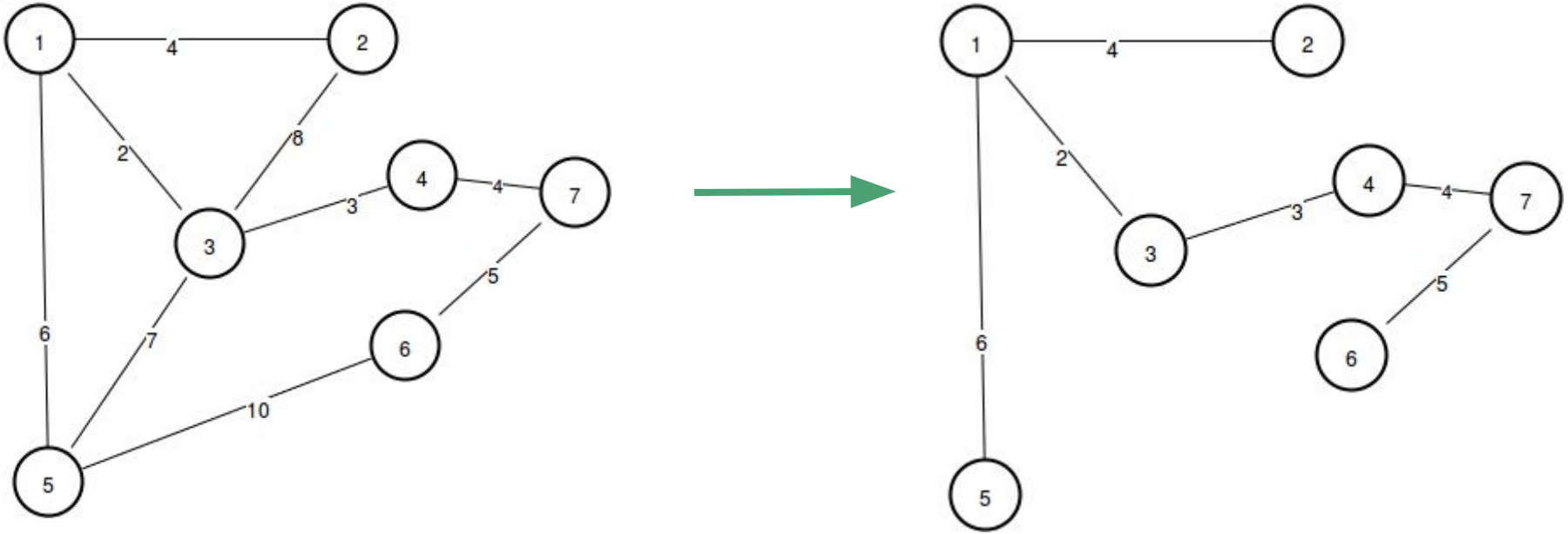
Spanning Tree

- A *spanning tree* of this graph is a subset of the vertices which forms a tree.



(Minimum) Spanning Tree

- A *minimum spanning tree* of this graph is a subset of the vertices which forms a tree and has the smallest sum of edge weights over all spanning trees.

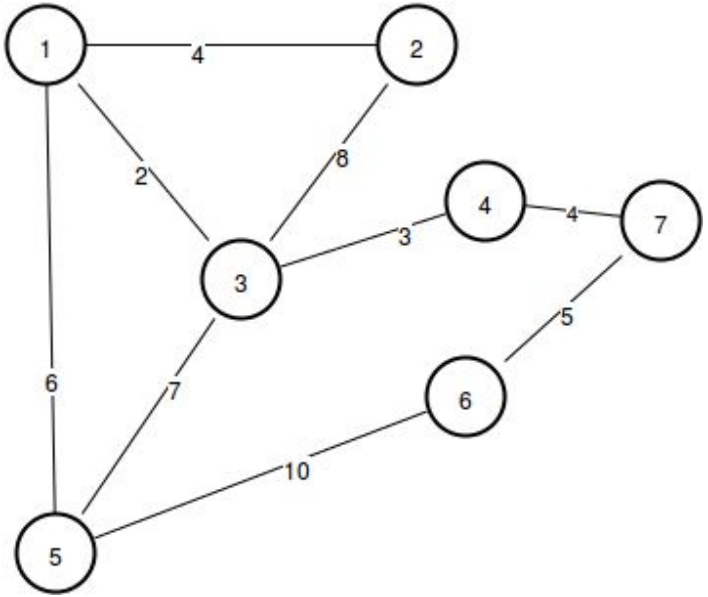


Finding Minimum Spanning Trees - Attempt #1

- **Bruteforce** - find every spanning tree, calculate its cost, and take the minimum out of all of them.
  - How do you even code this?
    - You've coded it. Now what is the runtime of this algorithm?
      - How many minimum spanning trees can there be?
        - A lot. An exponential amount: O(2^N)

Finding Minimum Spanning Trees - Attempt #2

- What to do now?
- Greedy?
  - Keep taking smallest edges?
  - ?

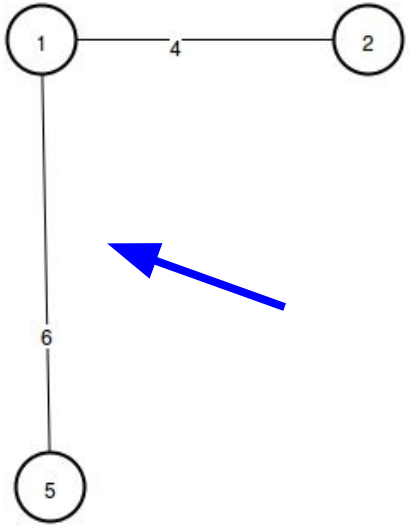


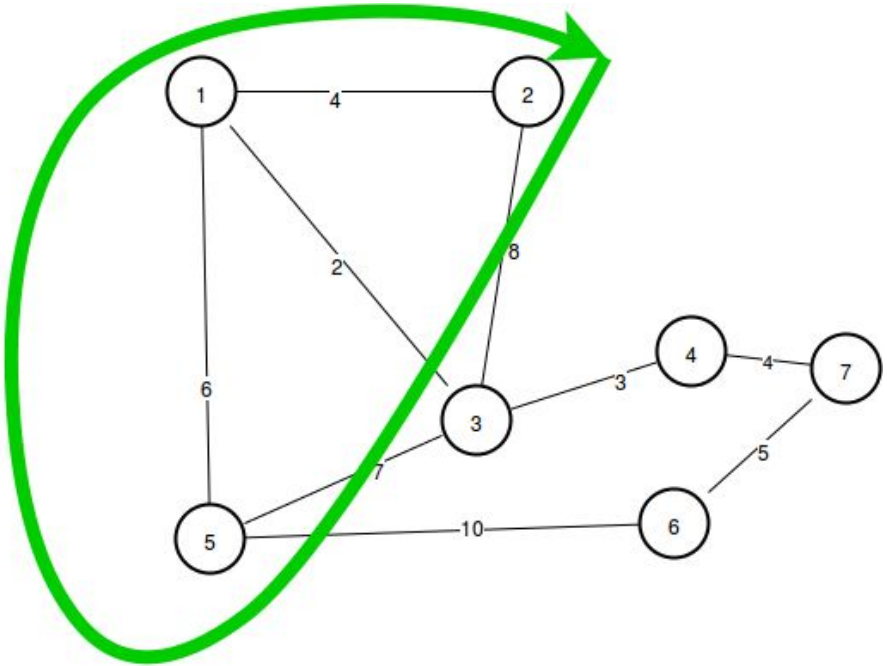
Cut Property

- In order to develop an efficient algorithm, we're going to have to make use of the ***cut property***.

Cut Property

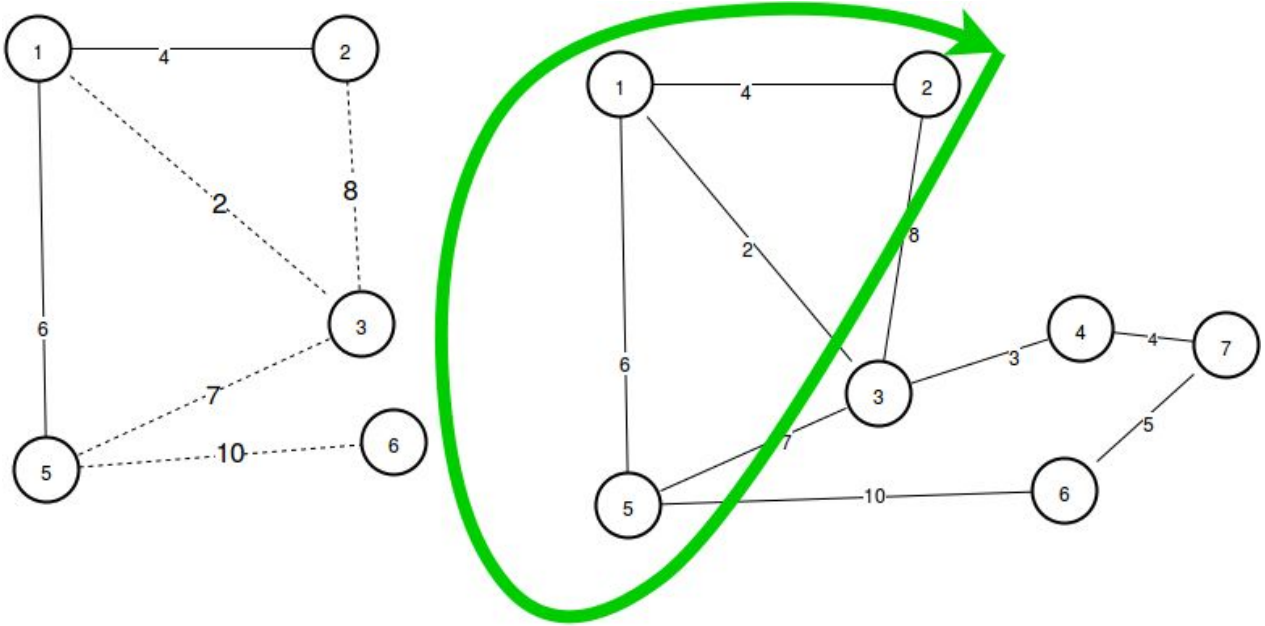
- Suppose we have a subset of vertices (and edges) which we know is a minimum spanning tree.





Cut Property

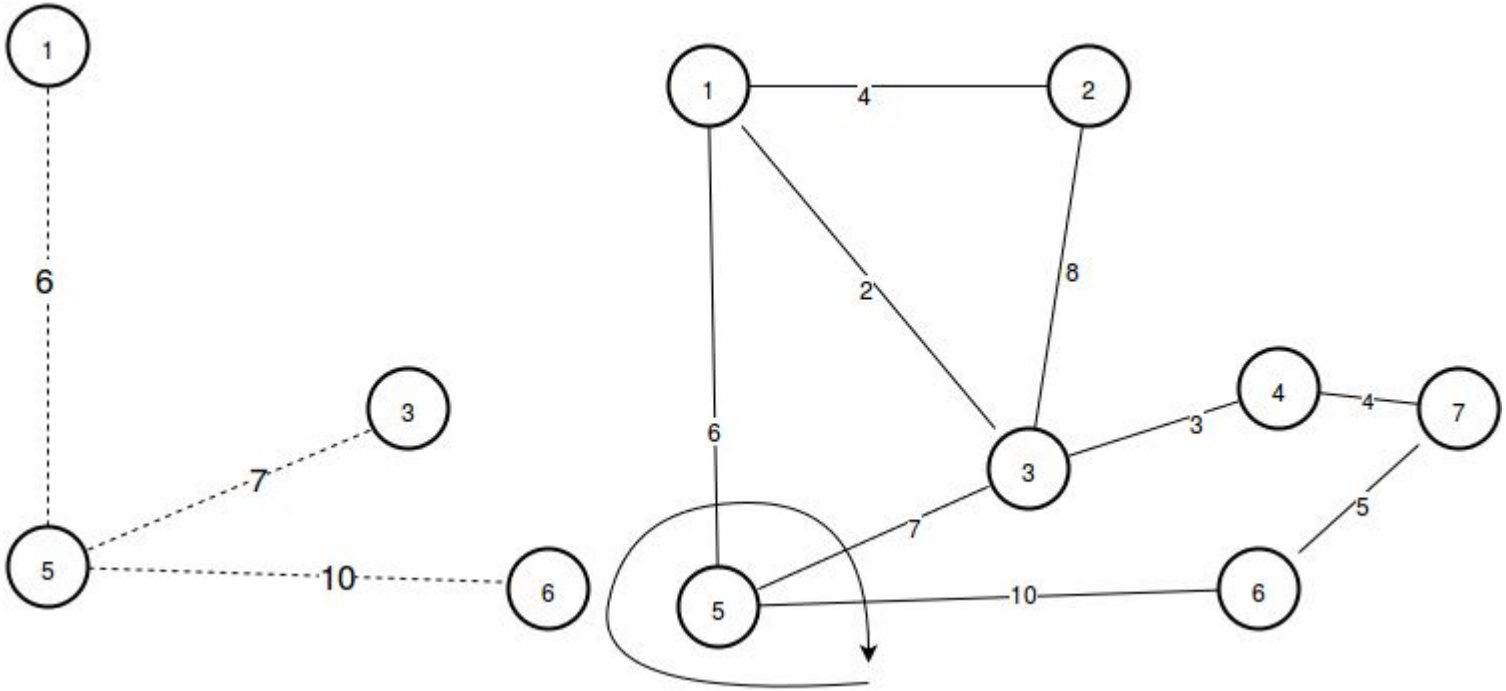
- The cut property states that the smallest edge leaving this subset ***is always in the minimum spanning tree.***



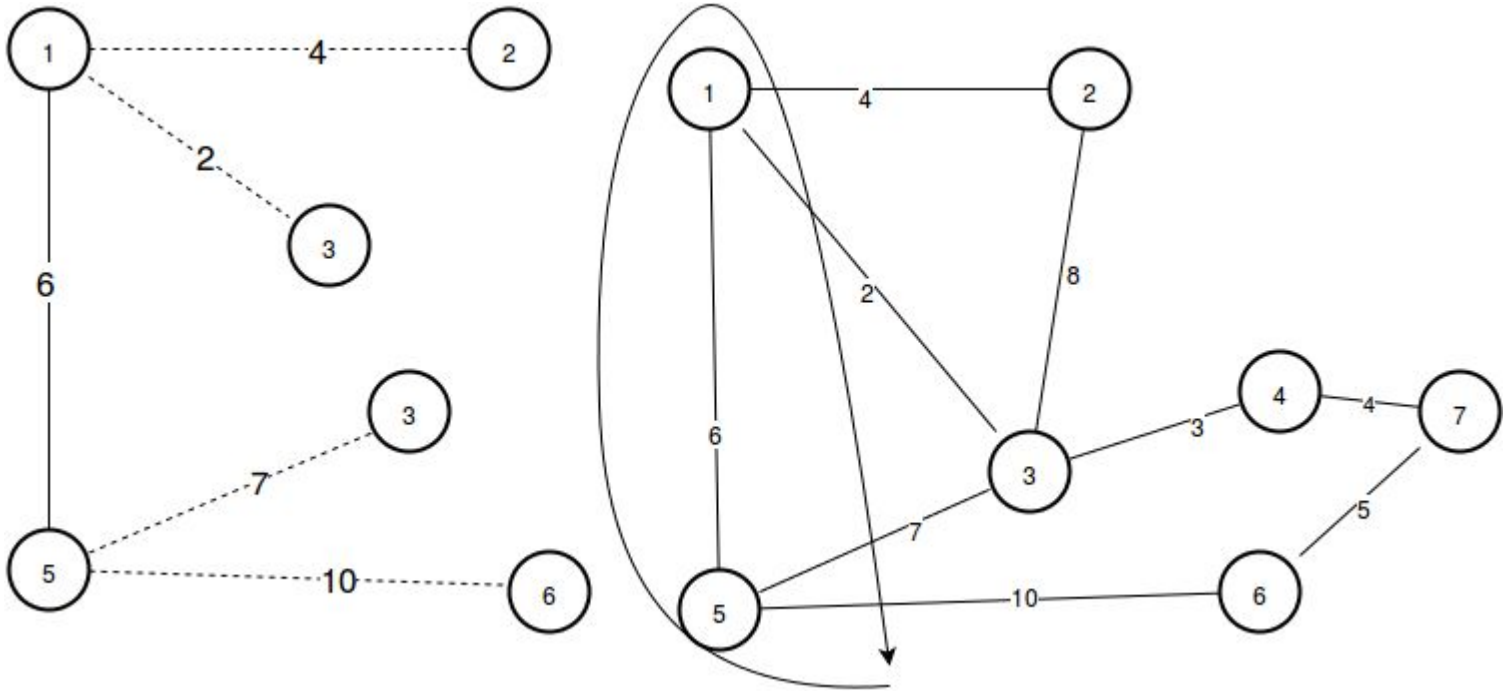
Cut Property - Building An Algorithm

- How can we use the cut property to build an MST algorithm?
- All we do is maintain a growing subset of vertices (and corresponding edges) which we know is part of the minimum spanning tree.
    - We pick an arbitrary vertex to start our subset of vertices off with. Since it's a single vertex, no edges are required to make it a spanning tree (it already is by definition!)
- To 'grow' the current subset, we look at every edge leaving the subset. We then choose the minimum one and add that edge (along with the vertex it leads to) into our subset of vertices.
    - Do this until there are no more vertices left.

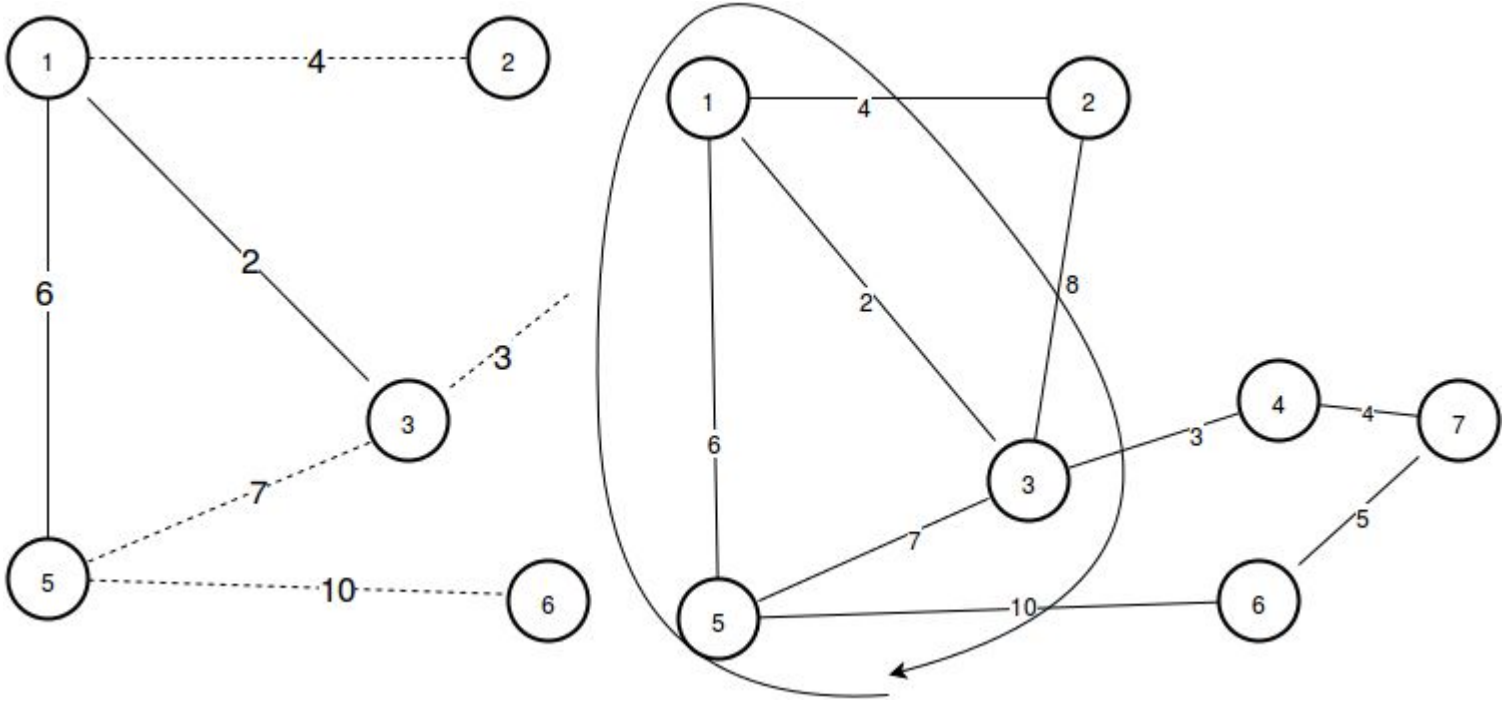
Cut Property - Building an Algorithm



Cut Property - Building an Algorithm



Cut Property - Building an Algorithm



Prim’s Algorithm

- How can we use the cut property to build an MST algorithm?
- All we do is maintain a growing subset of vertices (and corresponding edges) which we know is part of the minimum spanning tree.
    - We pick an arbitrary vertex to start our subset of vertices off with. Since it’s a single vertex, no edges are required to make it a spanning tree (it already is by definition!)
- To ‘grow’ the current subset, we look at every edge leaving the subset. We then choose the minimum one and add that edge (along with the vertex it leads to) into our subset of vertices.
    - Do this until there are no more vertices left.

How to code it?

- Try and code:
  - http://www.spoj.com/problems/CSTREET/
- If you're using Java, you may need to use PriorityQueue
- Also an array visited
  - visited[x] = true (if it's in our subset of known vertices)

Kruskal’s Algorithm

- Another algorithm for solving minimum spanning

Problems to Solve

- http://www.spoj.com/problems/CSTREET/
- https://goo.gl/vocxWP
- https://goo.gl/84Mr0A
- https://goo.gl/my8e0g
-