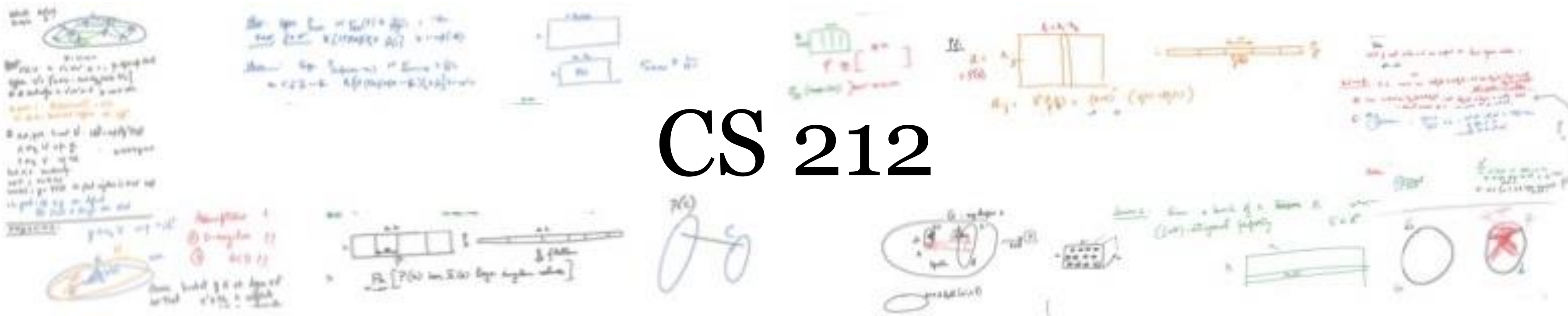


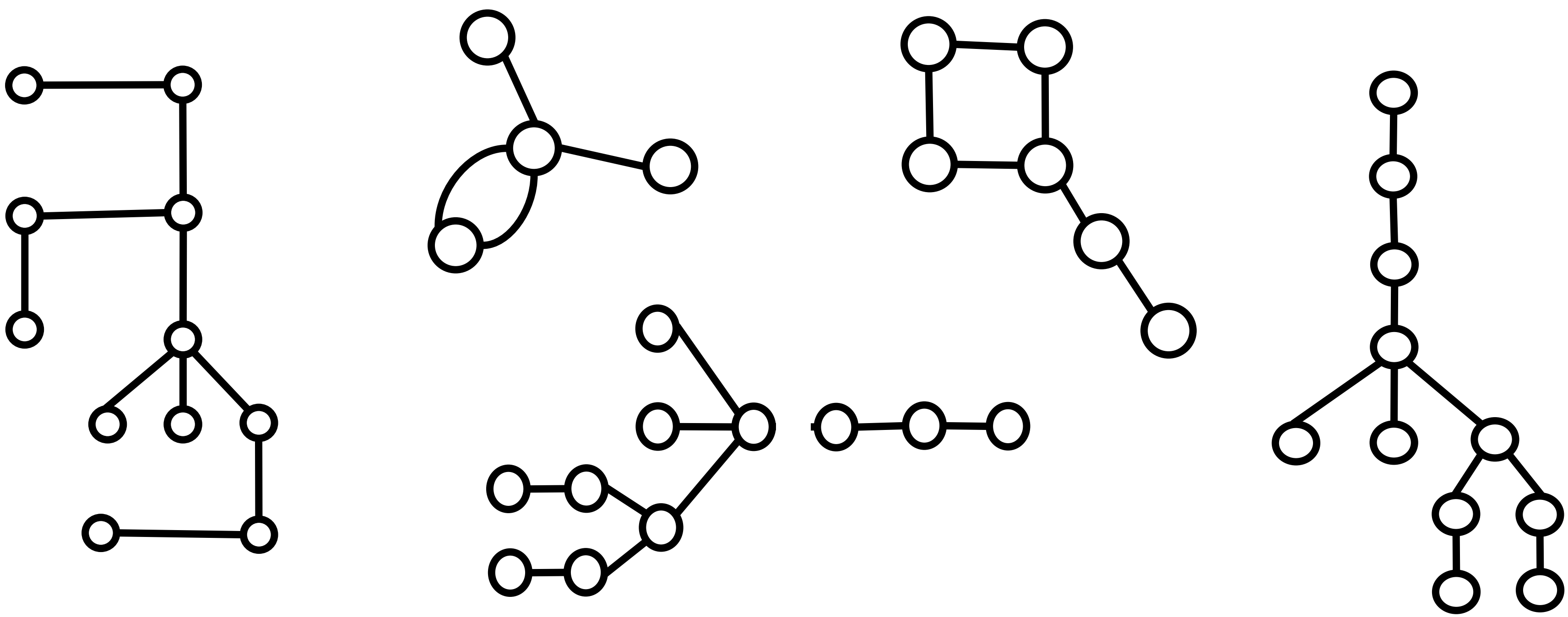
CS 212

Mathematical Foundations of Computer Science

Lecture 21: Spanning Trees

Trees

Which among the following graphs are trees?

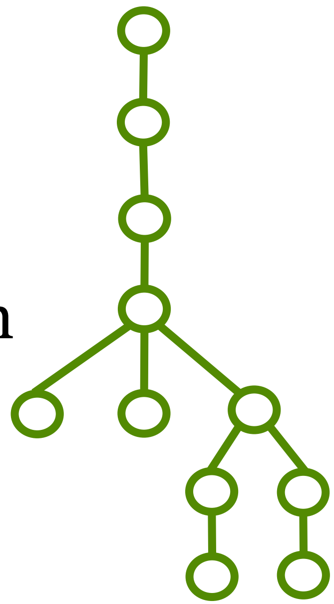


Trees: A connected graph with no cycles

Equivalent Definitions of Trees

Theorem: Let G be a graph with n vertices and m edges

The following are equivalent:

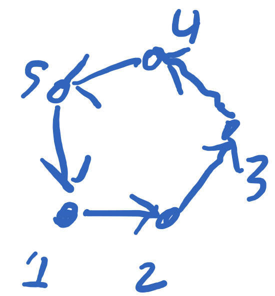


1. G is a connected and acyclic (i.e. G is a tree)
2. Every two vertices of G are joined by a unique path
3. G is connected and $m = n - 1$
4. G is acyclic and $m = n - 1$
5. G is acyclic and if any two non-adjacent nodes are joined by an edge, the resulting graph has exactly one cycle

Proof of the Equivalence

How many implications do we need to show?

$$5 \times 4 = 20?$$



To prove this, it suffices to show

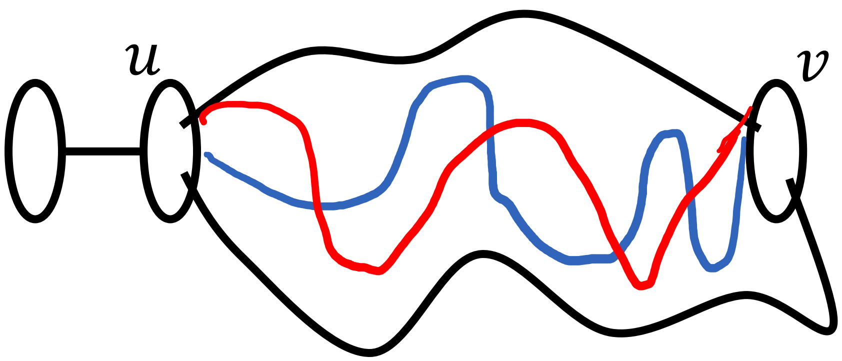
$$1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 1$$

Proof of $1 \Rightarrow 2$

Claim: If G is a tree (connected, acyclic), then every two nodes are joined by unique path.

Proof: (by contradiction). Suppose not.

Assume G is a tree that has two nodes u, v connected by two different paths:



Then there exists a cycle (formally: a closed walk. Then use PS5 #2)

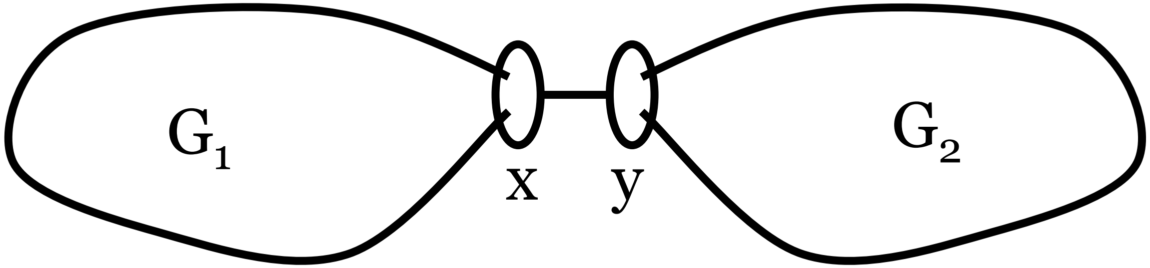
Proof of 2 ⇒ 3

Claim: If every two nodes are joined by unique path, then G is connected and $m = n - 1$.

Pf: Easy to see why connected. Prove m=n-1 by strong induction

Assume true for every graph with < n nodes. Let G have n nodes and let x and y be adjacent.

Let n_1, m_1 be number of nodes and edges in G_1
n_2, m_2 be number of nodes and edges in G_2



G_1 : graph on vertices connected to x.
G_2 : graph on vertices connected to y.

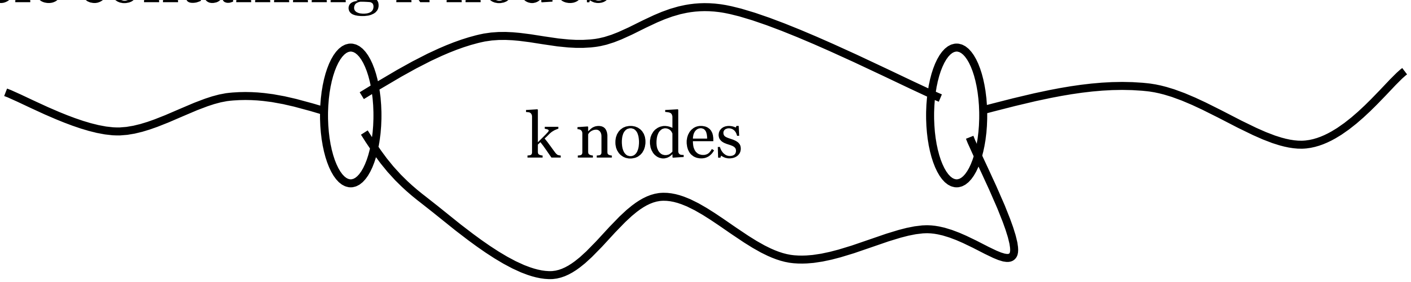
Then $m = m_1 + m_2 + 1 = (n_1 - 1) + (n_2 - 1) + 1 = n - 1$

Proof of 3 $\Rightarrow$ 4

Claim: If G is connected and $m = n - 1$, then G is acyclic and $m = n - 1$

Proof sketch: (by contradiction). Suppose not.

Assume G is connected with $m = n - 1$, and G has a cycle containing k nodes



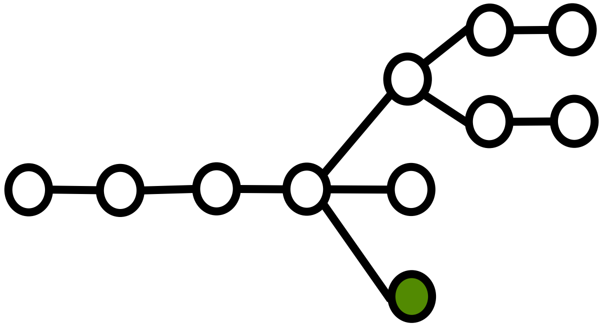
Proof of the Equivalence

Ex: Prove the other statements similarly:
$4 \Rightarrow 5$ and $5 \Rightarrow 1$

To show
$$1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 1$$

Leaves of a Tree

Leaf of a tree is any vertex with degree =1

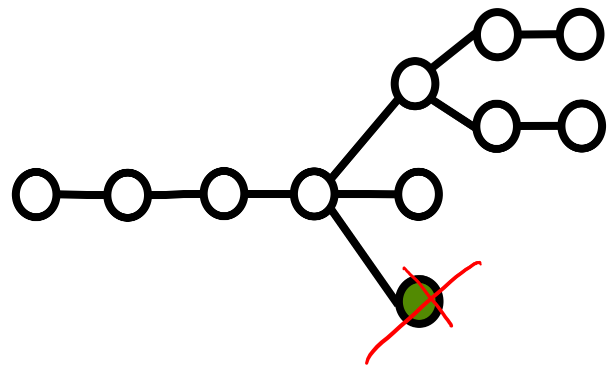


Theorem. There are at least 2 leaves in any tree on $n \geq 2$ nodes

Proof. A tree is connected. Hence every vertex has degree ≥ 1.
Suppose at most one vertex has degree $= 1$.

Leaves of a Tree

Leaf of a tree is any vertex with degree =1

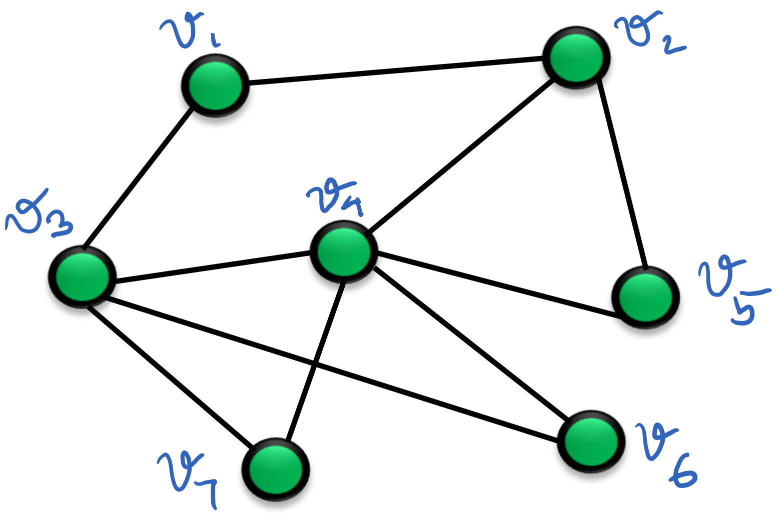


Theorem. There are at least 2 leaves in any tree.

- Very useful in Induction (to reduce tree size to n-1)

Spanning Trees

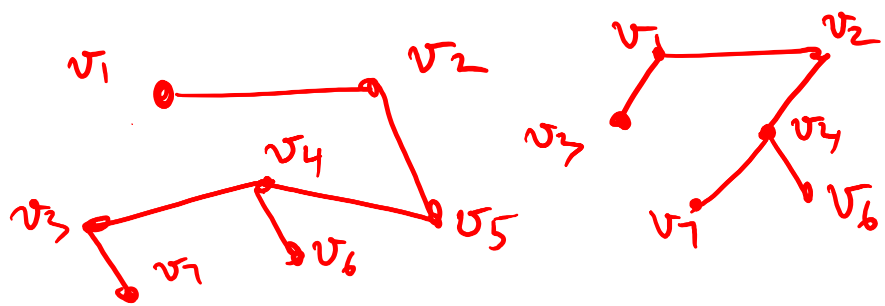
A spanning tree of a graph G is a tree that touches every node of G and uses only edges from G



$G(V,E)$

Every connected graph has a spanning tree

- Minimal subgraph on all vertices of G that is connected.



Fact. Every connected graph has at least $n-1$ edges

Finding Optimal Trees

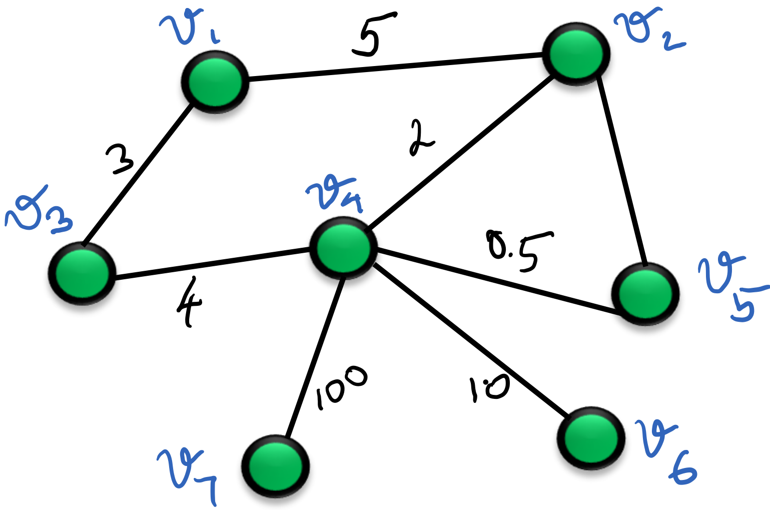
- Trees have many nice properties (connected, uniqueness of paths, no cycles, etc.).
- Great for Communication, Routing etc.

Problem: An ISP wants to set up the cheapest possible network between n people i.e. a tree with smallest communication link costs

Weighted Graphs

Weighted graphs $G(V, E, w)$

Edges have numbers associated with them, representing costs or extent of relations e.g. maps with distances.

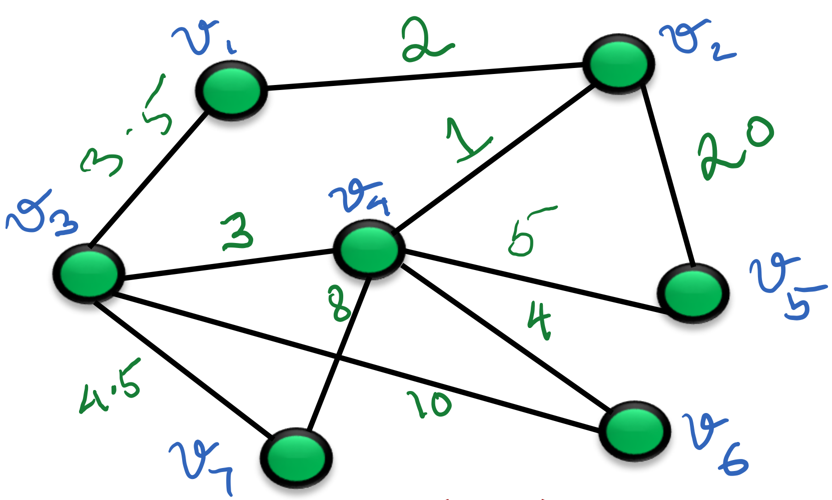


$G(V, E)$

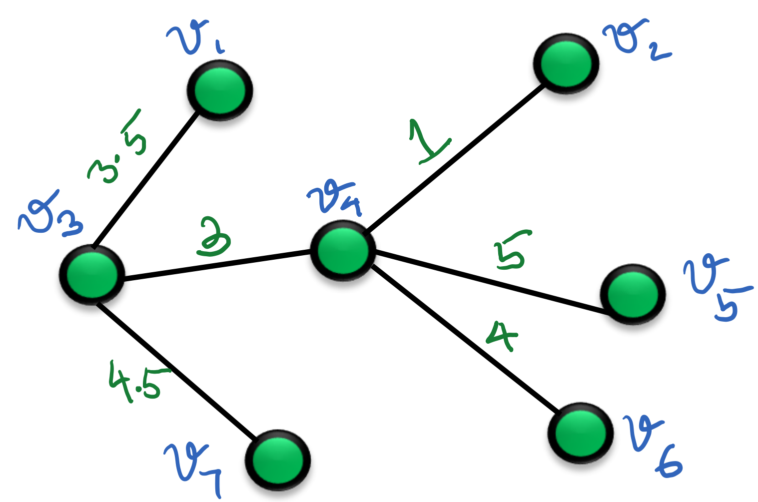
The weights/ costs are all non-negative.

Minimum Spanning Trees (MST)

Problem: Find a minimum spanning tree, that is, a tree on all n vertices of the graph, such that the sum of the edge weights is minimum.



$G(V, E)$



$G(V, E)$

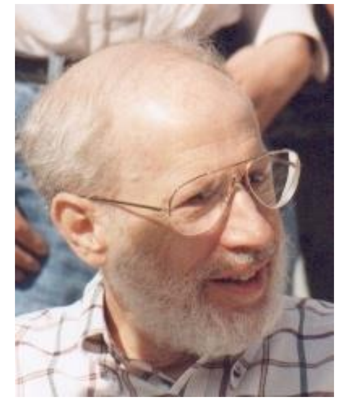
Can we do better?

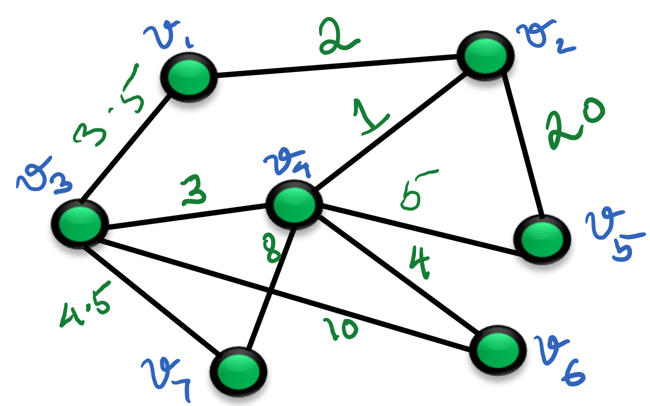
Kruskal's algorithm (1^{st} algorithm)

1. Start with empty graph on vertices of G.

2. Make a sorted list of edges S
(weights are 1, 2, 3, 3.5, 4, 4.5, 5, 8, 10, 20)

3. While S is non-empty:
   a. Pick an edge from S with minimal weight. Remove it from S, and try to include it in subgraph
   b. If it connects two different trees, add the edge. Otherwise discard it.





$G(V, E)$

Thm. Kruskal algorithm outputs a MST

Running the Algorithm

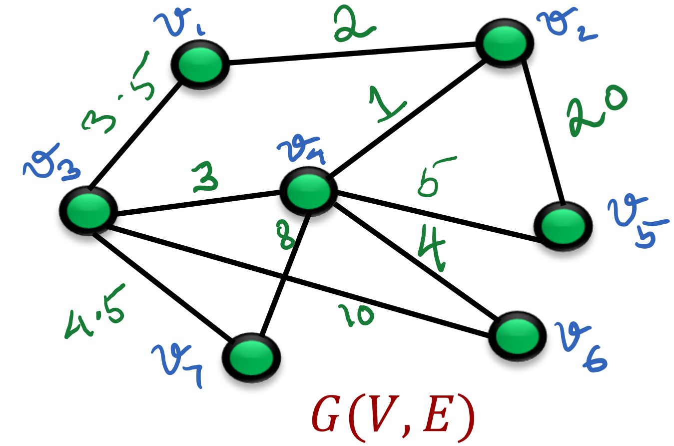
1. Start with empty graph on vertices of G.

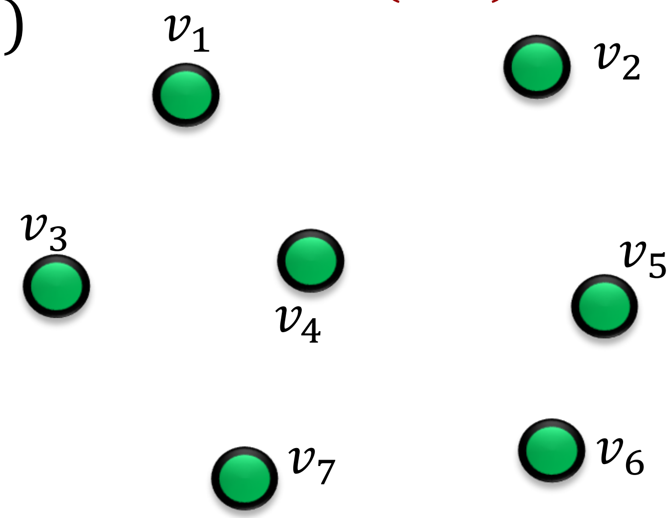
2. Make a sorted list of edges S
(weights are 1, 2, 3, 3.5, 4, 4.5, 5, 8, 10, 20)

3. While S is non-empty:

a) Take the edge with min. weight in S

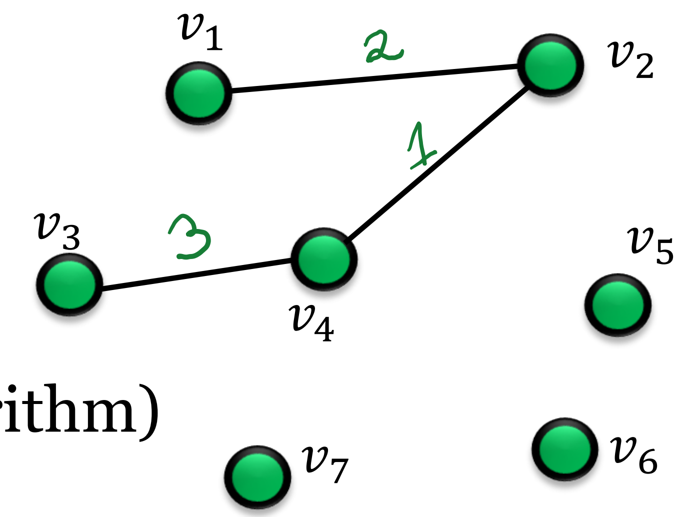
b) If it connects two different trees, add the edge. Otherwise discard it from S.





Proof of Kruskal MST Algorithm

For simplicity, assume all edge weights in graph are distinct



The algorithm outputs a spanning tree T.

Suppose that it's not minimal.

Let M be a minimum spanning tree.

Let e be the first edge chosen by T (algorithm) that is not in M.

If we add e to M, it creates a cycle. Since this cycle isn't fully contained in T, the cycle has an edge $f \in M$ but not in T.

$M' = M + e - f$ is another spanning tree (why?).

Analyzing the Algorithm

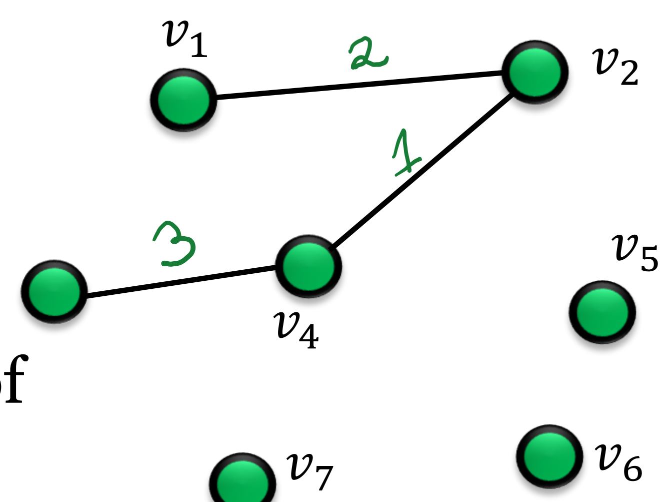
Recall: Algorithm output: T . Minimum spanning tree: M

$$e \in T \setminus M \;\; and \;\; f \in M \setminus T$$

Claim: Suppose $M' = M + e - f$ is another spanning tree, then $cost(e) < cost(f)$, and therefore cost$(M') < cost$(M)

Proof. Suppose not: $cost(e) > cost(f)$.

Then f visited before e by algorithm. But f not added: it would have formed cycle

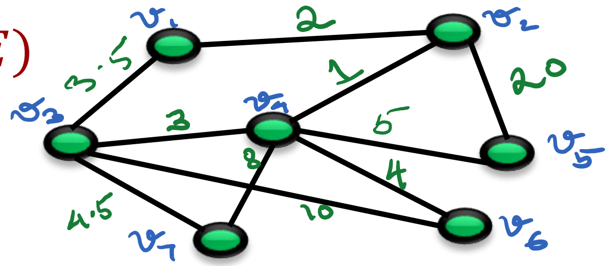


But all of these cycle edges are also edges of M, since e was the first edge not in M.

Hence M has a cycle!

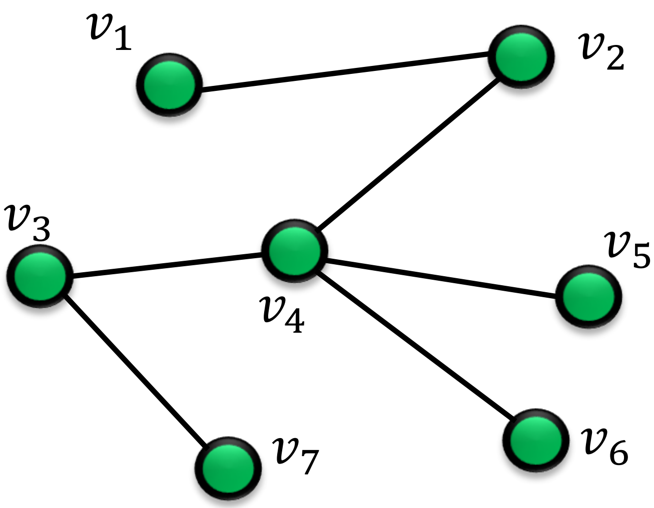
This contradicts the assumption M is a tree!

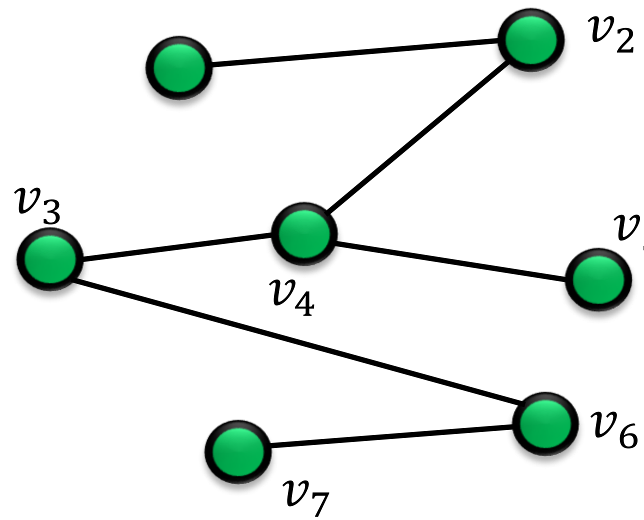
$G(V,E)$



Why it works

We have S = (1, 2, 3, 3.5, 4, 4.5, 5, 8, 10, 20)





Greed is Good (in this case…)

- Kruskal MST algorithm: a greedy algorithm, by adding the least costly edge in each stage succeeds!
- But — in math and life — if pushed too far, the greedy approach can lead to bad results.

Thank you!