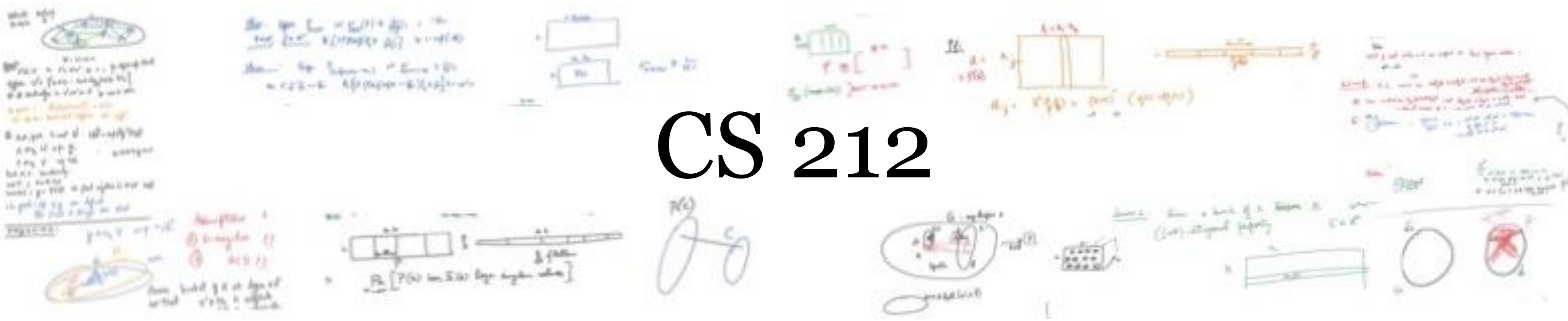


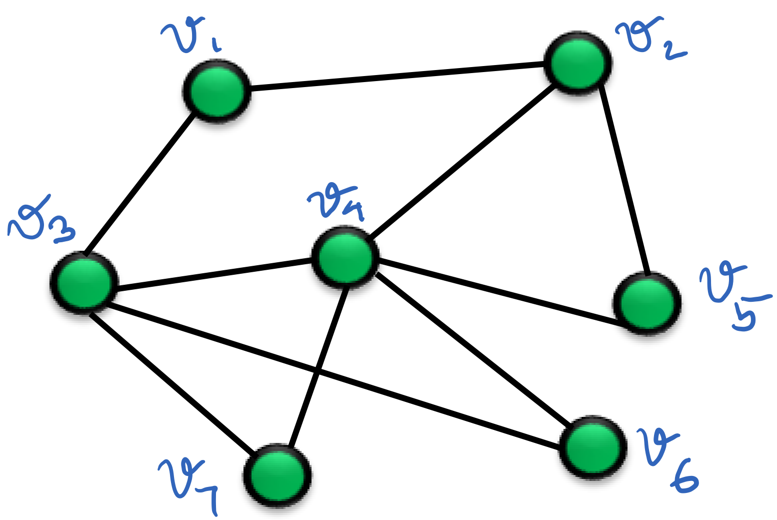
CS 212

Mathematical Foundations of Computer Science

Lecture 20: Spanning Trees

Spanning Trees

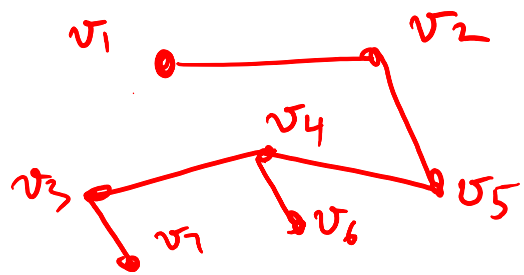
A spanning tree of a graph G is a tree that touches every node of G and uses only edges from G

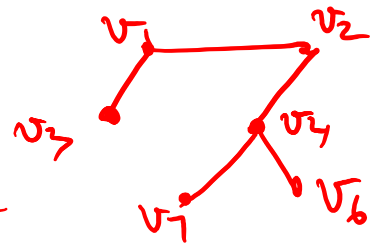


$G(V, E)$

Every connected graph has a spanning tree

- Minimal subgraph of given graph G that is connected.





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

Finding Optimal Trees

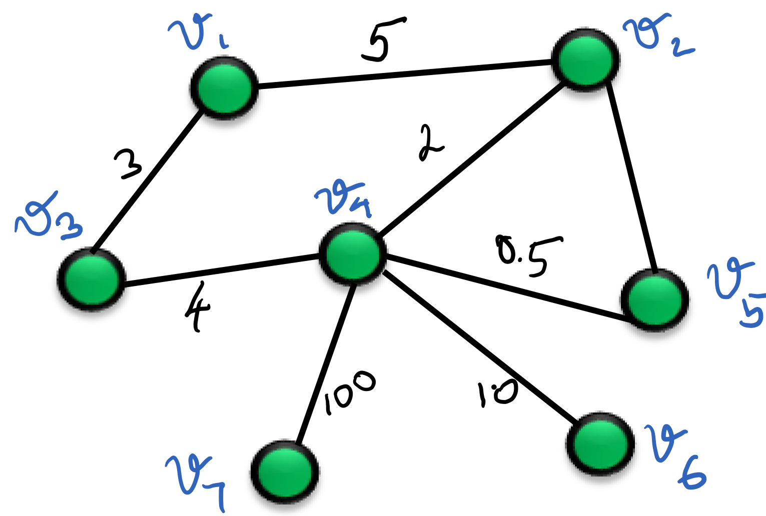
- Trees have many nice properties (connected, uniqueness of paths, no cycles, etc.).
- Great for Communication, Routing etc.

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.

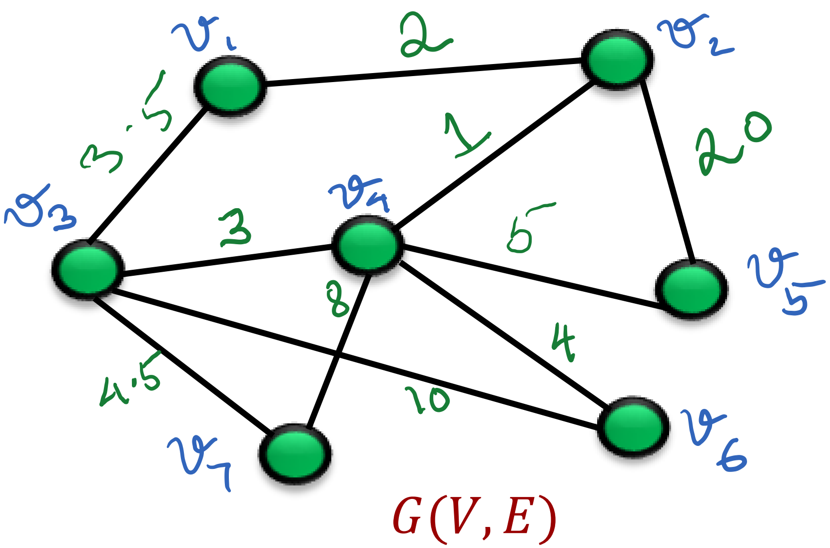


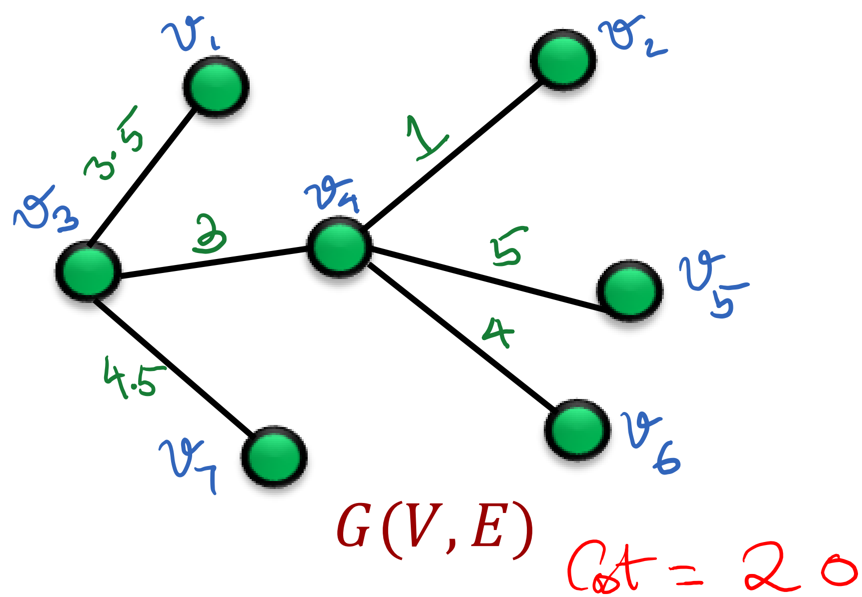
$G(V, E)$

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





Can we do better?

Kruskal's algorithm

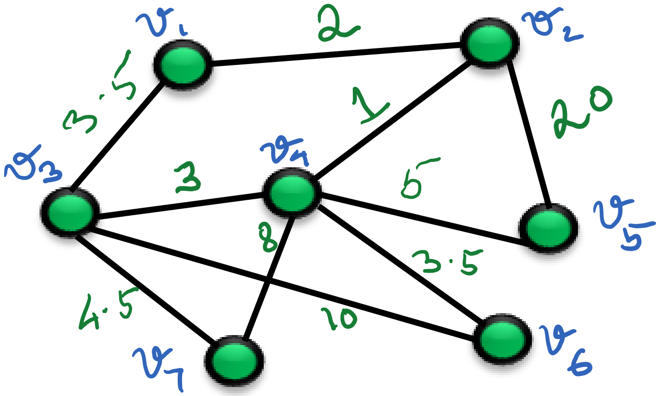
1. Create a forest (a collection of trees) where each node is a separate tree

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 tree/forest.

   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

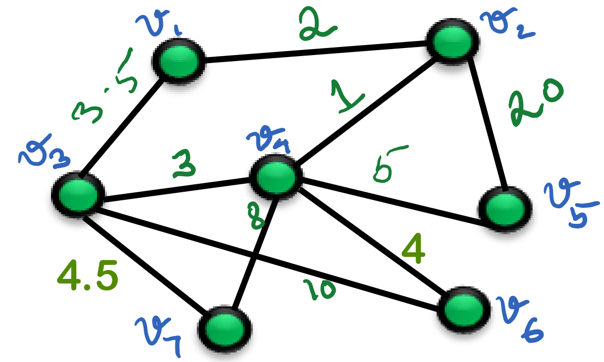
1. Create a forest (a collection of trees) where each node is a separate tree

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

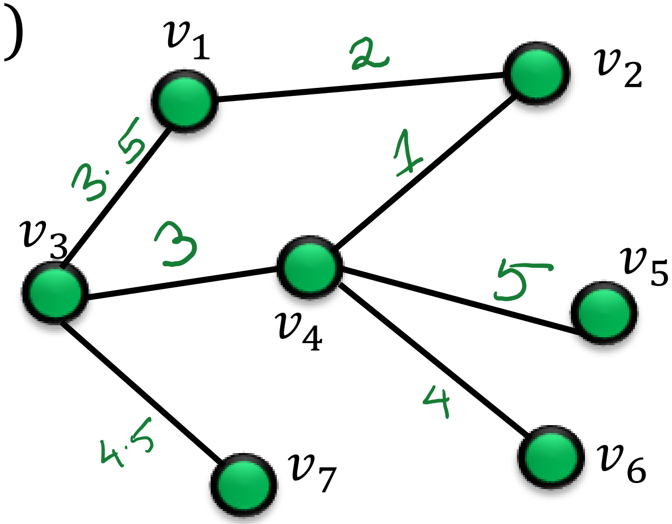
3. While S is non-empty:

 a) Take the edge with min. weight in S

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



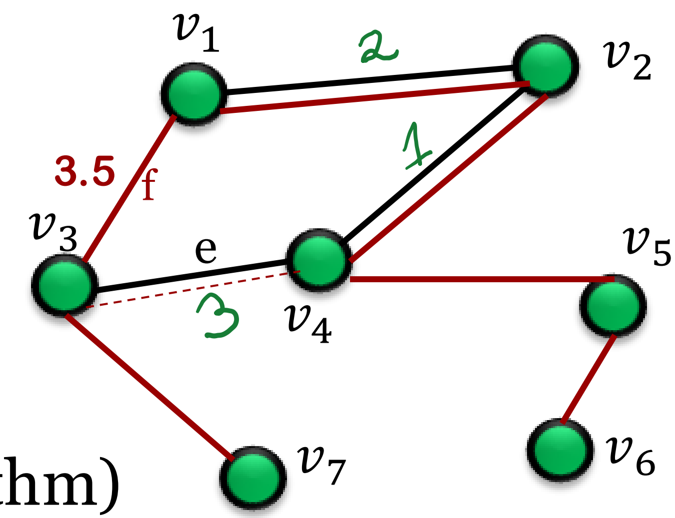
$G(V, E)$



Proof of Kruskal MST Algorithm

Use Contradiction

Let M be a minimum spanning tree.



The algorithm outputs a spanning tree T.
Suppose that it's not minimal.

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* $f \in M \setminus T$

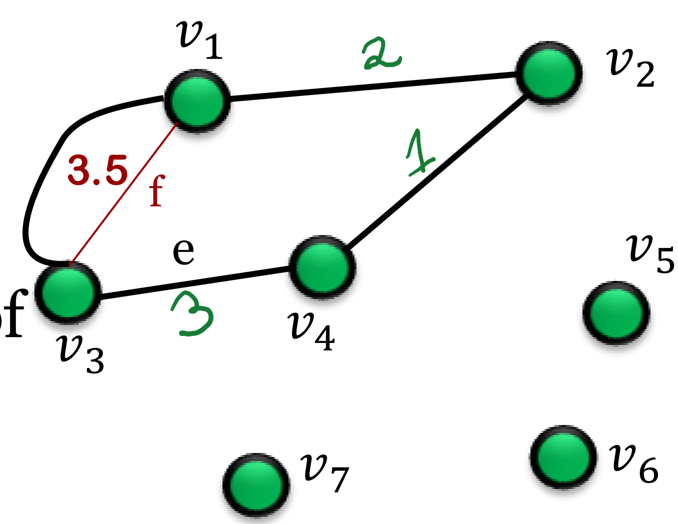
Claim: Suppose $M' = M + e - f$ is another spanning tree, then $cost(e) \leq cost(f)$, and therefore cost(M') $\leq cost$(M)

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 that M is a tree (claim) and that M is minimal (theorem)

Distinct edge weights

Claim: If the edge weights are distinct, there exists a unique minimum spanning tree

Proof: Use contradiction. Assume that there exist two minimum spanning trees, M and N, that are different.

Let e be the smallest edge in N but not in M. Then M+e contains a cycle.

Let f be an edge in the cycle, and therefore in M, but not in N.

Then either M+e-f must have a smaller weight than M, or N+f-e must have a smaller weight than N