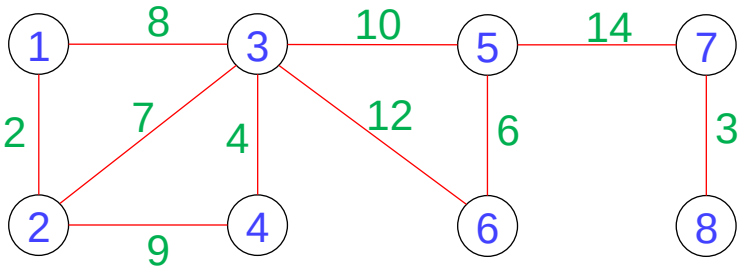
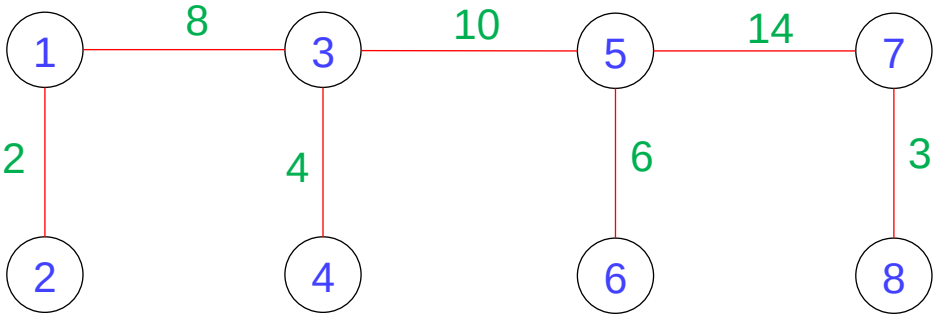
Minimum Spanning Trees

Spanning Tree

- Weighted connected undirected graph ***G = (V, E)***
- ***Spanning tree***

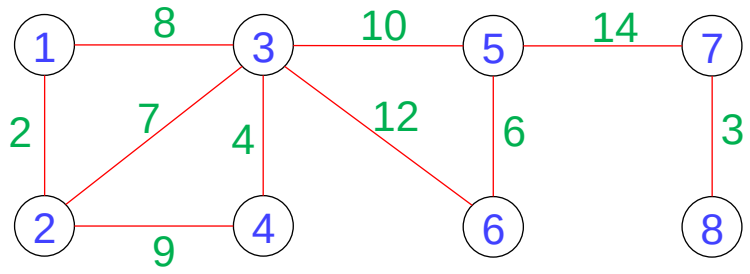


- **Tree** is a connected subgraph without any cycle
- Spanning means should include all the vertices
- So, how many vertices and edges should be there in the ***Spanning tree***?
- So, either we need to select 7 edges or discard 3

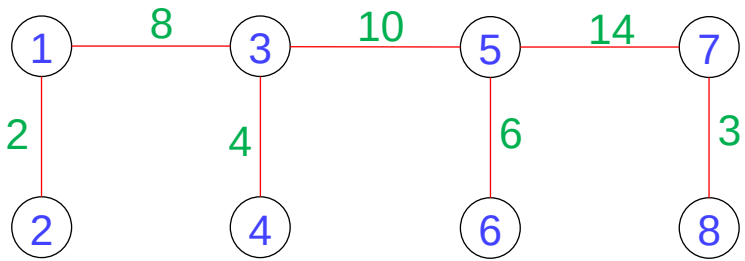


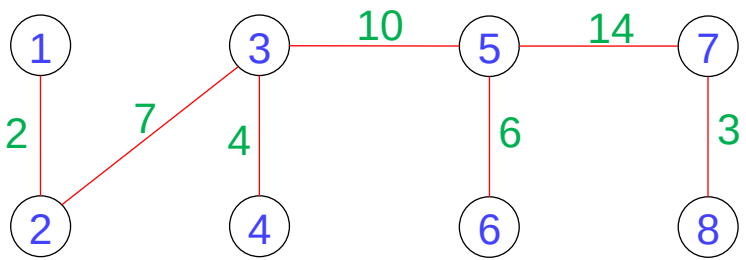
Minimum Spanning Tree

- ***Minimum Spanning** tree*
  - ***Minimum spanning tree*** is a spanning tree of minimum weight/cost/length
  - ***G = (V, E);*** ***c:E → IR⁺***
- What is the cost of the ***spanning tree***?
  - Sum of the costs of the edges in the tree

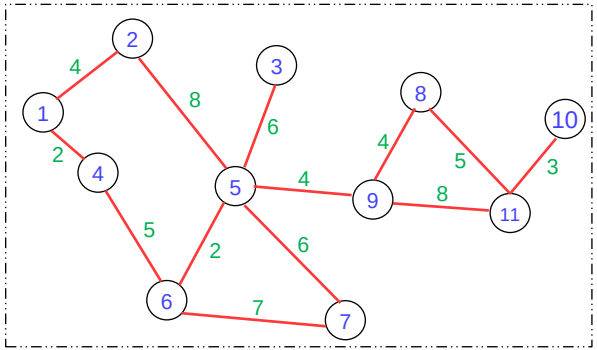


- We are interested to find the ***spanning tree*** with the minimum cost?
- **Note:** Free tree (no notion of root/starting vertex)

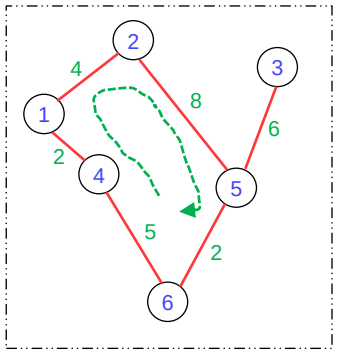




Tree vs. Subgraph vs. Spanning Tree

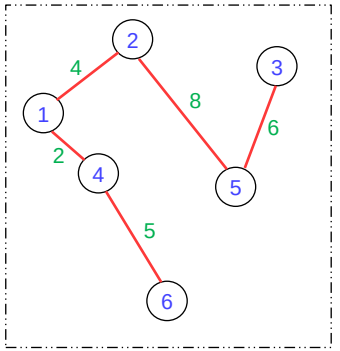


Graph ***G***



A subgraph of ***G***

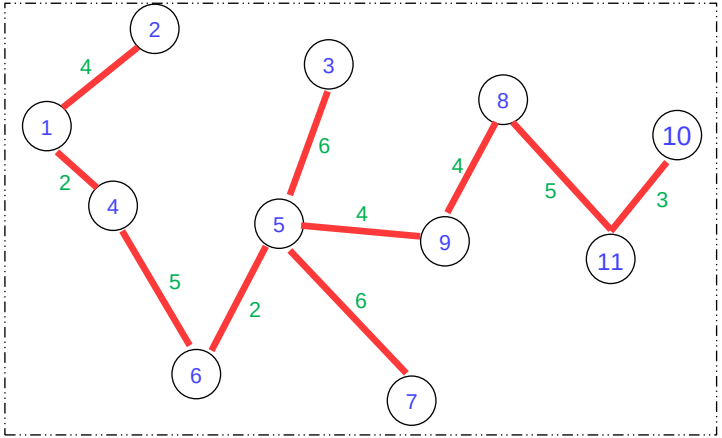
Not a tree



A tree of graph of ***G***

Not a spanning-tree

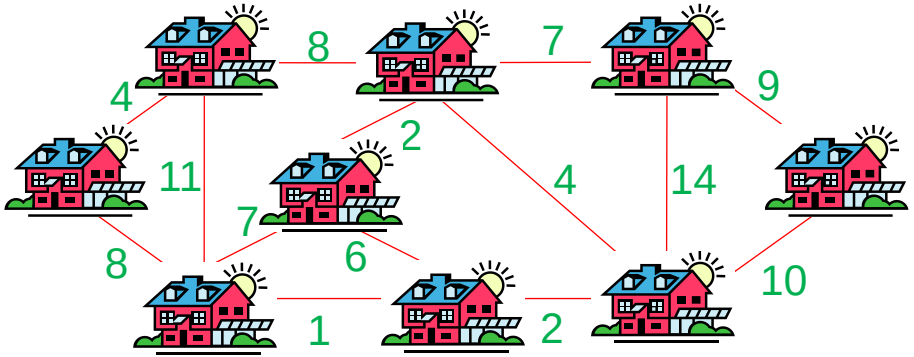
(may not include all the vertices)



A spanning-tree of graph ***G***

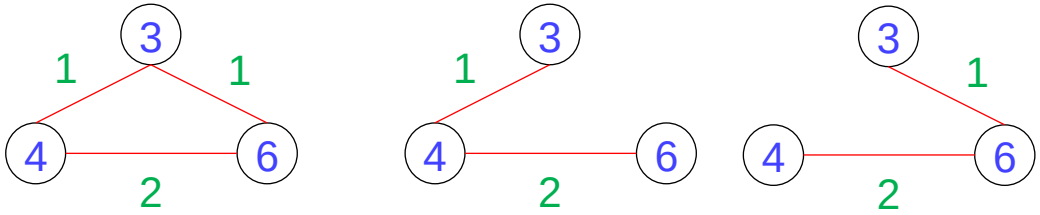
Applications

- Find cheapest way to wire your house
- Find minimum cost to send a message on the Internet
- Find the least expensive way to connect a set of cities, terminals, computers, etc.
- **Problem**
  - A town has a set of houses and a set of roads
  - A road connects 2 and only 2 houses
  - A road connecting houses ***u*** and ***v*** has a repair cost ***c(u, v)***
- **Goal:** Repair enough (and no more) roads such that:
  - Everyone stays connected, i.e., can reach every house from all other houses
  - Total repair cost is minimum



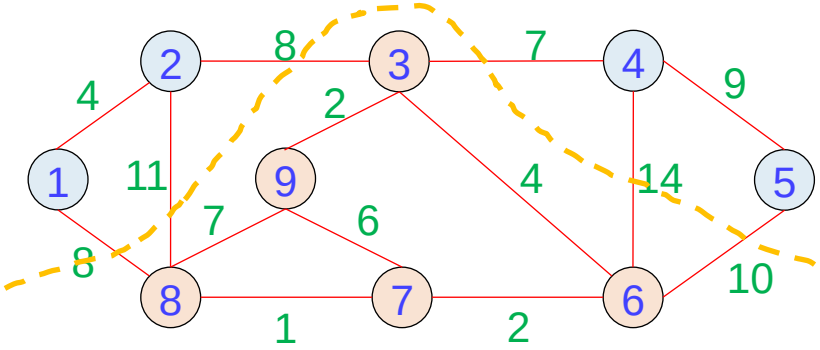
Definitions and Properties

- ***Minimum spanning tree (MST)*** is **not** unique
  - When it is unique?



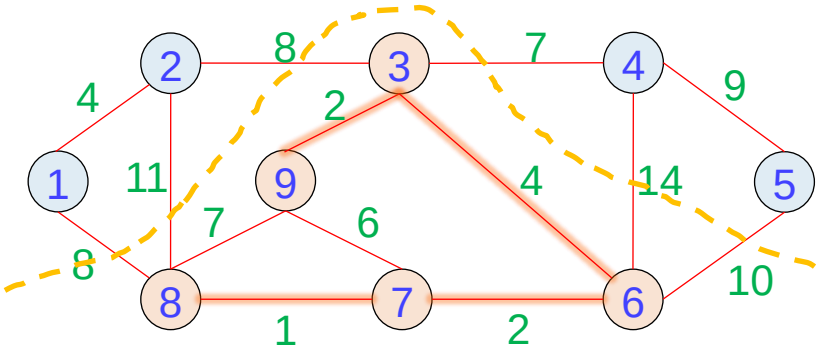
- ***MST*** has no cycles
  - We can take out an edge of a cycle, and still have the vertices connected while reducing the cost

- A ***cut*** (***S, S'***) is a partition of vertices into disjoint sets ***S*** and ***S'*** or ***V – S***
- Any ***cut*** determines a set of edges that have one endpoint in each subset of the partition



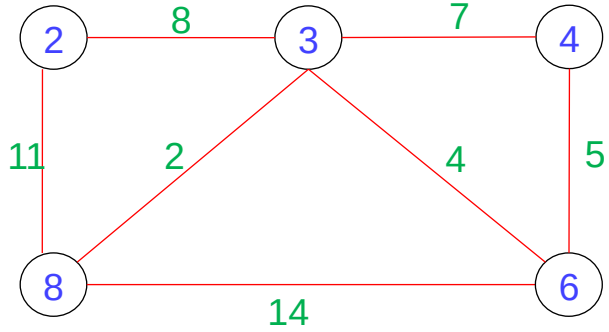
Definitions and Properties

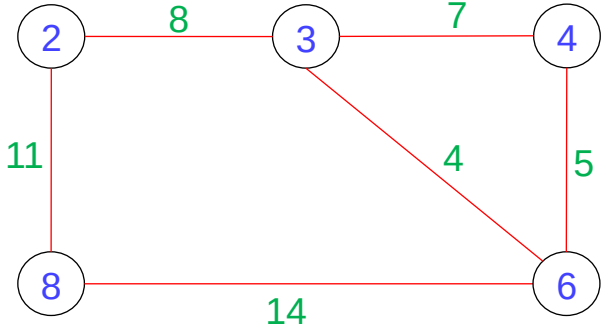
- ***Crosses:*** These edges are said to cross the cut
- ***Respecting:*** A cut respects a set ***A*** of edges if no edge in ***A*** crosses the cut
- ***Light edge:*** An edge is a light edge crossing a cut if its weight is the minimum of any edge crossing the cut
  - **Note:** There may be more than one light edge
- ***Spanning forest:*** If a graph is not connected, then there is a spanning tree for each connected component of the graph
- Inclusion of one edge in the spanning tree result a cycle

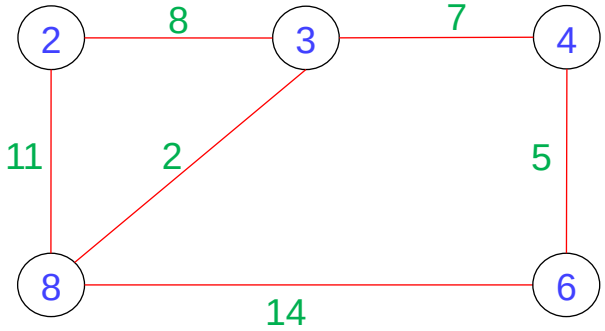


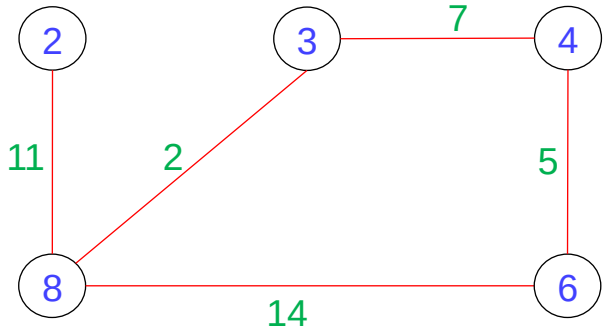
Definitions and Properties

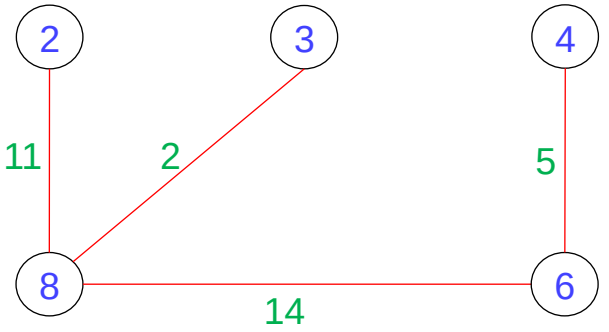
- Due to exclusion of any edge from a cycle, the vertices in the cycle would still be connected, i.e., there will be path between any pairs of vertices











Edge Selection Greedy Strategies

- This could be treated as an algorithm for so computing the ***minimum spanning tree***
- How did it work?
  - Start with some tree then try to include an edge
  - Look at the cycle that gets formed
  - Check if there is an edge of higher cost on that cycle that can be dropped
  - If there is, then you can drop that edge and therefore reduce the cost of the tree
  - Fairly ***expensive*** in terms of running time
- **Different algorithms:**
- Start with an ***N***-vertex ***0***-edge forest
- Consider edges in ascending order of cost
- Select edge if it does not form a cycle together with already selected edges
  - ***Kruskal's*** algorithm
- Start with a ***1***-vertex tree and grow it into an ***N***-vertex tree by repeatedly adding a vertex and an edge
- When there is a choice, add a least cost edge
  - ***Prim's*** algorithm

Edge Rejection Greedy Strategies

- Start with an ***N***-vertex forest
- Each component/tree selects a least cost edge to connect to another component/tree
- Eliminate duplicate selections and possible cycles
- Repeat until only ***1*** component/tree is left
  - ***Sollin's*** algorithm

Edge Rejection Greedy Strategies

- Start with the connected graph
- Repeatedly find a cycle and eliminate the highest cost edge on this cycle
- Stop when no cycles remain
- Consider edges in descending order of cost
- Eliminate an edge provided this leaves behind a connected graph

Next Lecture

Kruskal's Algorithm

Thank you for your attention...

Any question?

Contact:
Department of Information Technology, NITK Surathkal, India
6th Floor, Room: 13
Phone: +91-9477678768
E-mail: shrutilipi@nitk.edu.in, shrutilipi.bhattacharjee@tum.de