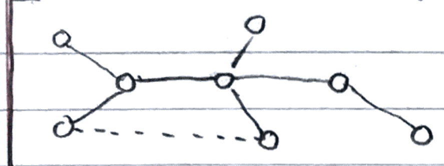
CSC 263 Monday Lecture Mar 19, 2018

- <u>Minimum Spanning Tree</u>

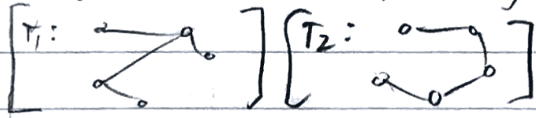
FACT 1] - A tree with m nodes has $m-1$ edges.

FACT 2] Adding any edge to a tree creates a unique cycle C. Removing any edge from cycle C results in a tree again.

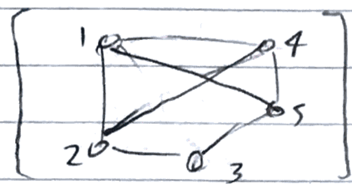
Ex

8 node
7 edges

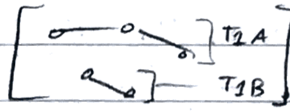
<u>Tree</u>: undirected connected graph w/ no cycles.

- $G=(V,E)$ - undirected connected graph
- $T=(V,E') \mid E' \subseteq E$: spanning tree of G



not unique



- spanning forest: "fragments" of tree :


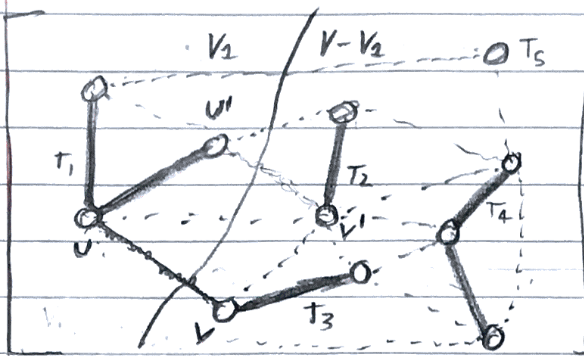
- suppose that each edge $e \in E$ has an associated weight $w(e)$
- weight of spanning tree $W(T) = \sum_{e \in E'} w(e)$
- MINIMUM SPANNING TREE : spanning tree w/ lowest weight
→ spanning trees are <u>not</u> unique for a given graph G

- Application: broadcast message to network most cheaply
- Application: connect cities via electrical grid most cheaply

- A complete graph G with n nodes has n^{n-2} spanning trees. Then brute force (generate all spanning trees & find cheapest) impossible!

- <u>MST Problem</u>: Given a weighted G, find an MST of G.

Example:



- Given graph G with existing minimum spanning forest; one tree seperated from rest into V_1.
- Puzzle: find one more link in MST.

Note: full theorem proof in coursenotes

Theorem

- Suppose some MST T of G contains the spanning forest $\{T_1, \ldots T_k\}$.
- Let (v,u) be an edge with minimum weight between some tree in the forest T_i and the other trees (some other tree in forest).
- Then some MST T^* of G that contains $\{T_1, \ldots T_k\} + (u,v)$.

Proof (short version)

- Case 1: the edge (u,v) belongs to T. Then $T = T^*$ correct edge selected.
- case 2: T does not contain (u,v). Then $T' = T + (u,v)$ has cycle C. Cycle C has <u>two</u> edges crossing from V_1 to $V - V_1$: (u,v) and some (u', v'). By definition, (u,v) has min weight, $\therefore$ by fact 2 we can remove (u', v') and have an MST. $\therefore$ T^* is MST. Note that $T^* = T + (u,v) - (u', v')$; $W(T^*) = W(T) + W((u,v)) - W((u', v'))$.

- Wednesday: we will explore 2 MST building algos.