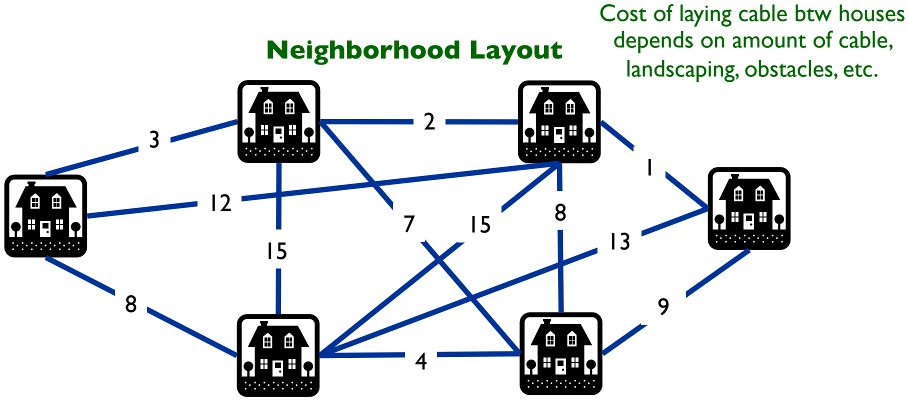
Objectives

- Minimum Spanning Tree

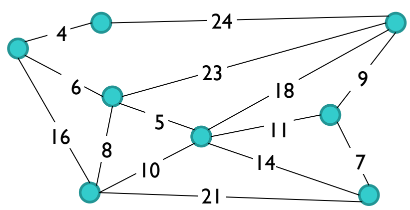
Laying Cable

- Comcast wants to lay cable in a neighborhood
  - Reach all houses
  - Least cost

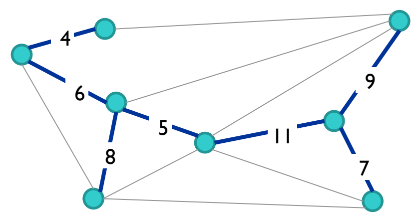


Minimum Spanning Tree (MST)

- **Spanning tree**: spans all nodes in graph
- Given a connected graph G = (V, E) with positive edge weights c_e, an ***MST*** is a subset of the edges $T \subseteq E$ such that T is a ***spanning tree*** whose **sum** of **edge weights** is ***minimized***



G = (V, E)

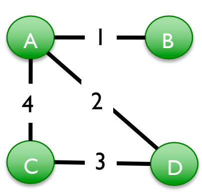


T, $\Sigma_{e \in T}\, c_e = 50$

Examples

Identify **spanning trees** and which is the ***minimal*** spanning tree.

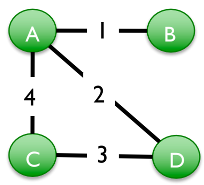
Graph



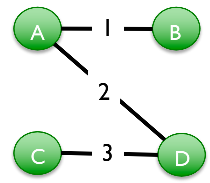
Examples

Identify **spanning trees** and which is the ***minimal*** spanning tree.

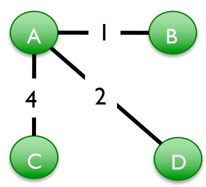
Graph

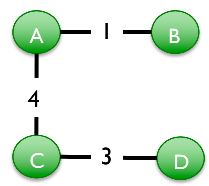


MST:



Other Spanning Trees:





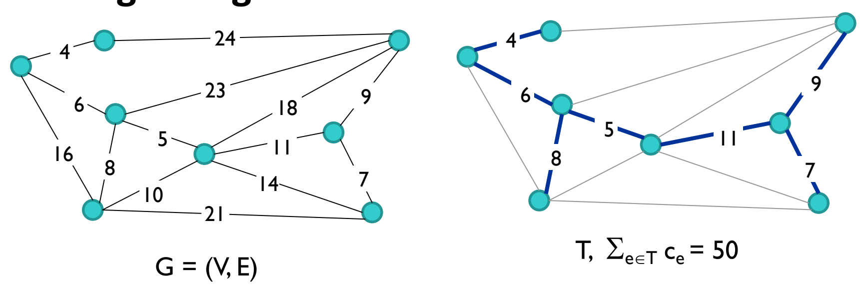
MST Applications

- Network design
  - telephone, electrical, hydraulic, TV cable, computer, road
- Approximation algorithms for NP-hard problems
  - traveling salesperson problem, Steiner tree
- Indirect applications
  - max bottleneck paths
  - image registration with Renyi entropy
  - learning salient features for real-time face verification
  - reducing data storage in sequencing amino acids in a protein
  - model locality of particle interactions in turbulent fluid flows
- **Cluster analysis**

http://www.ics.uci.edu/~eppstein/gina/mst.html

Minimum Spanning Tree

- Given a connected graph G = (V, E) with positive edge weights c_e, an ***MST*** is a subset of the edges $T \subseteq E$ such that T is a ***spanning tree*** whose **sum** of **edge weights** is ***minimized***



Why *must* the solution be a tree?

Minimum Spanning Tree

- Assume have a minimal solution that is not a tree, i.e., it has a cycle
- What could we do?
  - What do we know about the edges?
  - How does that change the cost of the solution?

Minimum Spanning Tree

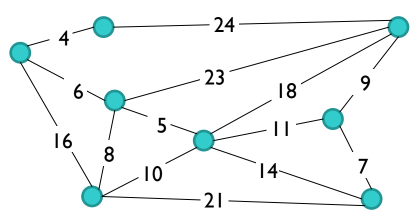
- Proof by Contradiction.
- Assume have a minimal solution V that is not a tree, i.e., it has a cycle
- Contains edges to all nodes because solution must be connected (spanning)
- Remove an edge from the cycle
  - Can still reach all nodes (could go "long way around")
  - **But** at lower total cost
  - Contradiction to our minimal solution

Ideas for Solutions?

- Cayley's Theorem. There are n^{n-2} spanning trees

  can't solve by brute force

- Towards a solution…
  - Where to start?



G = (V, E)

Greedy Algorithms

All three algorithms produce a MST

- Prim's algorithm.
  - Start with some root node *s* and greedily grow a tree *T* from *s* outward
  - At each step, add cheapest edge *e* to *T* that has exactly one endpoint in *T*
  - Similar to Dijkstra's (but simpler)
- Kruskal's algorithm.
  - Start with $T = \phi$
  - Consider edges in ascending order of cost
  - Insert edge *e* in *T* unless doing so would create a cycle
- Reverse-Delete algorithm.
  - Start with T = E
  - Consider edges in descending order of cost
  - Delete edge *e* from *T* unless doing so would disconnect *T*

What do these algorithms have/do/check in common?

What Do These Algorithms Have in Common?

- When is it safe to include an edge in the minimum spanning tree?

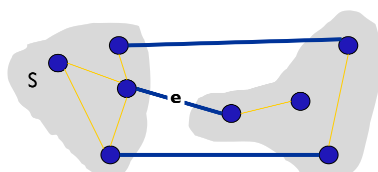
Cut Property

- When is it safe to eliminate an edge from the minimum spanning tree?

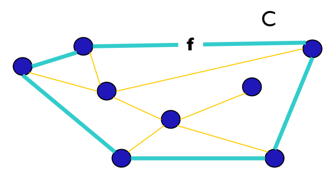
Cycle Property

Cut and Cycle Properties

- Simplifying assumption: All edge costs c_e are distinct
  - → MST is unique
- Cut property. Let *S* be any subset of nodes, and let *e* be the min cost edge with exactly one endpoint in *S*. Then MST contains *e*.
- Cycle property. Let *C* be any cycle, and let *f* be the max cost edge belonging to *C*. Then MST does *not* contain *f*.



Cut Property: e is in MST

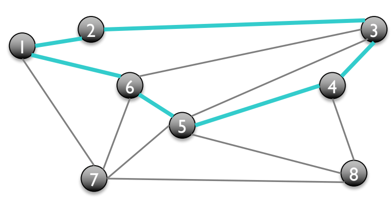


Cycle Property: f is **not** in MST

Let's try to prove these …

Cycles and Cuts

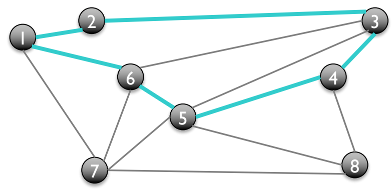
- Cycle. Set of edges in the form
  a-b, b-c, c-d, …, y-z, z-a



Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1

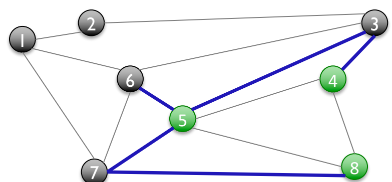
Cycles and Cuts

- Cycle. Set of edges in the form a-b, b-c, c-d, …, y-z, z-a



Cycle C = 1-2, 2-3, 3-4,
4-5, 5-6, 6-1

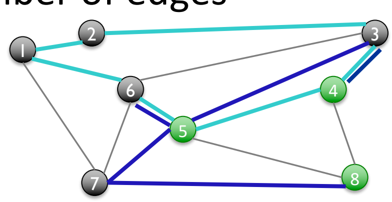
- Cutset. A *cut* is a subset of nodes *S*.
The corresponding *cutset D* is the subset of edges with *exactly one* endpoint in *S*.



Cut S = { 4, 5, 8 }
Cutset D = 5-6, 5-7, 3-4,
3-5, 7-8

Cycle-Cut Intersection

- Claim. A *cycle* and a *cutset* intersect in an even number of edges

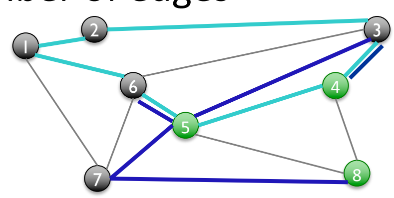


Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1
Cut S = { 4, 5, 8 }
Cutset D = 3-4, 3-5, 5-6, 5-7, 7-8
Intersection = 3-4, 5-6

What are the possibilities for the cycle?

Cycle-Cut Intersection

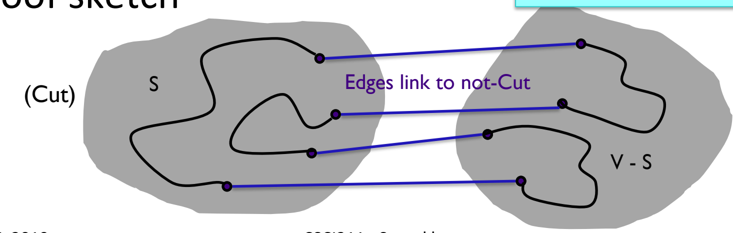
- Claim. A *cycle* and a *cutset* intersect in an even number of edges



Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1
Cut S = { 4, 5, 8 }
Cutset D = 3-4, 3-5, 5-6, 5-7, 7-8
Intersection = 3-4, 5-6

1. Cycle all in S
2. Cycle not in S
3. Cycle has to go from S→V-S *and* back

- Proof sketch



Proving Cut Property: OK to Include Edge

- Simplifying assumption: All edge costs c_e are distinct.
- **Cut property.** Let *S* be any subset of nodes, and let *e* be the min cost edge with exactly one endpoint in *S*.
  Then the MST T* contains *e*.
- Pf.?

Proving Cut Property: OK to Include Edge

- Simplifying assumption: All edge costs c_e are distinct.
- **Cut property.** Let *S* be any subset of nodes, and let *e* be the min cost edge with exactly one endpoint in *S*.
  Then the MST T* contains *e*.
- Pf. (exchange argument)
  - Suppose there is an MST T* that does not contain *e*
    - What do we know about T, by defn?
    - What do we know about the nodes *e* connects?

Looking Ahead

- Problem Set 5 due Friday