CSCI 1900
Discrete Structures

Minimal Spanning Trees
Reading: Kolman, Section 7.5

In-Class Exercise

A small startup airline wants to provide service to the 5 cities in the table to the right. Allowing for multiple connecting flights, determine all of the direct flights that would be needed in order to service all five cities.

City 1	City 2	Mileage
Cleveland	Philadelphia	400
Cleveland	Detroit	200
Cleveland	Chicago	350
Cleveland	Pittsburg	150
Philadelphia	Detroit	600
Philadelphia	Chicago	700
Philadelphia	Pittsburg	300
Detroit	Chicago	300
Detroit	Pittsburg	300
Chicago	Pittsburg	450

Source: http://www.usembassymalaysia.org.my/distance.html

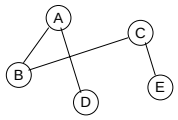
Undirected Graph

- If you recall from our discussion on types of relations, a symmetric relation is one that for every relation (a,b) that is contained in R, the relation (b,a) is also in R.
- If the relation is represented with a matrix, this means that the matrix is symmetric across the main diagonal.
- In the digraph of a symmetric relation, every edge is bidirectional, i.e., there is no defined direction.

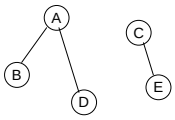
Connected Relation

- A relation is ***connected*** if for every a and b in R, there is a path from a to b.
- It is easier to see a connected relation using a digraph than it is to describe in using words.

Connected



Not Connected



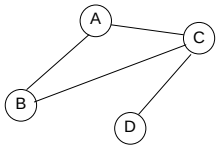
Spanning Tree

- Textbook definition: "If R is a symmetric, connected relation on a set A, we say that a tree T on A is a spanning tree for R if T is a tree with exactly the same vertices as R and which can be obtained from R by deleting some edges of R." p. 275
- Basically, a undirected spanning tree is one that connects all n elements of A with n-1 edges.
- To make a cycle connecting n elements, more than n-1 edges will be needed. Therefore, there are no cycles.

Weighted Graph

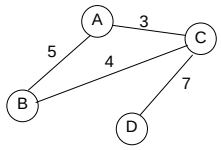
In the past, we have represented a undirected graph with unlabeled edges. It can also be represented with a symmetric binary matrix.

$$M_T = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



Weighted Graph (continued)

By giving the edges a numeric value indicating some parameter in the relation between two vertices, we have created a weighted tree.



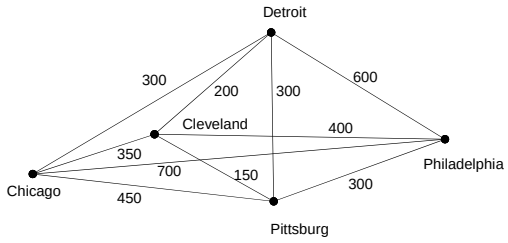
Weighted Graph (continued)

We can still use matrix notation to represent a weighted graph. Replace the 1's used to represent an edge with the edge's weight. A 0 indicates no edge.

$$M_T = \begin{bmatrix} 0 & 5 & 3 & 0 \\ 5 & 0 & 4 & 0 \\ 3 & 4 & 0 & 7 \\ 0 & 0 & 7 & 0 \end{bmatrix}$$

Back to In-Class Exercise

(Not drawn to scale)



Minimal Spanning Tree

- Assume T represents a spanning tree for an undirected graph.
- The total weight of the spanning tree T is the sum of all of the weights of all of the edges of T.
- The one(s) with the minimum total weight are called the ***minimal spanning tree(s)***.
- As suggested by the "(s)" in the above definition, there may be a number of minimal spanning trees for a particular undirected graph with the same total weight.

Algorithms for Determining the Minimal Spanning Tree

There are two algorithms presented in our textbook for determining the minimal spanning tree of an undirected graph that is connected and weighted.

- Prim's Algorithm: process of stepping from vertex to vertex
- Kruskal's Algoritm: searching through edges for minimum weights

Prim's Algorithm

From textbook, p. 281

Let R be a symmetric, connected relation with n vertices.

1. Choose a vertex v_1 of R. Let $V = \{v_1\}$ and $E = \{\}$.
2. Choose a nearest neighbor v_i of V that is adjacent to v_j, $v_j \in V$, and for which the edge (v_i, v_j) does not form a cycle with members of E. Add v_i to V and add (v_i, v_j) to E.
3. Repeat Step 2 until $|E| = n - 1$. Then V contains all n vertices of R, and E contains the edges of a minimal spanning tree for R.

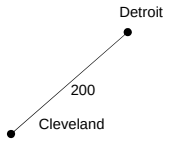
Prim's Algorithm in English

- The goal is to one at a time include a new vertex by adding a new edge without creating a cycle
- Pick any vertex to start. From it, pick the edge with the lowest weight.
- As you add vertices, you will add possible edges to follow to new vertices.
- Pick the edge with the lowest weight to go to a new vertex without creating a cycle.

In-Class Exercise (1st edge)

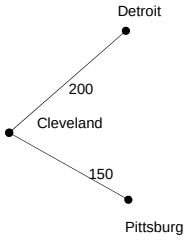
Pick any starting point: Detroit.

Pick edge with lowest weight: 200 (Cleveland)



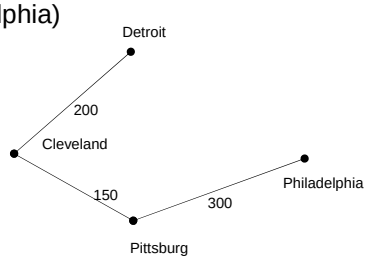
In-Class Exercise (2nd edge)

Pick any edge connected to Cleveland or Detroit that doesn't create a cycle: 150 (Pittsburg)



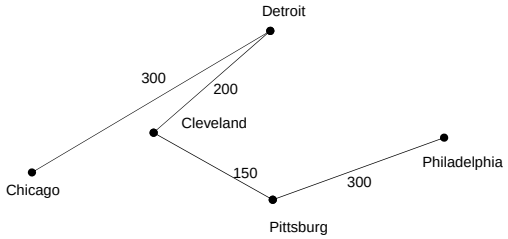
In-Class Exercise (3rd edge)

Pick any edge connected to Cleveland, Detroit, or Pittsburg that doesn't create a cycle: 300 (Philadelphia)



In-Class Exercise (4th edge)

Pick any edge connected to Cleveland, Detroit, Pittsburg, or Philadelphia that doesn't create a cycle: 300 (Chicago) This gives us 4 edges!



Kruskal's Algorithm

From textbook, p. 284

Let R be a symmetric, connected relation with n vertices and let $S = \{e_1, e_2, e_3, \ldots e_k\}$ be the set of all weighted edges of R.

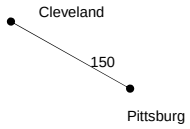
1. Choose an edge e_1 in S of least weight. Let $E = \{e_1\}$. Replace S with $S - \{e_1\}$.
2. Select an edge e_i in S of least weight that will not make a cycle with members of E. Replace E with $E \cup \{e_i\}$ and S with $S - \{e_i\}$.
3. Repeat Step 2 until $|E| = n - 1$.

Kruskal's Algorithm in English

- The goal is to one at a time include a new edge without creating a cycle.
- Start by picking the edge with the lowest weight.
- Continue to pick new edges without creating a cycle. Edges do not necessarily have to be connected.
- Stop when you have n-1 edges.

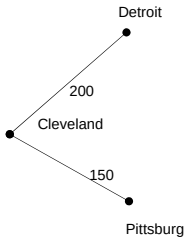
In-Class Exercise (1st edge)

First edge picked is lowest value of 150 (Cleveland to Pittsburg)



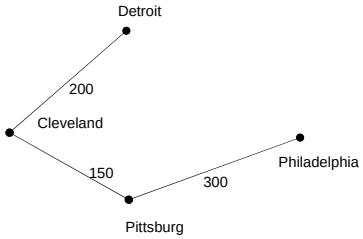
In-Class Exercise (2nd edge)

Next lowest edge is 200 (Cleveland to Detroit)



In-Class Exercise (3rd edge)

There are a few edges of 300, but Detroit to Pittsburg cannot be selected because it would create a cycle. We go with Pittsburg to Philadelphia.



In-Class Exercise (4th edge)

The next lowest edge that doesn't create a cycle is 300 edge from Detroit to Chicago. This is the 4th edge!