Lesson 10, Combinatorial Models: Minimum Spanning Tree

1 Combinatorial Models

Many optimization problems are naturally modeled by a combinatorial structure, such as a **graph**. For the next few weeks, we will be talking about several famous combinatorial optimziation problems: *minimum spanning tree*, *traveling salesperson*, and *vehicle routing problem.*

Combinatorial optimization problems are usually more general than the network problems we've seen before as they operate on a **graph** instead of a **network**. Recall that a network is a special type of graph.

For now, we will develop integer programs to model these problems. Integer programming is appropriate for traveling salesperson and vehicle routing problems (they're NP-Complete which means **really** hard to solve); but, like the other networks we have done so far, there are more efficient ways to handle MST.

2 Graph Terminology

Suppose $G = (V, E)$ is an *undirected* graph. (So far we have worked with directed graphs.) Recall that:

- V is the set of vertices or nodes
- E is the set of edges

An undirected graph is different in that the edges can now be traversed in both directions.

As a result, it's good to be consistent in naming the edges. For example, using our old style of naming, an edge connecting nodes 1 and 2 could be both $x_{1,2}$ and $x_{2,1}$. As a result, we will always name edges from lower index to higher index.

Graph G is **connected** if for *every* pair of vertices $a,\ b \in V$, there exists a **path** of edges in G connecting a and b.

Example:

A **cycle** is a closed path of nodes meaning that the first and last node in the path is the same vertex.

Example:

If G is a connected graph that contains a **cycle**, then the removal of a single edge from the cycle does not destroy the connectivity of G.

1. Illustrate a cycle and prove the fact that if G has a cycle, removing any edge from the cycle does not make the graph disconnected.

A connected graph that contains no cycles is called a **tree**.

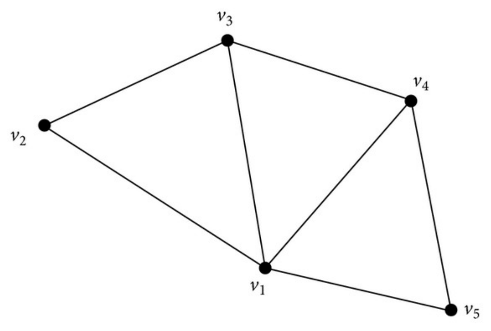
Trees are **minimally connected**: if we remove any edge from a tree, the resulting graph is disconnected.

2. Convince yourself of the previous fact by drawing a tree on 6 vertices. Are there any edges you can remove from the graph without losing connectivity?

Side note: Trees are really important in tons of real world problems. For example, every time you make a Google search or use GoogleMaps an algorithm is called where one of the key parts is analyzing a huge tree.

A **spanning tree** of a graph G is a subset of the edges of G that form a *tree that connectes every vertex in* V (i.e., it spans the nodes of G).

3. Consider the graph G on 5 vertices below.



(a) Draw two different spanning trees of G.

(b) How many edges does each spanning tree have? $\boxed{}$

(c) List the elements (edges) of the spanning tree.

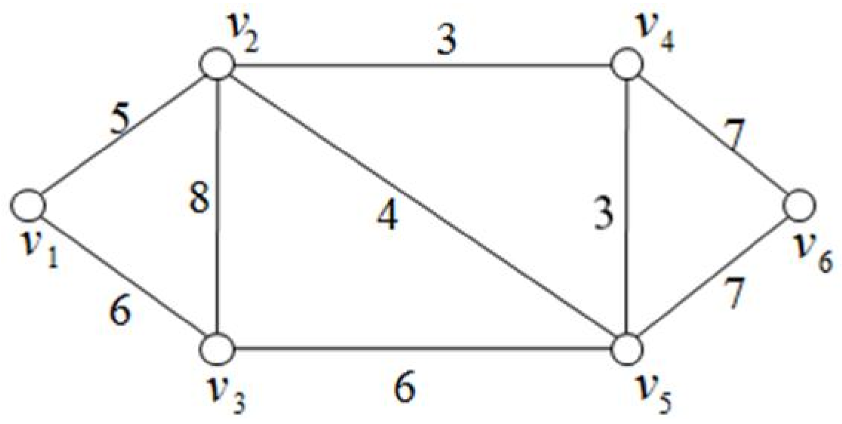
(d) How many edges does a spanning tree of a graph G with vertex set V have in general?

3 Minimum Spanning Tree (MST) Problem

Associate a cost c_{ij} with every edge $(i, j) \in E$. The problem of finding the *spanning tree* of graph G with *minimum edge cost* is known as the **minimum spanning tree problem**.

- Real world applications of the minimum spanning tree problem include planning road, electrical, data, phone, and water networks.

4. Solve the minimum cost spanning tree problem for the graph shown below. (The numbers on the edges are the edge costs.)



Minimum spanning tree $T =$

4 MST IP Formulation

It is more efficient to solve a min-cost spanning tree problem via an algorithm (like maybe from SA403), it is still instructive to model it using IP. Specifically because fully understanding the MST problem formulation will be helpful in understanding traveling salesperson and vehicle routing.

A local phone company is interested in laying cable from the main road (where the Main switch is located) to a new housing subdivision, and wants to do so in the least expensive way. It has the option of laying cable from the road to any house, or it can lay cable between the houses. Each house must be connected through some path to the road. The following matrix gives the total cost of laying cable between any two locations, where the first location is the main road.

$$C = \begin{bmatrix} 0 & 25 & 25 & 15 & 10 & 30 \\ 25 & 0 & 10 & 25 & 20 & 15 \\ 25 & 10 & 0 & 20 & 30 & 15 \\ 15 & 25 & 20 & 0 & 15 & 20 \\ 10 & 20 & 30 & 15 & 0 & 20 \\ 30 & 15 & 15 & 20 & 20 & 0 \end{bmatrix}$$

How should the phone company connect the houses to the road in order to minimize its total cost?

5. Draw a graph to represent this problem below.

6. Define the **sets**, **variables**, and **parameters** for the problem.

7. Write the objective function in both concrete and parameterized form.

8. Minimum spanning trees must touch every vertex. Write a concrete constraint that ensures that vertex 1 is covered by the spanning tree returned by the solver. Do the same for vertices 2 and 3. Finally, write an parameterized class of constraints that ensures that every vertex is covered by an edge. (This should remind you of set covering hopefully!)

9. Spanning trees contain $\boxed{}$ edges. Write a concrete constraint that ensures that the solution returned by the solver has exactly $\boxed{}$ edges. Convert this constraint to parameterized form.

10. At this point, we have constraints that enforce:

    - Every node in the graph is connected to one edge
    - The graph has exactly $n - 1$ edges

    Is there a solution that satisfies these two constraints while not producing a spanning tree?

11. Write a concrete constraint that prevents the graph that you sketched above. Then write a set of parameterized constraints that prevents ANY graphs of this kind from being returned by the solver. (There is one constraint for every subset of vertices of G.)

These constraints above are called **cycle-elimination** or often **subtour-elimination** constraints. In real-life applications, there are usually way too many possible “cycle-elimination” constraints to include them all in a model. In practice, these constraints may be added iteratively to eliminate cycles in a solution returned by the solver. We will do something like this in the context of vehicle routing.

12. Combining all of the steps above, write the full parameterized model for MST.