Lesson 10. Combinatorial Models: Minimum Spanning Tree

1   Today...

- Combinatorial Models

- Graph Terminology

- Minimum Spanning Tree (MST) Problem

- MST IP Formulation

2   Combinatorial Models

Many optimization problems are naturally modeled by a combinatorial structure, such as a **graph**. In this section, we will explore two famous **combinatorial optimization** problems: *minimum spanning tree* and *traveling salesperson*. *Shortest path* is another famous combinatorial optimization problem that we have already encountered. SA403 Graph and Network Algorithms is all about combinatorial problems.

Problem 1. 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? Brainstorm a little bit on how you might go about modeling this problem. We will come back to this later on in the lecture.

3 Graph Terminology

Suppose G is an *undirected* graph. (So far we have worked with directed graphs.) Let $\mathcal{V}$ denote the set of vertices/nodes in G. Let $\mathcal{E}$ denote the set of edges in G.

- Technically, edges are <u>unordered</u> pairs of vertices, since the edges in $\mathcal{E}$ are undirected. For this reason, sometimes set notation, (i, j), is used to represent edges in an undirected graph.

- When programming in Python, we have to be careful about undirected edges, because we have to use tuples (which are naturally ordered) to represent edges as keys in dictionaries. The following edge-naming convention works well:

  ○ Use positive integers to label the vertices: $\mathcal{V} = \{1, 2, \ldots, |\mathcal{V}|\}$. Let (i, j), <u>where $i < j$,</u> represent the edge connecting vertices i and j.

  ○ We have to keep this convention in mind as we model in Python, so we will use the same naming convention in our handwritten models.

- The following is some graph terminology that we will be using.

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

> A **cycle** is a sequence of connected edges that begins and ends at the same vertex. In a cycle, no edge is repeated, and the only vertex that is visited twice is the first vertex (which is also the last).

> 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.
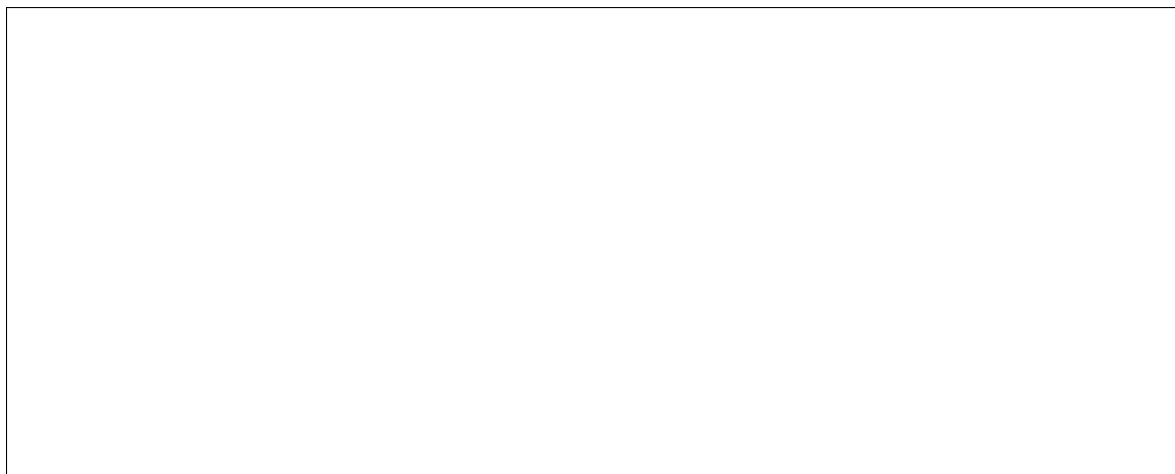
Problem 2. Convince yourself of the previous fact by drawing a connected graph on 5 vertices with a single cycle. Can you remove any edge from the graph without losing connectivity?
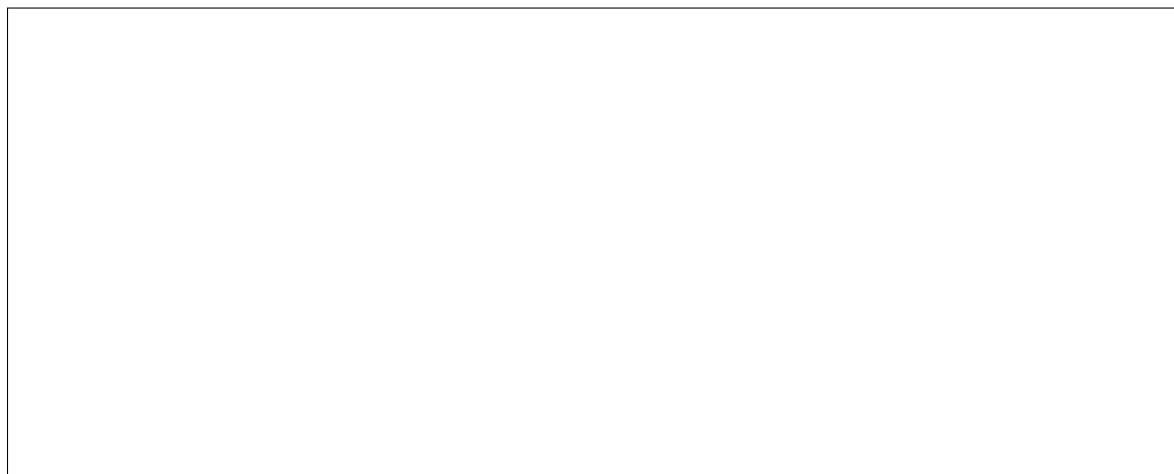
A connected graph that contains no cycles is called a **tree**.

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

Problem 3. 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?

A **spanning tree** of a graph G is a subset of the edges of G that form a *tree that connectes every vertex in* $\mathcal{V}$.

Problem 4. Consider any graph G having 5 vertices.
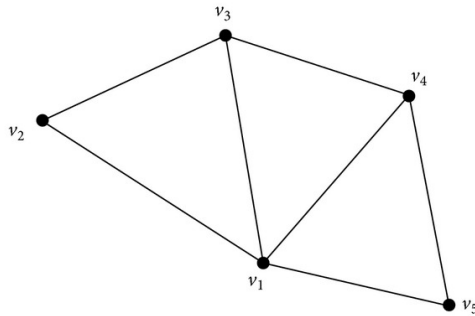
(a) Draw two different spanning trees of G, one in each box.

(b) How many edges does each spanning tree have?

(c) List the elements (edges) of the spanning tree you drew on the right using the naming convention from page 1.

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

4

4   Minimum Spanning Tree (MST) Problem

> Let c_{ij} be the cost associated 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.

Problem 5. Think of a military problem that could be solved by finding a minimum spanning tree.

Problem 6. Solve the minimum cost spanning tree problem for the graph shown below. (The numbers on the edges are the edge costs.)
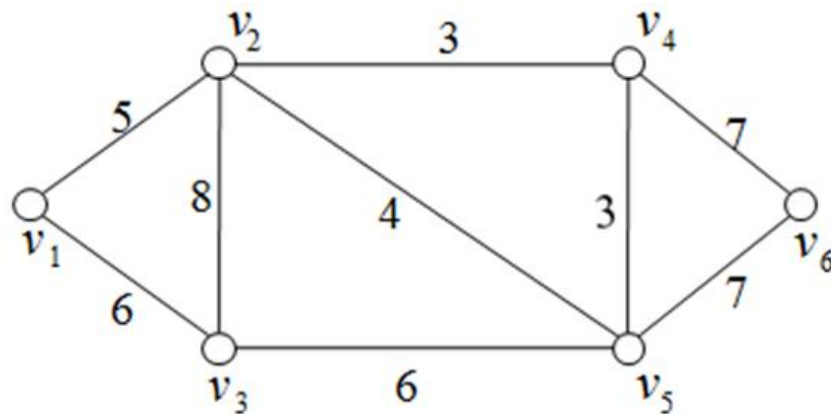
Minimum spanning tree $T =$

(Provide the set of edges of the minimum cost spanning tree. Just listing the costs of the edges is not enough information to define the solution.)

5 MST IP Formulation

Note: it is more efficient to solve a min-cost spanning tree problem via an algorithm (you might learn such an algorithm in SA403). However, as is the case for many combinatorial optimization problems, an IP formulation does not require you to write an algorithm, and can it can allow you to easily incorporate more complicated modeling elements.

Problem 7. 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?

(a) Label the rows and columns of matrix C to represent nodes of the graph (using numbers).

(b) Draw the graph of the network below. Include node labels and edge costs.
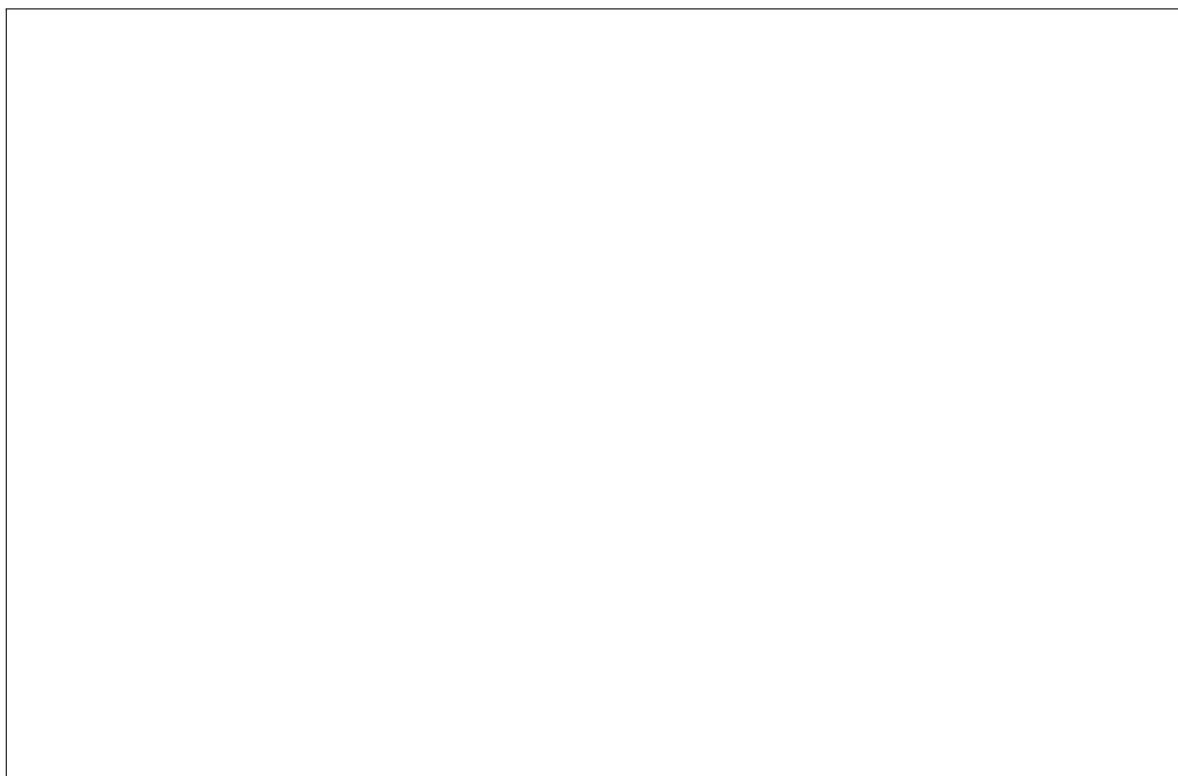
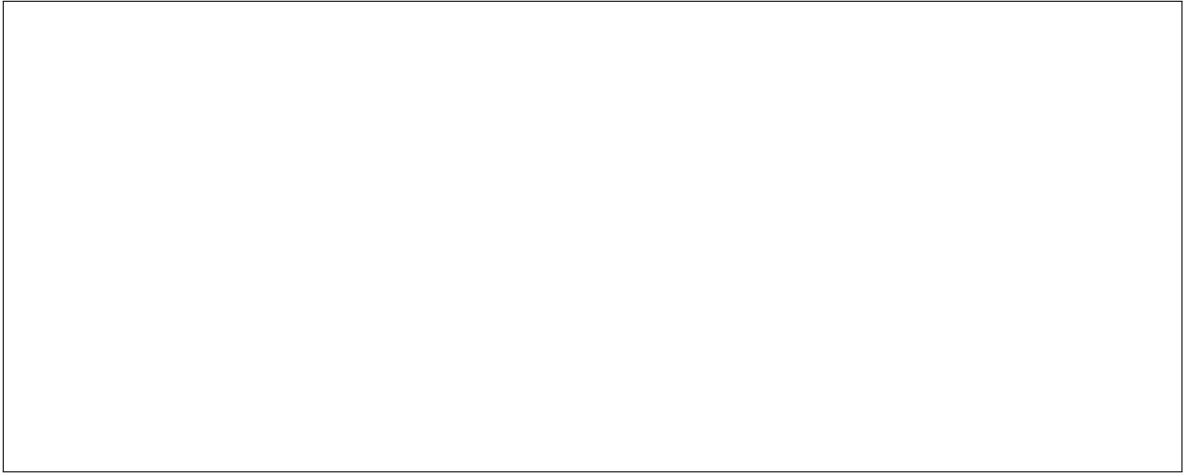(c) Define the **sets**, **variables**, and **parameters** for the problem. Remember the edge naming convention.

(d) Write the objective function in concrete form, and then in abstract form.

(e) 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 abstract class of constraints that ensures that every vertex is covered by an edge.
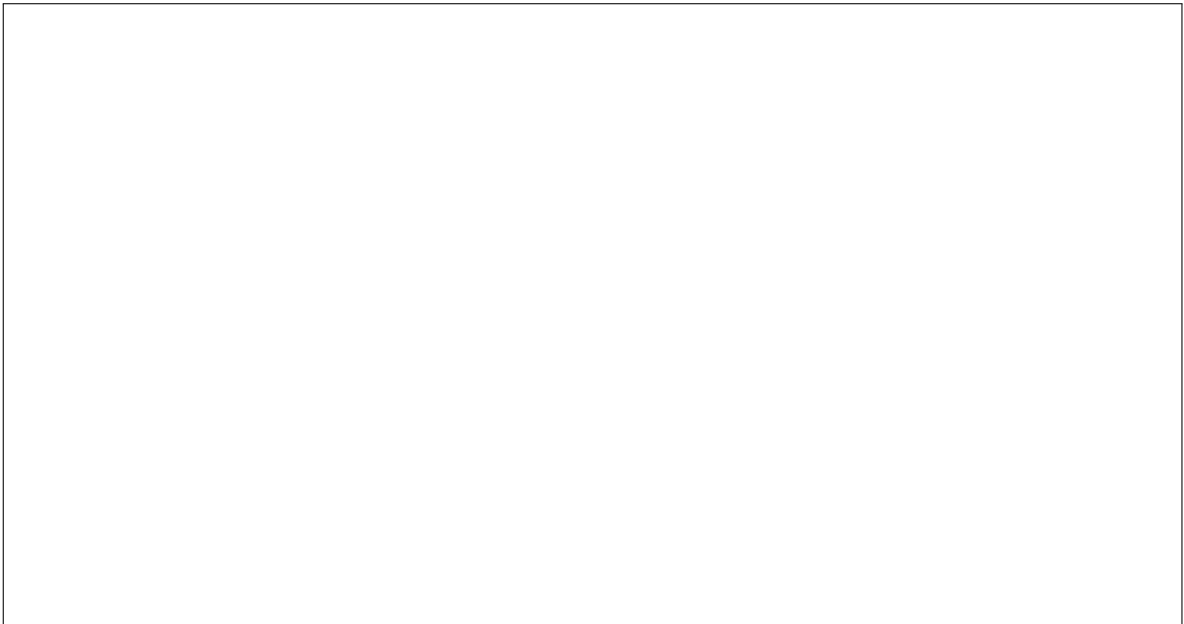
(f) Spanning trees contain [____] edges. Write a concrete constraint that ensures that the solution returned by the solver has exactly [____] edges. Convert this constraint to abstract form.

(g) Do we have all of the constraints that we need? Can you think of a subgraph of the graph G associated with this problem that contains the correct number of edges and touches every vertex, but is not connected? Sketch this graph below.

(h) Write a concrete constraint that prevents the graph that you sketched above. Then write a set of abstract constraints that prevents ANY graphs of this kind from being returned by the solver. (There is one such constraint for every subset S of vertices of G, such that $|S| \geq 3$.)

The constraints above are called **cycle-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.