ENGRI 1101 | Engineering Applications of OR | Fall 2020 | Lab 3

The Minimum Spanning Tree Problem

Name: ____________________

Objectives:

- Introduce students to the graph theoretic concept of spanning trees.
- Show three different combinatorial algorithms for solving the minimum spanning tree problem.
- Demonstrate a practical use of minimum spanning trees.

Optional Reading Assignment:

- Read Handout 4 on the minimum spanning tree problem.

Brief description: In this lab, we review some of the applications of the minimum spanning tree problem, along with the concept of the spanning tree in an undirected graph (and why these are the desired solutions for the problem), some algorithms for solving the minimal spanning tree (MST) problem, and sensitivity analysis for this problem.

1 An application: communication network design

You are the engineer in charge of designing a new high speed fiber optic Internet network between several Operations Research departments throughout the U.S.. Your objective is to design a system that connects various campuses. However, so that this network can be brought online quickly, we must install the fiber optic line within existing physical infrastructure. The possible physical cable routes between cities and the cost of installing the fiber optic cable (in millions of dollars) are given by the data shown on the graph in figure **??**. How do you suppose you would go about designing such a system? Since you can only use the edges shown in the attached graph, you must choose a subgraph of the given graph, or in other words, a subset of the possible edges. Every location must be serviced which means that the subgraph must be spanning. You should be able to get to any location from any other location. This means the subgraph should be connected. Because you are trying to minimize cost, the subgraph should also be minimal, meaning that you cannot remove any of the edges while maintaining the other necessary properties. A minimal connected spanning subgraph is called a spanning tree. There are many other ways of defining trees. In operations research terminology, we want to find a minimum spanning tree of the given graph.

An example of a spanning tree is indicated using thick edges in figure **??**. Do you think that this is the best possible? Can you briefly give a convincing argument why or why not?

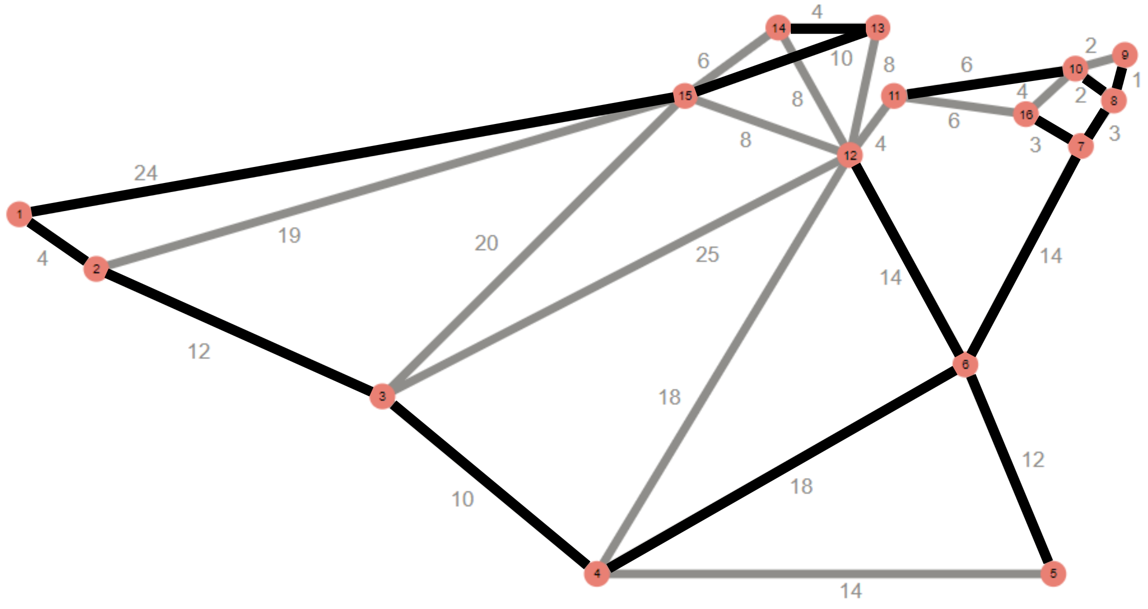


Figure 1: A spanning tree of the communications network.

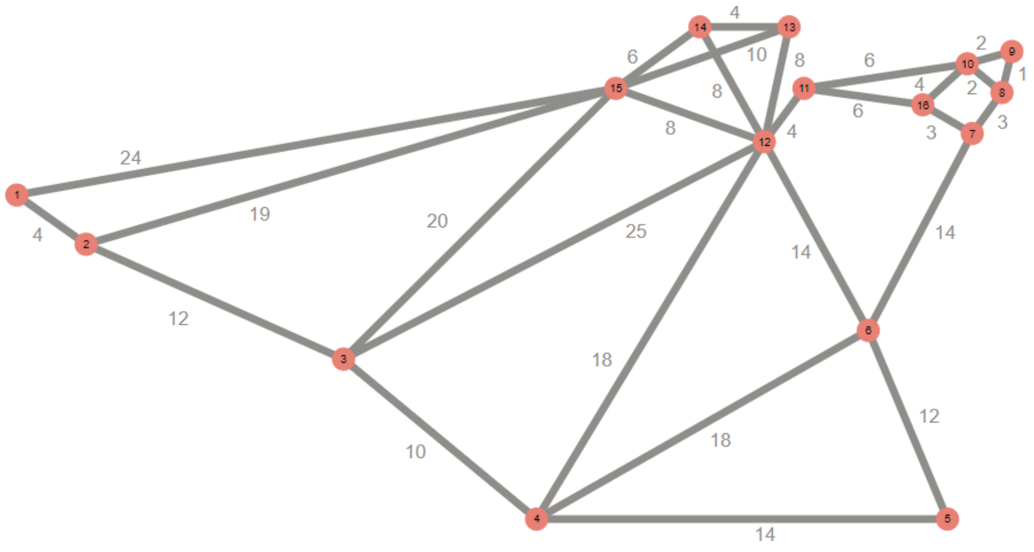
2 Minimum spanning tree algorithms

In this section we will investigate three different algorithms for solving the Minimum Spanning Tree Problem.

First you will try the algorithm that you have seen on lecture: Prim's algorithm. This algorithm works as follows:

1. Choose any node from which to begin, say node 1, and start the tree with the cheapest edge from node 1 to one of the other nodes in the graph.
2. At each subsequent step, add the cheapest edge that maintains connectivity of the current tree and adds a new node. In other words, add the cheapest edge that connects a new node to those already connected.
3. Continue to do this until all nodes are connected.

Run the first 5 iterations of this algorithm on the map below by hand. Show your work: indicate the order in which you add the edges (e.g. by clearly writing 1 next to the first edge you added, 2 next to the second edge you added, etc).


14
4
13
6
10
8
6
2
9
10
1
4
2
8
15
11
16
8
8
6
3
3
4
12
7
24
1
19
20
25
4
2
14
14
12
6
3
18
12
18
10
4
14
5

We will now let the computer help us. To get set up, do the following:

- Go to `http://engri1101.orie.cornell.edu/` on a web browser.
- Click on `Minnimum Spanning Tree`.

Now use the software to run the first 5 iterations of Prim's

- Click `Prim's` within the list of commands at the bottom of the screen (it should already be clicked for you!).
- Click the node labeled number 1 to start running Prim's from node 1. Node number 1 should turn blue, and you will be prompted "Click the first edge to start."
- Click the first edge you labeled while starting Prim's by hand. If that was the correct first edge to add, the edge will darken. If it was not the correct edge to add, it will flash red and allow you to try again.
- Repeat this process, iteratively the first five edges of the minimum spanning tree.

Did you make any mistakes during your first five iterations?

The program uses colors to distinguish different types of information. Using your observations about the first 5 iterations, explain when each of the following represents.

- red nodes
- blue nodes

Hit the `Hint` button at the bottom of the graph. Several edges will flash light blue, and in the next iteration you will add one of these edges. Which of the highlighted edges should you add next?

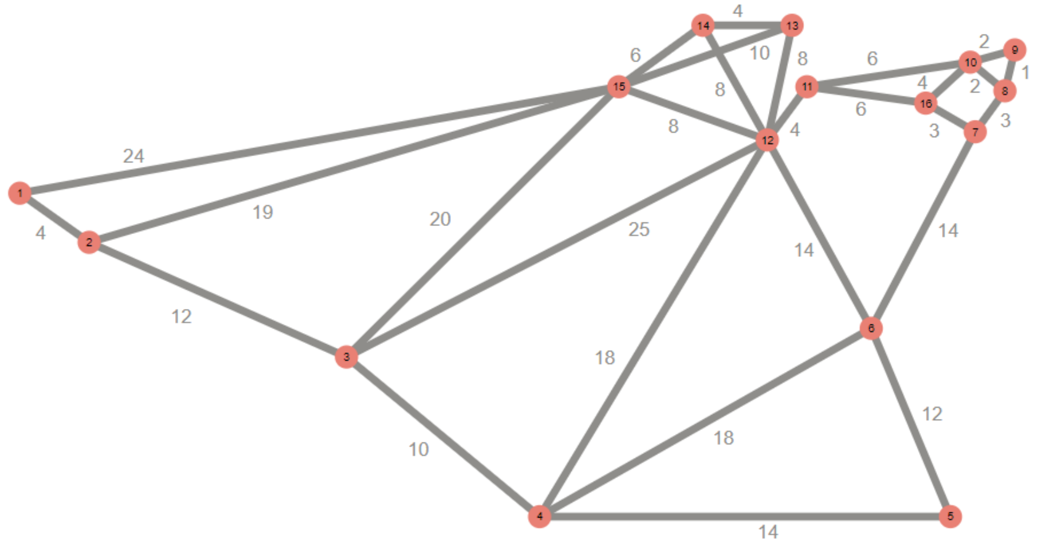
What characterizes which edges the program flashes? For example, which edge(s) would the `Hint` feature have flashed after you had only added the first edge? Feel free to restart the program and check your work.

Now continue with the program until it completes the computation of a minimum spanning tree. Add the edges one at a time, predicting what the algorithm will do in its next step. What is the total cost of this tree, and how does this compare with the original tree?

Now we turn to another algorithm which is called Kruskal's Algorithm. This is an example of a so- called Greedy Algorithm, i.e. an algorithm that always takes the step that "looks best" currently. Greedy algorithms are widely used in computer science beyond just solving the MST. Kruskal's algorithm works as follows:

1. Begin with the cheapest edge (break ties arbitrarily).
2. At each step, add the cheapest edge not already in the system that does not create a cycle, or a loop in the system.
3. Continue adding edges until you get a spanning tree.

Run the first 5 iterations of this algorithm by hand on the map below.



Indicate edges that you have decided to include on the above blank copy of the map, and also indicate (differently) any edges of these that you you considered adding (but decided not to because they created a cycle or loop). Again, specify the order in which you are adding the edges.

Now you will trace a complete execution of Kruskal's algorithm on this data. You can start Kruskal's algorithm by clicking `Kruskal's` on the bottom of the page, and then following prompts: click the edges to add them in order.

First start by comparing the first 5 iterations of the computer's run with your first 5 iterations. Did you make any mistakes?

Now continue running the algorithm until it finishes. As before, try to anticipate each of the algorithm's steps. What is the cost of the final solution?

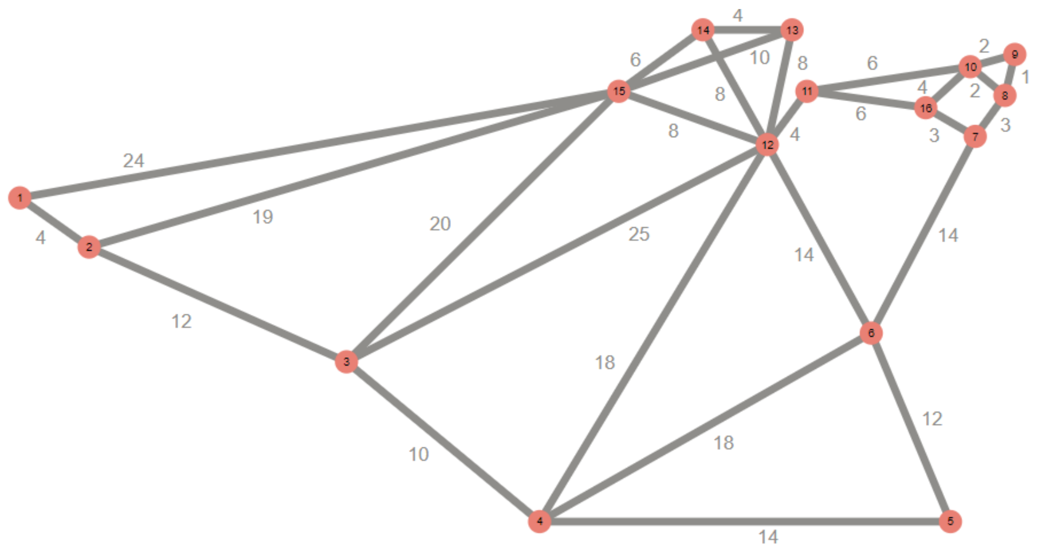
Is the same spanning tree found by the two algorithms?

(Why is the previous question not a dumb question?)

The following algorithm is called the Reverse Greedy Algorithm, and it effectively does Kruskal's in reverse.

1. Start with the entire graph.
2. At each step, check if the graph has a cycle. If it does, remove the most expensive edge in the cycle (break ties arbitrarily, and pick any cycle you'd like).
3. Continue to do this until the graph remaining is a spanning tree.

Try the algorithm by hand on this last blank copy of the map. Do the first five iterations by hand. Then use the software to check your work: Start Reverse Kruskal's by clicking `R-Kruskal's` and then hit Fast-Forward, watching the software remove edges one at a time. Note: this software works a little bit different than the above process, and it always looks for the most expensive edge it can eliminate. It will run a version of Reverse Kruskal's, but won't run in the full generality as above.



What is the cost of the resulting spanning tree?

How does the spanning tree and its cost compare to those obtained by the previous algorithms?

Which of the three algorithms was the easiest for you to follow? Why?

3 Analyzing minimum spanning trees

Suppose you start with some spanning tree, like the first one given in your lab handout, can you devise a way to systematically improve it? In other words, given a spanning tree, can you tell if it is one of minimum cost and, if not, can you improve it (without recomputing a minimum spanning tree from scratch)? Suggest such an algorithm.

Next we will study how the solution changes when problem parameters are altered. This is referred to as Sensitivity Analysis. Consider edge $\{4, 6\}$ which was not used in the minimum spanning tree found by Prim's algorithm. Just this one edge's cost will be changed. Should it increase or decrease if it will be included in the new minimum spanning tree? Exactly what must the cost of $\{4, 6\}$ be changed to for this to occur?

Check your work using the software:

- Click `Sensitivity`
- Enter node number 4 into the prompt and hit OK
- Enter node number 6 into the prompt and hit OK
- Enter 1 less than the cost you answered above[1] and hit OK.
- Click `Prim's`, click node 1, and then click `Fast Forward` to quickly rerun Prim's with the new edge length

Is the edge $\{4, 6\}$ now in the tree? Follow the above process and reset the length of the edge $\{4, 6\}$ to be 2 more than it currently is[2] and re-run Prim's from node 1. Is the edge $\{4, 6\}$ no longer in the tree?

In general, what does this suggest about how the cost of any one "not included" edge must be changed if it is to be included in the minimum spanning tree for the modified data?

Now consider an edge that is in the minimum spanning tree, such as $\{1, 2\}$. How must this be changed for

[1]For example, if you answered that the cost must be 7 for it to be included in the new MST, you'd enter 6 at this step.

[2]E.g. if you just reset the cost of edge $\{4, 6\}$ to be 6, you'd now set it to be 8 – one more than you answered above

this edge in order for this edge to be forced out of the optimal solution? Again, also figure out the general rule for forcing minimum spanning tree edges out of the minimum spanning tree.