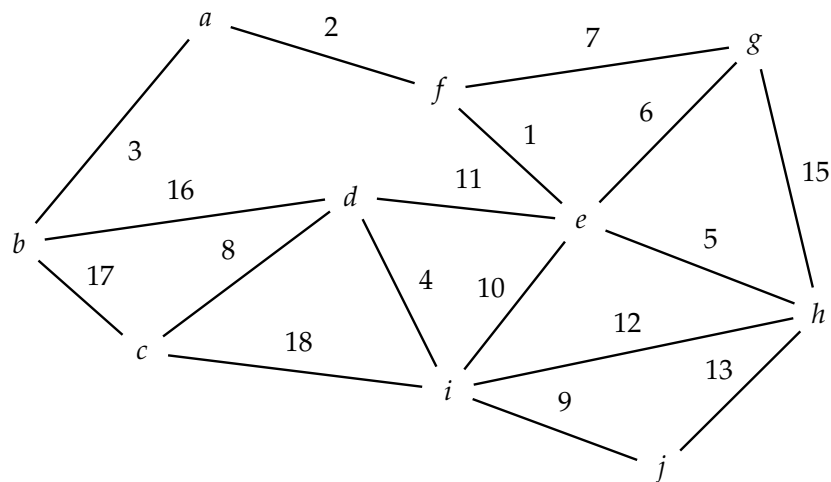


Algorithms: Minimum Spanning Trees

Model 1: Minimum-weight Spanning Subgraphs (13 mins)

Definition 1. Given an undirected graph $G = (V, E)$, a *spanning subgraph* of G is a connected subgraph $G' = (V, E')$ of G ; that is, a connected graph with the same vertices as G and a subset of its edges.

Definition 2. Given a *weighted*, undirected graph G , the *minimum-weight spanning subgraph* (MWSS) of G is the spanning subgraph of G whose total weight (i.e. the sum of the weights of all its edges) is as small as possible.



- 1 Draw any spanning subgraph of the example graph from Model 1.
- 2 Can a graph G have more than one spanning subgraph?
- 3 Is it possible for a graph G to be a spanning subgraph of itself?
Why or why not?
- 4 Is it possible for a disconnected graph to have a spanning subgraph? Why or why not?

- 5 Can a graph G have more than one minimum-weight spanning subgraph? Give an example, or explain why it is not possible.
- 6 Find a MWSS for the graph in Model 1. Does it have more than one?
- 7 Which of the following scenarios could be modeled by finding a MWSS?
- (a) A railroad company wants to connect a given set of cities by train routes as cheaply as possible. Connecting two cities by a route costs an amount of money proportional to the distance between them.
 - (b) A company wants to network a set of data centers with high-speed fiber optic connections, and they want to minimize the total amount of fiber optic cable used. There must be at least two routes between any pair of data centers, so that any one fiber optic link going down will not disconnect the network.
 - (c) Given a network of train routes between a collection of cities and the cost of each route, find the cheapest route between two given cities.
 - (d) You have the latest Megazorx puzzle toy, which can be in any one of a number of different states. There are various moves which can transform the Megazorx from one state to another. In order to impress your friends, you want to be able to transform it from any starting state into any other requested state (which may in general require a whole sequence of moves). You want to be able to do this
 - (i) ... using the fewest number of moves possible for each given pair of start and end states.
 - (ii) ... without having to memorize any more moves than absolutely necessary.



8 (Review) What is the definition of a *tree* graph?

9 Prove: a MWSS must always be a tree.

Hint: use a proof by contradiction.



Because of the result from Question 9, we typically refer to a minimum-weight spanning subgraph as a *minimum spanning tree* (MST). Now that you understand the definition of a MST and some scenarios that MSTs can be used to model, let's explore some algorithms for finding them.

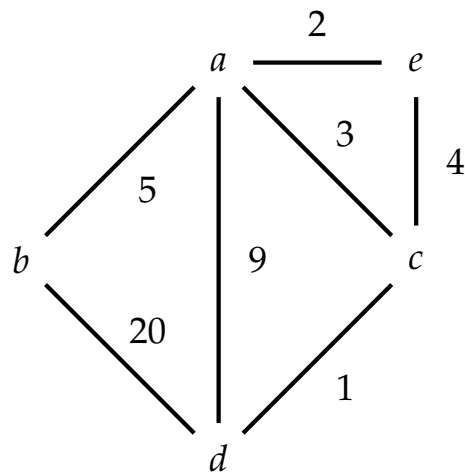
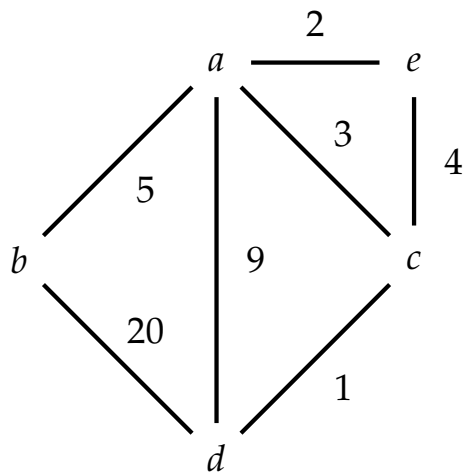
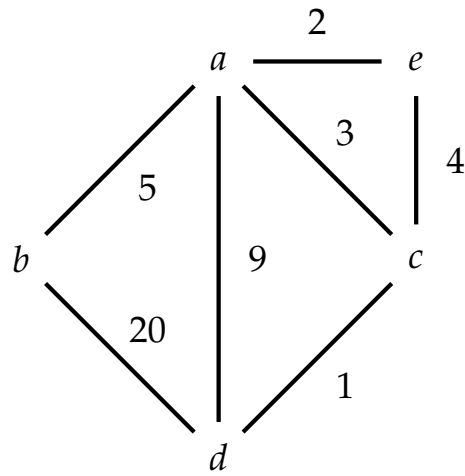
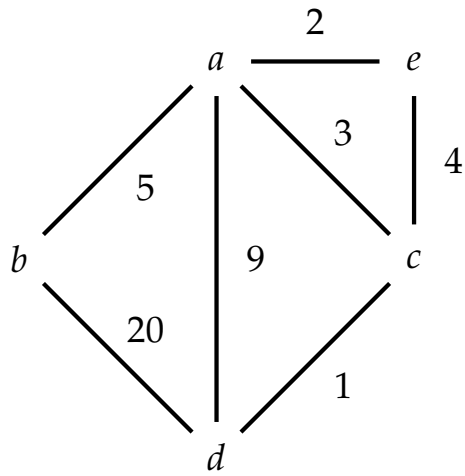
Model 2: Four algorithms (15 mins)

Given a weighted, undirected graph G , the goal of each algorithm is to pick a subset of the edges of G which constitute a MST.

- **(Kruskal's)** Consider all the edges in order from smallest to biggest weight; for each edge, pick it if and only if it does not complete a cycle with previously chosen edges.
- **(Prim's)** Choose an arbitrary vertex to start and mark it as visited. At each step, pick the smallest edge which connects any visited vertex to any unvisited vertex and which would not complete a cycle with previously chosen edges; mark the other end of the edge visited.
- **(Johnson's)** Choose any vertex to start. At each step, pick the smallest edge connected to the current vertex which has not already been chosen and would not complete a cycle with previously chosen edges. Repeat until running out of options; then pick a new starting vertex and repeat the entire process. Do this until all vertices are connected.
- **(Reverse delete)** Start with all edges initially "picked", and consider them in order from biggest to smallest weight. For each edge, throw it out (*i.e.* "unpick" it) if and only if doing so would not disconnect the remaining edges.

- 10 On the next page are four copies of the same graph. Use these to trace the execution of each algorithm in the model.
- 11 One of these algorithms is fake. Which one? Give an example showing why it does not work.





Model 3: The Cut Property (15 mins)

Definition 3. A *cut* in a graph $G = (V, E)$ is a partition of the vertices V into two sets S and T , that is, every vertex is in either S or T but not both. We say that an edge e *crosses* the cut (S, T) if one vertex of e is in S and the other is in T .

Theorem 4 (Cut Property). Given a weighted, undirected graph $G = (V, E)$, let S and T be any partition of V , and suppose e is the smallest-weight edge crossing the (S, T) cut. Then any minimum spanning tree of G includes e .

- 12 Give three examples of cuts in the graph from Model 1 and identify the smallest edge crossing each cut.

Let's prove the cut property.

Proof. Let G be a weighted, undirected graph $G = (V, E)$, let S and T be an arbitrary partition of V into two sets, and suppose $e = (x, y)$ is the smallest-weight edge with one endpoint in S and one in T . We

wish to show that _____.

The proof is by contradiction. Suppose T is a _____

but _____. We will show this is impossible.

Since T is a _____ it contains a unique _____

between any two _____. So consider the unique _____

in T between _____. It must cross the cut at least once since

_____ ; suppose it crosses at $e' = (x', y')$. Take T and

replace _____ with _____ ; the resulting graph is

still _____ because _____,

but it has a smaller _____ because _____,



which contradicts our assumption that T was _____.

Therefore _____ □

The cut property can be used to directly show the correctness of several MST algorithms. Let's prove the correctness of Kruskal's Algorithm; the proofs for the other algorithms are similar.

Theorem 5. *Kruskal's Algorithm is correct.*

Proof. Suppose at some step the algorithm picks the edge $e = (x, y)$. Let X be the set of vertices connected to x by edges which have been picked so far (not including e), and let Y be all other vertices. $x \in X$

by definition. We know that $y \notin X$ since if it was, e would _____

but then Kruskal's Algorithm wouldn't _____.
 e therefore crosses the cut (X, Y) . No other edges which have been

picked previously cross the cut, since _____.

Therefore e must be the _____

because _____.

Therefore by the Cut Property e must be in any MST and Kruskal's Algorithm is correct to pick it. □

