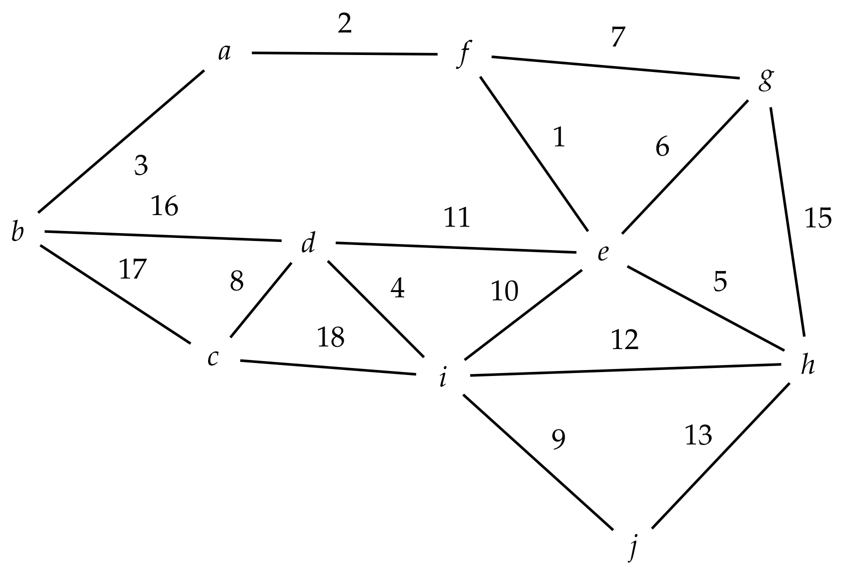
Algorithms: Kruskal's Algorithm (key)

In the previous activity you learned about minimum spanning trees and experimented with several different algorithms for finding them. In today's activity we will focus on Kruskal's Algorithm and prove that it works correctly.

Model 1: Kruskal's Algorithm (12 mins)



```
Require: Undirected, weighted graph $G = (V, E)$
1: $T \leftarrow \varnothing$                              ▷ $T$ holds the set of edges in the MST
2: Sort $E$ from smallest to biggest weight
3: for each edge $e \in E$ do
4:     if $e$ does not make a cycle with other edges in $T$ then
5:         Add $e$ to $T$
```

1 Simulate Kruskal's Algorithm on the graph in Model 1. What is the total weight of the resulting spanning tree?

All edges from 1–10 except edge 7 are included; the total weight is 48.

2 The way the algorithm is written in Model 1, one must iterate through every single edge in E. However, this is not always neces-

sary. Can you think of a simple way to tell when we can stop the loop early?

Since a tree with V vertices has exactly $V - 1$ edges, we can just count how many edges have been picked so far and stop when we get to $V - 1$.

3 Explain why even in the worst case, $\Theta(\lg V) = \Theta(\lg E)$ in any graph.

In the worst case, E is $O(V^2)$, in which case $\Theta(\lg E) = \Theta(\lg V^2) = \Theta(2 \lg V) = \Theta(\lg V)$.

4 In the above algorithm, how long does line 2 take? Simplify your answer using the observation from the previous question.

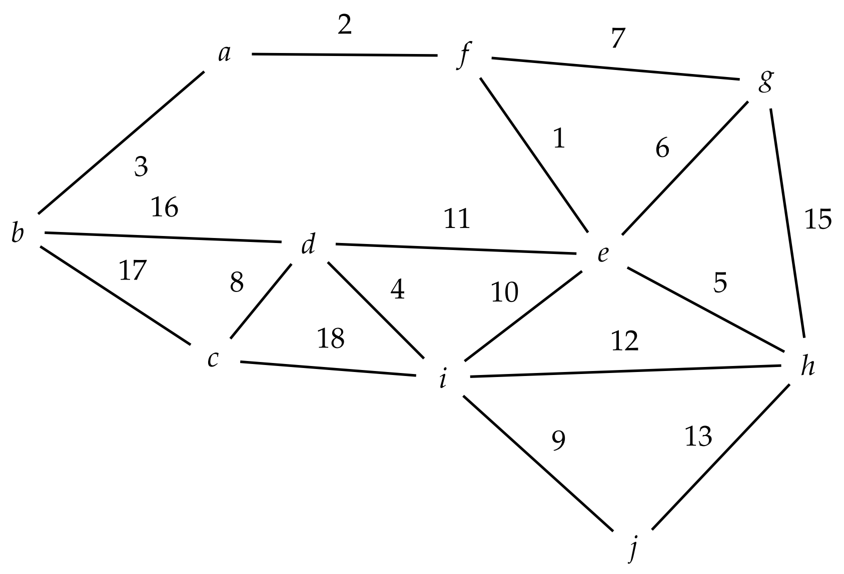
We are sorting a list of E edges; if we use an efficient $\Theta(n \lg n)$ sorting algorithm, it will take $\Theta(E \lg E) = \Theta(E \lg V)$ time.

5 Can you think of a way to implement line 4? How long would it take?

Do a DFS (or BFS) in T from one endpoint of e. e makes a cycle if and only if the other endpoint of e is reachable. This would take $\Theta(V)$ time (BFS or DFS in general take $\Theta(V + E)$ time, but in this case since T has no cycles it can't have more edges than vertices). Hence the total time for the whole algorithm would be $\Theta(E \lg V) + \Theta(E \cdot V) = \Theta(VE)$.

STOP

Model 2: The Cut Property (20 mins)



Definition 1. 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 2 (Cut Property). *Given a weighted, undirected graph $G = (V, E)$, let S and T be any partition of V, and suppose e is some edge crossing the (S, T) cut, such that the weight of e is strictly smaller than the weight of any other edge crossing the (S, T) cut. Then every minimum spanning tree of G must include e.*

6 Give three examples of cuts in the graph from Model 2 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 X and Y be an arbitrary partition of V into two sets, and suppose $e = (x, y)$ is the smallest-weight edge with one endpoint in X and one in Y. We wish to show that $\underline{\text{every MST of } G \text{ must include } e}$.

We will prove the contrapositive. Suppose M is a spanning tree of G which does **not** contain the edge e. Since M is a $\underline{\text{tree}}$ it contains a unique $\underline{\text{path}}$ between any two $\underline{\text{vertices}}$. So consider the unique $\underline{\text{path}}$ in M between $\underline{x \text{ and } y}$. It must cross the cut at least once since

$x \in X$ and $y \in Y$; suppose it crosses at $e' = (x', y')$, with $x' \in X$ and $y' \in Y$. We know that the weight of e is smaller than the weight of e', since we assumed e is the smallest-weight edge crossing the cut. Now take M and replace e' with e; the result is still connected because any path that went through e' can detour around through e, but it has a smaller total weight because we removed e' and replaced it with e which has a smaller weight.

So, we have shown that any spanning tree M which does not contain the edge e can be made into a smaller-weight spanning tree, which means that M is not a minimum spanning tree. □

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 3. *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 make a cycle but then Kruskal's Algorithm wouldn't have picked it. e therefore crosses the cut (X, Y). No other edges which have been picked previously cross the cut, since by definition the vertices in Y aren't connected to x. Therefore e must be the smallest edge crossing the cut because the edges are considered in order from smallest to biggest weight. Therefore by the Cut Property e must be in any MST and Kruskal's Algorithm is correct to pick it. □