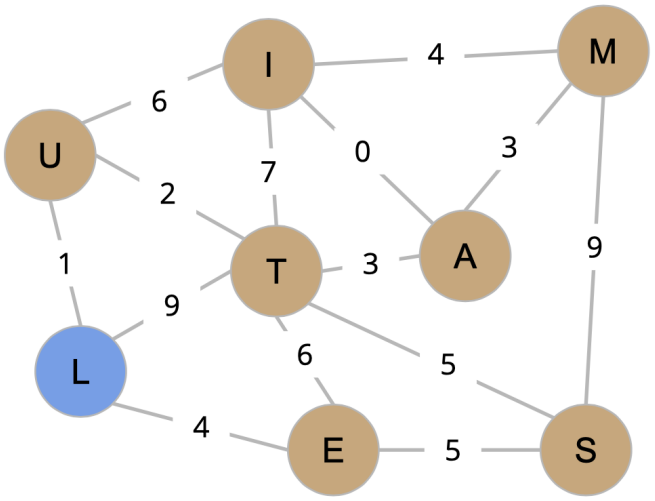
Kruskal's



(a) We want to run Kruskal's, but we have no cycle detection, so we terminate upon inserting $V - 1$ edges. Will this produce a valid MST on the graph above? If not, determine which edge(s) need to be changed, and to what. If there are many possibilities, choose the one that involves the minimum added/removed weight.

Assume ties are broken alphabetically, and edges are written in alphabetical order, and compared as such. For instance, if edges (A, Z) and (E, H) are equal, (A, Z) would be chosen before (E, H).

Solution:
This will not produce a valid MST. The problem is that we consider `IM` before adding the last vertex `S` to the MST. So, for Kruskal's to work, you *either* need to

1. change `IM` to 5
2. change `ES` to 4

so that the edge `ES` is considered before `IM`.

Here is a video walkthrough of the solutions.

(b) After completing the previous part, Sohum wondered if it's possible to run Kruskal's with limited cycle detection. More specifically, he pondered: what if we can only detect a maximum of **k** cycles during one run of Kruskal's?

Looking at the specific instance of a 6 vertex graph, what is the **minimum** value of **k** for which we can ensure that Kruskal's will always work?

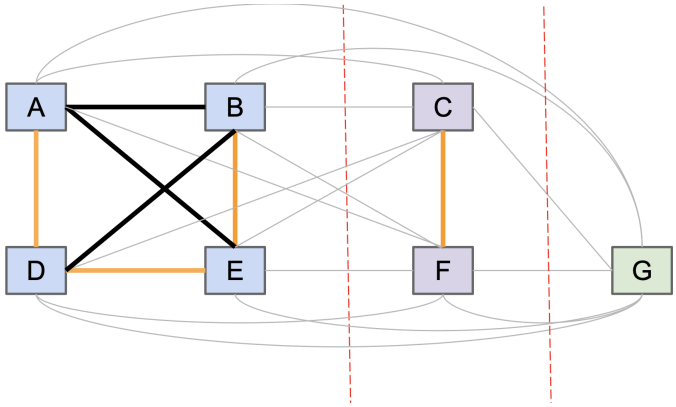
Solution: 6

Explanation:
To find the minimum value of **k** for which we can ensure Kruskal's will always work,

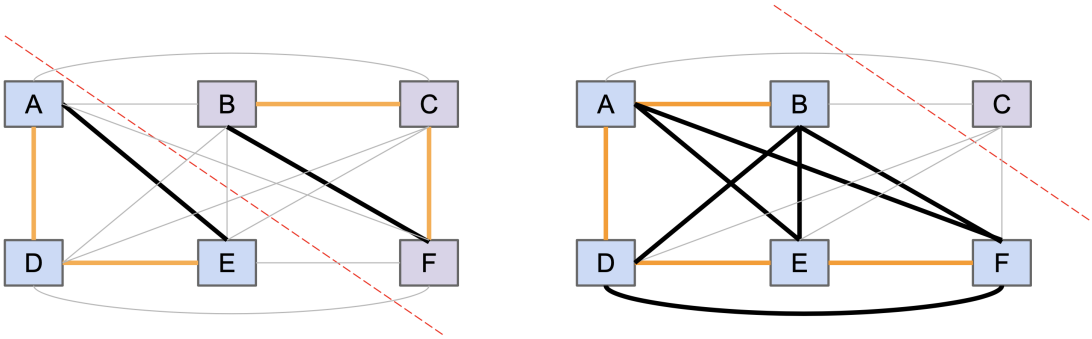
we want to find the **maximum** number of edges we could possibly consider in one run of Kruksal's before terminating. Well, when do we terminate in Kruskal's? We terminate after adding **V - 1** edges, i.e. when the MST contains all the vertices. So, if we haven't finished Kruskal's, it means that we haven't added **V - 1** edges and we have two or more components that are *not* connected to each other. In order for there to be two or more components that are not connected to each other, note that there must be some cut that we haven't considered *any* crossing edges of. For instance, looking at the partial state of an example of Kruskal's below, notice that we haven't considered any crossing edges of two cuts.

Note that:

1. orange edges: in the MST
2. black edges: edges considered but cause cycle
3. light grey edges: edges yet to be considered)



Next, since we want to find the **maximum** number of edges we could possibly consider in Kruksal's before terminating, let's focus the case when Kruskal's is *about* to finish and we have considered the maximum number of edges prior to this point. If Kruskal's is about to finish, it means there is only one more edge to be added, i.e. one more cut to find a crossing edge through. Since we want to consider the maximum number of edges, we want the number of edges in this "final" cut to be as few as possible, since the more edges there are in the final cut, the fewer edges we could've considered prior to this point. And, we want there to be as many edges as possible "outside" of this cut.



Looking at the two possible "almost finished" states of Kruskal's above, i.e. states

with two connected components remaining and one cut to find a crossing edge through, notice the number of edges in the cut on the right is far fewer than the number of edges on the cut on the left. To generalize this finding, the fewest number of crossing edges we can have in *any* cut between sets `A` and `B` occurs when the set `A` or `B` contains only one vertex. As such, the **maximum** number of edges we can consider in one run of Kruskal's occurs in the specific instance of the right graph above where the crossing edges of one "small" cut in the graph haven't been considered while every edge in the remaining graph has.

Notice that out of the edges we've considered in the right graph above, the `4` orange edges don't need cycle detection since they are part of the MST. However, for the remaining `6 black edges`, they need cycle detection, and we can say the **minimum** value of **k** is **6**.

If that was a lot, here is a video walkthrough of the solutions.