Math Explorers' Club - Graph Theory Session 2

Hugo Mainguy

1 §The Minimum Spanning Tree

There are three words in this title - what might they mean? (It might be easier going from right to left)

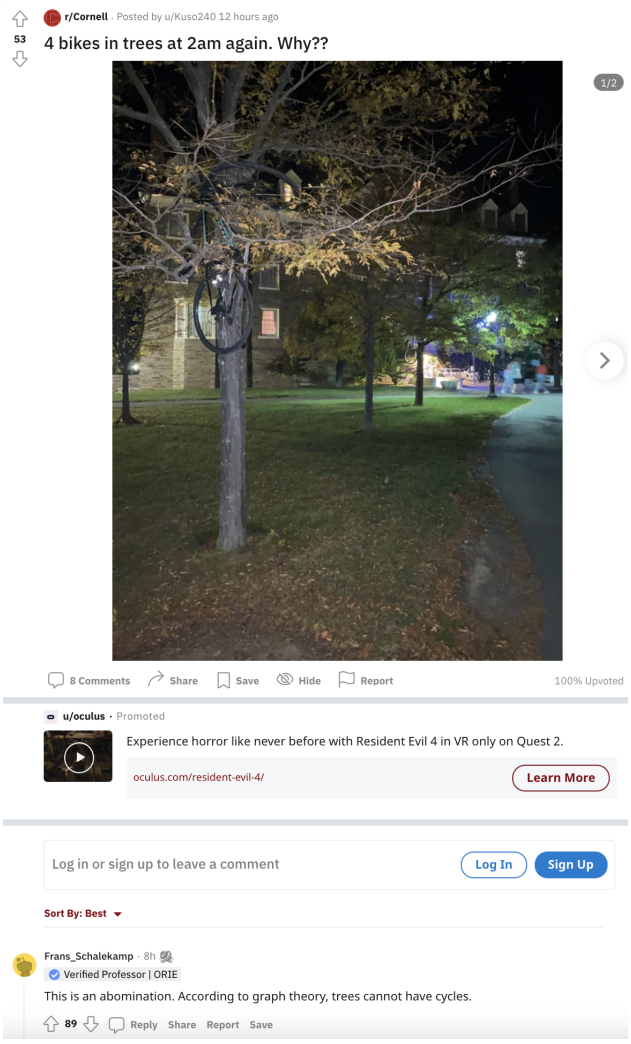


Figure 1: Just everyday posts on the Cornell Reddit - at least it may help you remember what a tree is!

Tree: A tree is a graph which has no cycles - that is, you cannot form a loop using the edges of the graph. (Imagine an actual tree with branches - the branches do not form a loop.)

Spanning: A tree is a spanning tree if every vertex is connected to every vertex by some path, so that it is possible to travel from any vertex to any other. More generally, we say a graph is connected if such a path exists between any pair of vertices.

Minimum: We want to make this tree as short as possible! We give each edge a length, and want to minimize the total length of the tree.

To make sure we understand what is going on, let's try to draw the Minimum Spanning Tree on this graph:

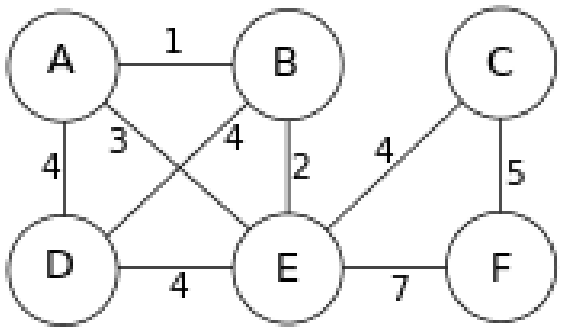


Figure 2: A graph with edge lengths are labeled. Try to find the Minimum Spanning Tree!

We can now take a look at the solution:

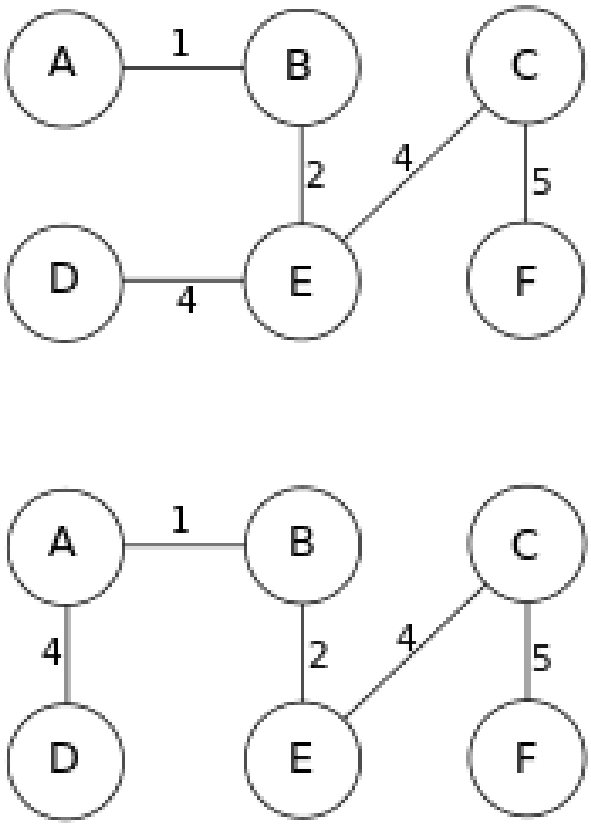


Figure 3: Two solutions to the problem above.

Why do we care about Minimum Spanning Trees? What are some potential applications?

Now suppose we know all the distances between any two pair of vertices. Can we find a systematic way to obtain the Minimum Spanning Tree? *Hints: Are there edges we know will be for sure in the Minimum Spanning Tree? Are there edges that can be excluded from consideration, once certain edges have been added to the Minimum Spanning Tree?*

There are many different approaches, but two are used most of the time, and usually work similarly and as efficiently. One thing that they both use is that shorter edges are more desirable, so we should consider them first. But there will be some edges we should not add - keep in mind we do not want to create any cycles.

Kruskal's algorithm: We start by ranking the edges from shortest to longest, then keep adding new edges one at a time - unless adding the edge creates a cycle.

Prim's algorithm: (a bit more complicated to understand): We also rank the edges from shortest to largest, but then, we add new edges if and only if they have one endpoint in the tree. So at any point in time, we have just one tree - whereas Kruskal's algorithm may cause us to have multiple trees temporarily (which is called, you guessed it, a forest).

What applications do Minimum Spanning Trees have? They were first invented for electrical grids, but can be used in transportation networks, computer networks, and many more. There are a lot of applications that sound completely disconnected, such as taxonomy (classifying species), clustering, or even describing financial markets!

2 §Dijkstra's Algorithm

Suppose you are in a city, and want to figure out what the shortest path to other nearby cities is using the available roads. Let's try from the leftmost vertex on this graph: what are the shortest paths to all the cities/vertices?

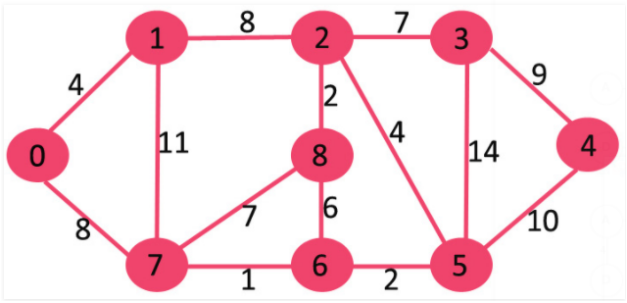


Figure 4: A graph to use Dijkstra's Algorithm.

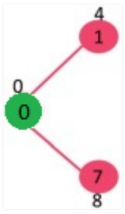
After doing this, can we find a systematic approach to obtain all the shortest path - as an algorithm, or recipe, to follow?

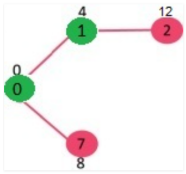
Step 1: Start by assigning a distance value from the starting vertex to a given vertex. All values are originally set to infinity, since have no edge used at first.

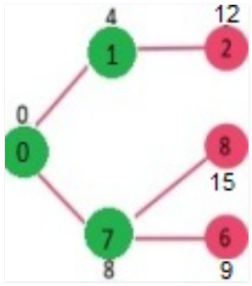
Step 2: Take the edges from the current vertex and use them to connect to the new vertices. If the new distance from the starting point to a vertex is shorter, we use this distance instead, and can remove the now unused edges.

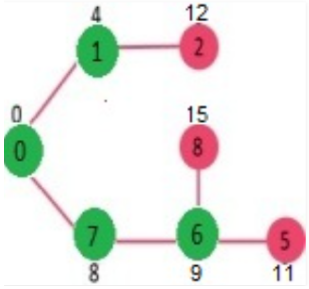
Step 3: We now select the vertex closest to the currently selected vertex, which has not yet been selected (since the distance from the origin to this vertex can now no longer be improved). We then return to step 2.

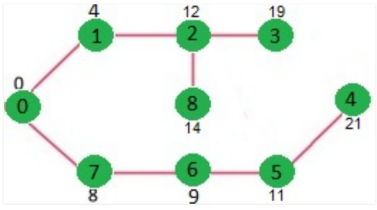
Can you apply this to the example given?











What are some of the applications of Dijkstra's Algorithm?

3 §The Traveling Salesperson (Salesman) Problem

Imagine you are a salesperson, and you need to visit a number of cities. How can we model this situation, and what would a requirement be to ensure that every city has been visited? Now what is a characteristic of the best path you want to come up with?

Now, try to find the TSP cycle for this graph (remember that a cycle means it should start and end in the same place - similarly, TSP paths do exist as well).

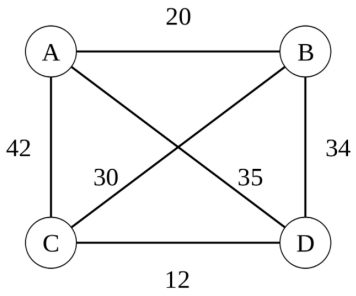


Figure 5: A graph to find the Traveling Salesperson Cycle on!

How did you come up with your solution, and how do you know it is optimal (the best)? Do you have an idea of a systematic formula to apply?

Here are some options:

Nearest Neighbor algorithm: At each step, go to the nearest vertex not visited yet.

Greedy algorithm: At each step, add the shortest edge that would not create a subcycle (this means that you would have a loop that does not include all vertices).

However, both of these may fail in practice. Can you think of examples where they would? *Note that this is a rather difficult question (both creating the heuristic, and seeing if it ever fails or proving it doesn't). But this is the kind of work that some mathematicians do!*

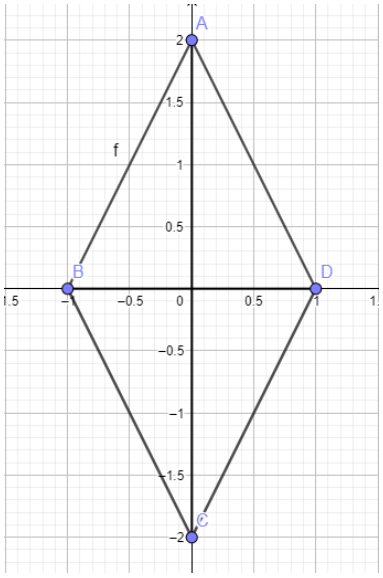


Figure 6: Does the Nearest Neighbor heuristic work here? Start at B.

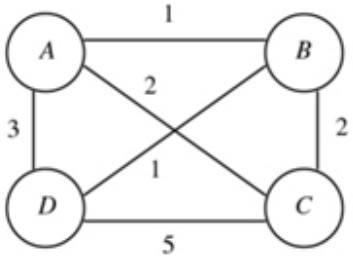


Figure 7: Does the greedy heuristic work here?

With all that being said, thank you for coming to the Math Explorers' Club modules! Hope that you liked it, and that you have gotten something out of it! If you would like to contact me with feedback, comments, or questions, you can do so at hm396@cornell.edu (please specify that your email is related to the Math Explorers' Club)!