League Of Patience Write-up
Daniel Kopeloff

A) Describe an algorithm solution to this problem. Feel free to talk about how you would adapt an algorithm we covered in class.

We could use Dijkstra algorithm to solve this since none of the weights are going to be negative .

B) What is the complexity of your proposed solution in (a)?

The Time complexity of the algorithm I suggest is O(n log(n)) since we have to go through n-1 veritices and every edge.

(c) See the file LeagueOfPatience.java, the method "genericShortest". Note you can run the LeagueOfPatience.java file and the method will output the solution from that method. Which algorithm is this genericShortest method implementing?

It is implementing Bellman-Ford Algorithm.

D) In the file LeagueOfPatience.java, how would you use the existing code to help you implement your algorithm? The existing code only handles one piece of data per edge, so describe some modifications. Note the helper methods available to you, including one that simulates the game's API that returns the next quest time.

I would use the existing code as a way to traverse through the graph. I would have to add some checks for the quest times by comparing the their start time + length of the quest to find the total time for each edge. Once I have the total time of the edge I can compare it to other edges and see which is better move.

E) **What's the current complexity of "genericShortest" given V vertices and E edges? How would you make the "genericShortest" implementation faster? Describe any algorithm changes or data structure changes. What's the complexity of the optimal implementation?**

Right now, the complexity is O(V*E) because it’s going through all the edges for all the vertices. To make it better is by identifying if your ith iteration did not change anything and then just stopping. This way you do not have to go through each iteration doing nothing.