Soham Shah

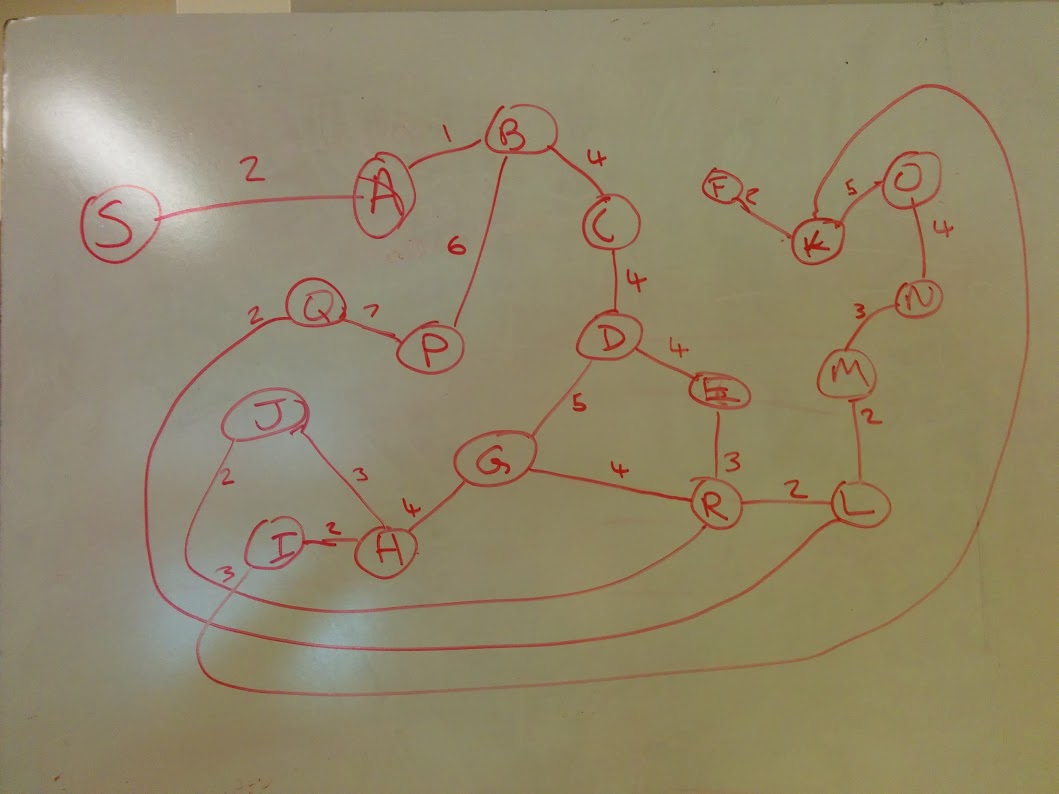
Professor Hoenigman

Intro to AI

Assignment 3:

Purpose: the purpose of this assignment is to generate a graph from a text file and then see which graph search algorithm is more efficient – A\* search or Dijkstra’s

Here is a diagram of the graph that gets generated based on the text file.



Procedure: I implemented the graph as a dictionary in the Graph class with each key being the node and the values being a dictionary of the adjacent nodes with the values for that being the weights. Afterwards, I was able to implement both Dijkstra’s and A\* Star search on this graph.

Results:

The result for Dijkstra’s was:

{'Path': ['S', 'A', 'B', 'C', 'D', 'G', 'H', 'I', 'K', 'F'], 'Nodes Evaluated': 60, 'time taken': 0.00012612342834472656, 'Algorithm': "Dijkstra's", 'Distance': 26}

The result for A\* search was

{'Path': ['S', 'A', 'B', 'C', 'D', 'G', 'H', 'I', 'K', 'F'], 'Nodes Evaluated': 34, 'time taken': 6.103515625e-05, 'Algorithm': 'A\* Search', 'Distance': 26}

Analysis of results:

As is shown above, both algorithms calculated the same shortest path based on edge weights. However, A\* search is markedly superior when it comes to complexity because it guaranteed finding the shortest path while also evaluating about half as many nodes as Dijkstra’s thanks to using a heuristic to estimate the best possible path. This shows in the times taken as well as A\* takes about half the time Dijkstra’s takes to evaluate.