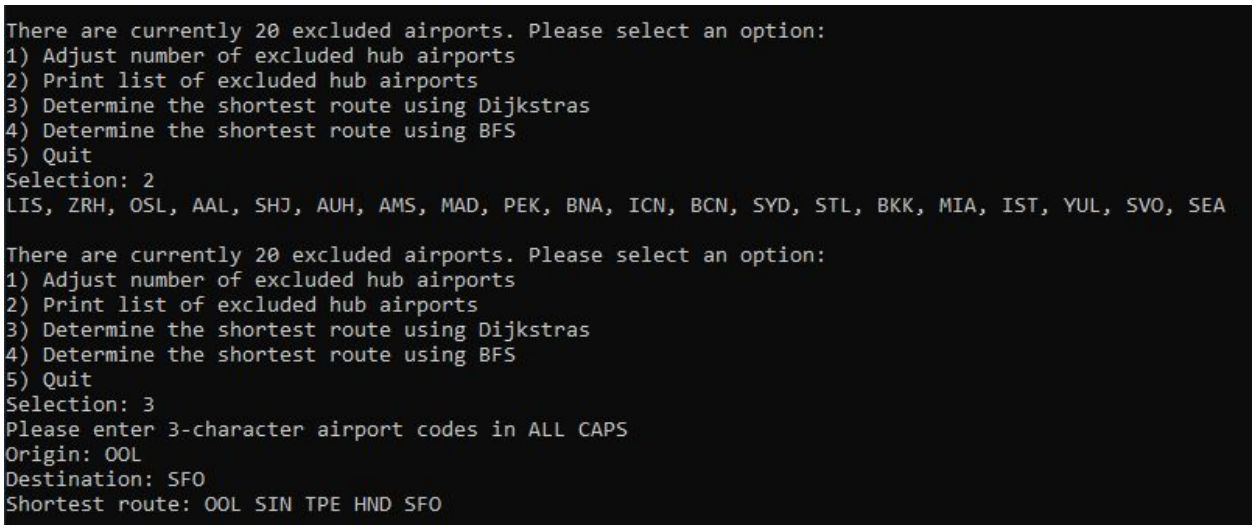
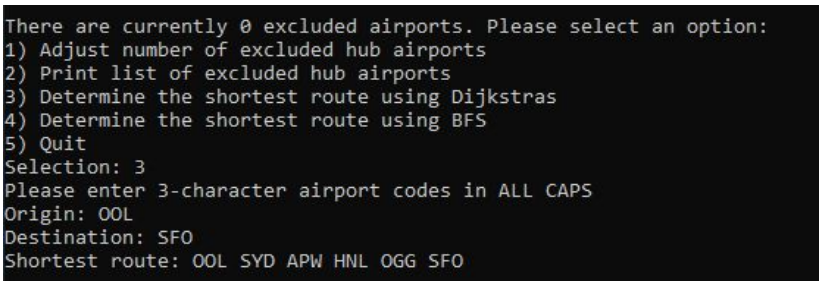
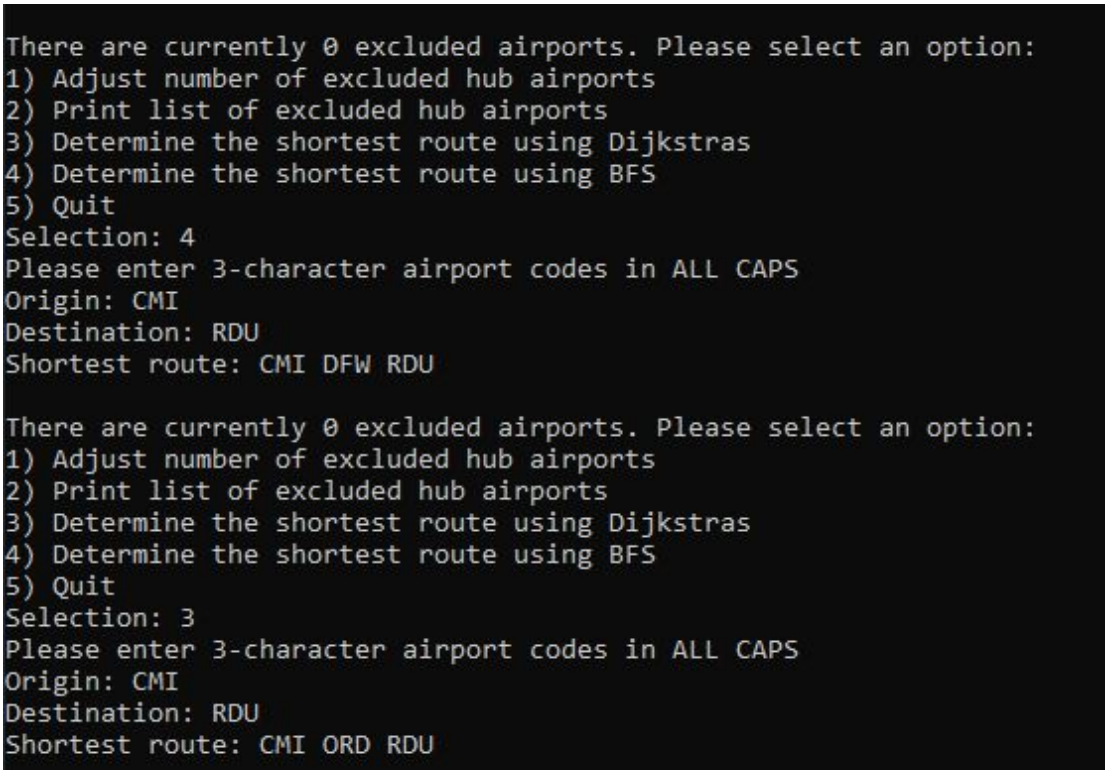
Overall our project was successful as we achieved all of our deliverables using the algorithms we set out to use. The main area we can improve on is in runtime as it can take up to a couple of minutes to calculate which airports are central hubs. But this can be solved by utilizing memoization in the Dijkstra's algorithm to save state between runs. Also we could save the hub airports from the Betweenness Centrality algorithm in a file so that we don't have to run it as often as the hub airports don't change between runs. Lastly another solution is to give the user more options to choose regions that they want to calculate on to further reduce runtime.



This image shows the top 20 'hub' airports as determined by betweenness centrality, which will be excluded when looking for the shortest path. This can be seen in the route between OOL - SFO, which is OOL - SIN - TPE - HND - SFO, when excluding the top 20 hubs. When the route between OOL - SFO is calculated without excluding the top 20 airports, as seen below, the route is OOL - SYD - APW - HNL - OGG - SFO. So you can see that when SYD is excluded since it is a hub, the whole route changes to avoid that airport, which is what we were trying to find with this project.





This image shows the difference in the routes when BFS is used compared to Dijkstra's. When using BFS going from CMI - DFW, the calculated route is CMI - DFW - RDU. However, when Dijkstra's is used, which takes into account the distances, the edge weights in the graph, the route generated is CMI - ORD - RDU. This second route makes more sense in a geographical sense and shows the benefit of Dijkstra's, since it is clearly a shorter route.

[Presentation Link]