**CS 255 Final Project Report**

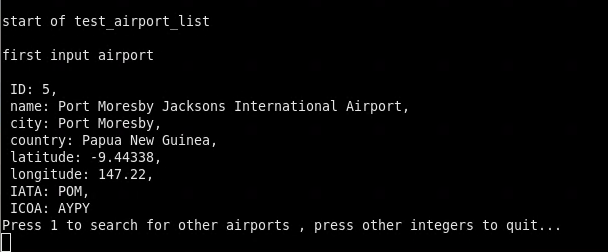
Our final project successfully implements all of the three algorithms in the proposal, including BFS, Dijkastra, and Betweenness Centrality.

First, for BFS, we use the algorithm to traverse all of the airport nodes in the graph by calling the specific function shown in the picture. The result of this function can be found in our Github repository *jibingy2-bojia2-chenxul2-sw18/output/bfs\_all\_output*. This algorithm is successful because all of the paths shown in the BFS output can be proved by calling our shortest path function that implements Dijkastra algorithm. The path can also be proved checking source airport’s *destination airport list* and destination airport’s *incoming airport list.*

Second, for Dijkastra algorithm, we use it to generate the shortest path between input source airport and destination airport. To verify our output of shortest path function. We calculate and compare the distance difference between shortest paths with one or more stop nodes and their corresponding direct distance. We find the difference between two distances are always short, no more than 1000km, for the cases we have tested. Also, in our research, we find that the tested shortest paths always have little curvature in the world map. In addition, we find that the stop nodes’ betweenness centrality is always very large and their destination airport list’ size is large as well, more than 200 airports. Thus, we consider our implementation of Dijkastra algorithm as successful.

Last, for Betweenness Centrality, our function that implements the algorithm is based on data from shortest paths. We count every airport’s occurrence number as shortest paths’ transit nodes. The one with largest occurrence number must be the node with largest Betweenness Centrality. Therefore, based on the correctness of our Dijkastra algorithm, our Betweenness Centrality algorithm is also correct.

The following is an example that shows the output of our Dijkastra algorithm and Betweenness Centrality.



The above picture shows the example tested node’s information.



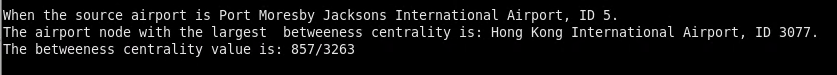
The above picture shows the destination airport list of the tested airport.

Both of the above picture’s data corresponds to our dataset.文本

描述已自动生成文本

描述已自动生成

The above two pictures show two shortest paths when source airport is tested airport.



The above picture shows the output of our Betweenness Centrality function.

*Note: All of the above pictures are generated through our final project, and the code that are used to generate these outputs can be found in main.cpp, graph.cpp, and helper.cpp.*