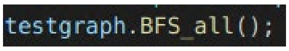
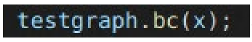
CS 255 Final Project Report

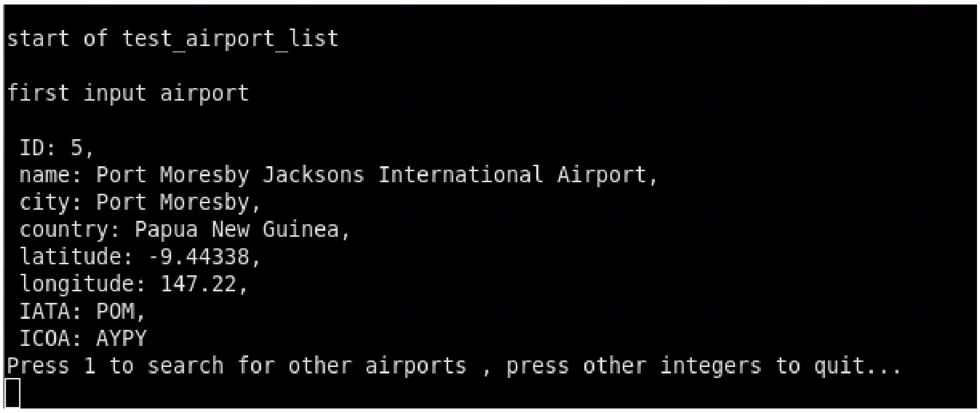
Our final project successfully implements all of the three algorithms in the proposal, including BFS, Dijkastra, and Betweenness Centrality.

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.  `testgraph.BFS_all();` 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.

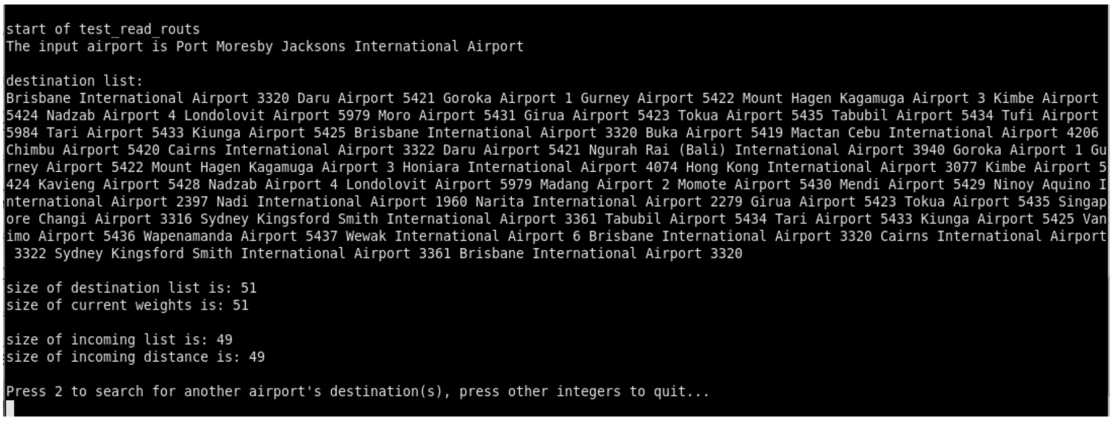
Last, for Betweenness Centrality, our function  `testgraph.bc(x);` that implements the algorithm is based on data from shortest paths. We count every airport's occurrence number as shortest paths' transit nodes. The one with largest occurrence number must be the node with largest Betweenness Centrality. 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.



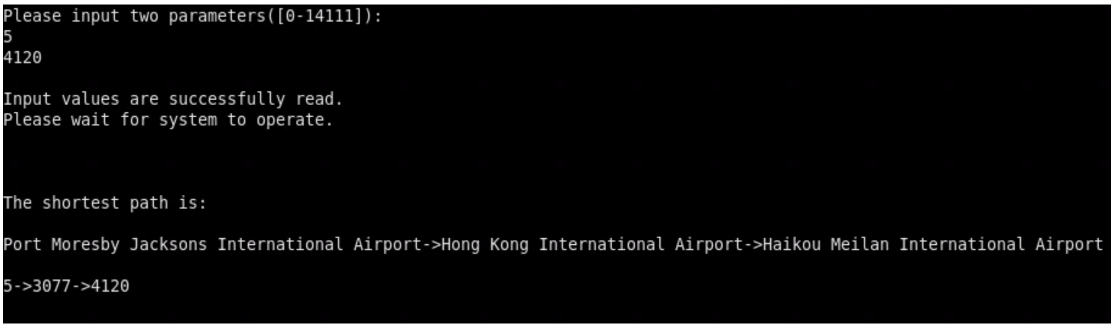
```
start of test_airport_list

first input airport

 ID: 5,
 name: Port Moresby Jacksons International Airport,
 city: Port Moresby,
 country: Papua New Guinea,
 latitude: -9.44338,
 longitude: 147.22,
 IATA: POM,
 ICOA: AYPY
Press 1 to search for other airports , press other integers to quit...
```

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

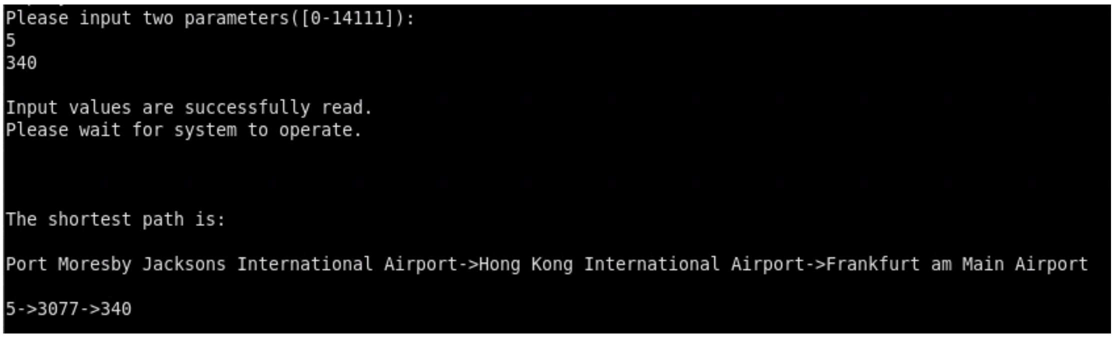


```
start of test_read_routs
The input airport is Port Moresby Jacksons International Airport

destination list:
Brisbane International Airport 3320 Daru Airport 5421 Goroka Airport 1 Gurney Airport 5422 Mount Hagen Kagamuga Airport 3 Kimbe Airport
5424 Nadzab Airport 4 Londolovit Airport 5979 Moro Airport 5431 Girua Airport 5423 Tokua Airport 5435 Tabubil Airport 5434 Tufi Airport
5984 Tari Airport 5433 Kiunga Airport 5425 Brisbane International Airport 3320 Buka Airport 5419 Mactan Cebu International Airport 4206
Chimbu Airport 5420 Cairns International Airport 3322 Daru Airport 5421 Ngurah Rai (Bali) International Airport 3940 Goroka Airport 1 Gu
rney Airport 5422 Mount Hagen Kagamuga Airport 3 Honiara International Airport 4074 Hong Kong International Airport 3077 Kimbe Airport 5
424 Kavieng Airport 5428 Nadzab Airport 4 Londolovit Airport 5979 Madang Airport 2 Momote Airport 5430 Mendi Airport 5429 Ninoy Aquino I
nternational Airport 2397 Nadi International Airport 1960 Narita International Airport 2279 Girua Airport 5423 Tokua Airport 5435 Singap
ore Changi Airport 3316 Sydney Kingsford Smith International Airport 3361 Tabubil Airport 5434 Tari Airport 5433 Kiunga Airport 5425 Van
imo Airport 5436 Wapenamanda Airport 5437 Wewak International Airport 6 Brisbane International Airport 3320 Cairns International Airport
 3322 Sydney Kingsford Smith International Airport 3361 Brisbane International Airport 3320

size of destination list is: 51
size of current weights is: 51

size of incoming list is: 49
size of incoming distance is: 49

Press 2 to search for another airport's destination(s), press other integers to quit...
```

The above picture shows the destination airport list of the tested airport. Both of the above picture's data corresponds to our dataset.

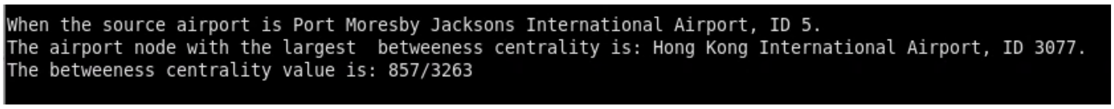


```
Please input two parameters([0-14111]):
5
4120

Input values are successfully read.
Please wait for system to operate.


The shortest path is:

Port Moresby Jacksons International Airport->Hong Kong International Airport->Haikou Meilan International Airport

5->3077->4120
```



```
Please input two parameters([0-14111]):
5
340

Input values are successfully read.
Please wait for system to operate.


The shortest path is:

Port Moresby Jacksons International Airport->Hong Kong International Airport->Frankfurt am Main Airport

5->3077->340
```

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



```
When the source airport is Port Moresby Jacksons International Airport, ID 5.
The airport node with the largest  betweeness centrality is: Hong Kong International Airport, ID 3077.
The betweeness centrality value is: 857/3263
```

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.