CS 225 Project Results

Overview

As a quick recap on what our project goals are, our project's primary aim is to identify the shortest route between two airport destinations. We created a graph that consists of airports as vertices and routes between airports as edges. Two algorithms were used to accomplish this feat—Dijkstra's algorithm when there exists a direct path between two vertices, and Landmark algorithm when there does not exist a direct path between two vertices.

Results

We tested our code with many subsets of data including large international and domestic airports—Los Angeles (LAX), Chicago O'Hare (ORD), London Heathrow (LHR), Singapore (SIN), Barcelona (BCN) as well as smaller, suburban or rural airports—Champaign (CMI), Chennai (MAA), Buenos Aires (EZE). After viewing the results of some of these tests, we have come to the mind-bending conclusion that the shortest route between large, domestic airports will be direct flights from the source to the destination, with no layovers! See figure 1 for our results for the shortest route between LAX and ORD.

Of course, the above case is simple—the shortest route between two large U.S. cities. What happens if we try to find the shortest route between a large domestic city and a large international one? We have concluded that the shortest route between a large U.S. city and a large international city typically involves one layover flight, often in another large international airport. For instance, when travelling from LAX to SIN, it takes one extra layover in Dubai International Airport (DXB) before we can arrive in Los Angeles. See figure 2 for the results for the shortest route between LAX and SIN.

Let's explore a boring case: the shortest route between a small domestic airport and a large U.S. airport. What is the shortest distance between LAX and our very own CMI? Not surprisingly, there is one small layover flight in ORD… See figure 3 for the results for the shortest route between LAX and CMI.

Now let's discuss the most exciting case—the shortest distance between a small domestic U.S. airport and a small domestic international airport. Calculating the shortest distance it takes to get from CMI to MAA, we see that it requires an infuriating three layovers to get from our source to our destination! You may want to pack an extra pack of instant coffee… Check out figure 4 for the results for the shortest route between CMI and MAA.

One limitation of the applicability of our project is that our program calculates the shortest *distance* between two locations—not the most cost effective or most practical way of travel. For instance, if we wanted to travel from CMI to SIN, one might expect to have one inconvenient layover. However, if we test this case, it turns out that the shortest distance requires us to have *two* layovers. See figure 5 for the results of the shortest route between CMI and SIN.

Say, for instance, Professor Evans decided to award our group with a free trip around the world to as many cities as we wanted as long as we can provide him with the shortest route between all the destinations. No problem! We can use the Landmark algorithm to find the shortest path between all of our dream destinations. Let's say we start in Chicago and we want to visit Los Angeles, London, Barcelona, Dubai, and Singapore. First, we set up our source (landmark) and destination vectors (Figure 6). After examining our output (Figure 7), we can conclude that all of the cities are large enough such that there are direct flights to each of the cities. Wonderful.

Appendix

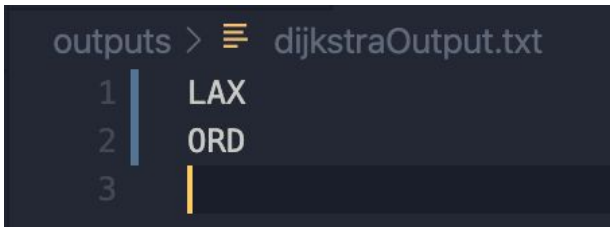

Figure 1—Shortest route between LAX and ORD.

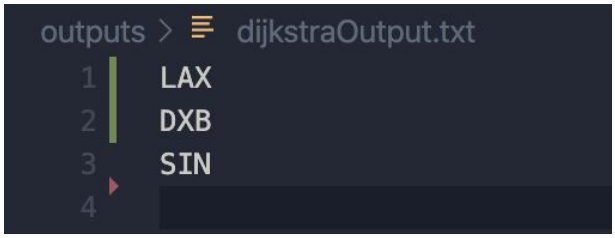

Figure 2—Shortest route between LAX and SIN.

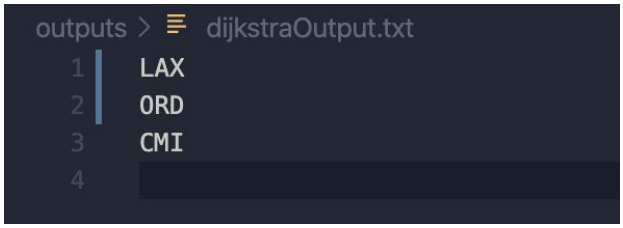

Figure 3—Shortest route between LAX and CMI.

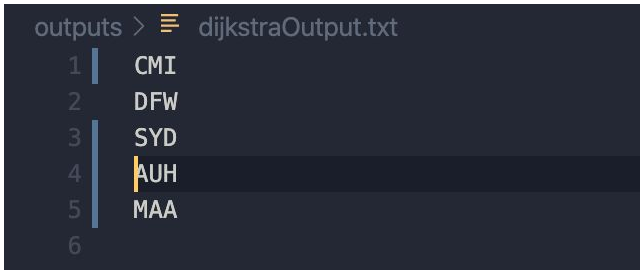

Figure 4—Shortest distance between CMI and MAA.

Figure 5—Shortest distance between CMI and SIN.

```
42    write.open("outputs/landmarkOutput.txt");
43    vector<Vertex> destinations;
44    destinations.push_back("LAX");
45    destinations.push_back("LHR");
46    destinations.push_back("BCN");
47    destinations.push_back("DXB");
48    destinations.push_back("SIN");
49
50    vector<Vertex> landmark = Landmark(g,"ORD", destinations);
```

Figure 6—Setting our source and destinations

outputs > landmarkOutput.txt

```
1  ORD
2  LAX
3  LAX
4  LHR
5  LHR
6  BCN
7  BCN
8  DXB
9  DXB
10 SIN
11
```

Figure 7—Our class trip around the world.