CS 225 Final Project Results: Finding Shortest Paths Between Airports Using Graphs

Alex Youtsey, Renzo Ledesma, Katie Sanders, Justin Kang

Our final project uses an open source dataset, OpenFlights, to create a graph of airports and flight routes around the world. Using this graph, we implemented three algorithms to explore the dataset: Breadth First Search (BFS) traversal, Dijkstra's algorithm, and A* search algorithm. A point of interest is that Dijkstra's and A* search are both shortest path algorithms with different implementations, meaning we can compare the runtime of both algorithms when using the same input parameters.

To load the dataset into a graph, we created a FileReader class that reads the list of airports and list of routes to create a directed, weighted graph to be used by all our algorithm implementations. Some of the challenges we faced in parsing our data included data points with incomplete information, such as routes having no source or destination, and data points using commas within the .csv format that caused issues when using a comma as a delimiter. Because each airport holds additional information such as its name, latitude, and longitude, we also refactored the Vertex, originally a typedef of the std::string class, to be a struct that can contain this additional information. This change allows us to output the names of each airport and use the latitude and longitude coordinates for additional calculations.

In our implementation of the Breadth First Search traversal of the graph, we created a function that returns a vector of the airports with a minimum number of outgoing routes. This number of outgoing routes is determined by the parameter entered during the function call. Since some of the routes from the OpenFlights dataset are formatted with invalid parameters, not all airports have outgoing routes listed. In order to return a vector of the full BFS traversal of the graph, 0 can be entered as the minimum number of outgoing routes. Interestingly, when running the BFS traversal with a parameter of 100, only 25 of the 28 airports returned by the traversal appear in the list of 50 busiest airports worldwide by passenger traffic in 2019, according to the Port Authority of New York and New Jersey. We predicted that all of the airports returned by the BFS traversal with a parameter of 100 would have appeared in the top 50 for passenger traffic. The three airports that did not appear in the list were Vienna International Airport, Domodedovo International Airport, and Brussels Airport. Perhaps these airports have many outgoing flight routes but with low passenger totals, causing a

high ranking in our BFS traversal but no appearance in the top 50 airports by passenger traffic.

In our priority queue implementation of Dijkstra's algorithm, we created a function that returns a vector of vertices containing the shortest path from a source airport vertex to a destination airport vertex. This path vector contains each of the airports one would visit on the shortest route from source to destination. If no flight/connections exist from one airport to another, this function returns an empty path vector. The edge weight used by our Dijkstra's implementation is the throughput of the edge, defined as 1/n where n is the number of routes along the edge.

Dijkstra's algorithm and A* search algorithm both find the shortest path (represented by a vector<Vertex>) between two airports (each represented by a Vertex). However, A* is optimized to find a single best path by using a heuristic function to further inform the algorithm of the best path. This heuristic function is not strictly defined in the description of the algorithm. Rather, it is situational depending on the kind of data you are performing the search on. In our case, the heuristic function returns the physical distance between two airports in kilometers, i.e. the shortest distance between two points (airports) on a sphere (Earth). This calculation requires the latitude and longitude coordinates of each airport, making use of additional information in the dataset granted by refactoring the Vertex. Ideally, this heuristic function informs A* to prefer the path with the shortest distance when dealing with two or more equally weighted paths.

In conclusion, our graph model of airports and flight paths between them allowed us to make some interesting discoveries and comparisons. For instance, the BFS traversal of the graph's airports with at least 100 outgoing routes returned results somewhat contrary to our initial prediction. It seemed clear that all of the returned airports would be large airports with domestic and international flights carrying high numbers of passengers. As it turns out, while there appears to be a correlation between the number of outgoing routes and passenger traffic, it is not a perfect correlation. However, many of the airports with at least 100 outgoing routes, such as Hartsfield Jackson Atlanta International Airport and Beijing Capital International Airport, appear in many shortest paths; this finding makes sense as these airports would commonly serve as connections or layovers between other flights.

References

"2019 Annual Air Traffic Report." Port Authority of New York and New Jersey, 2019. www.panynj.gov/airports/en/statistics-general-info.html. Accessed December 10, 2020.

Wikipedia contributors. "Dijkstra's Algorithm." Wikipedia, 7 Dec. 2020, en.wikipedia.org/wiki/Dijkstra%27s_algorithm. Accessed December 10, 2020.

Wikipedia contributors. "A* search algorithm". Wikipedia, December 2, 2020. https://en.wikipedia.org/wiki/A*_search_algorithm. Accessed December 10, 2020.