**Shiyak: A Flight Mapping Application**

Jack Dunbar, jdunbar

My project will solve the problem of finding the cheapest flight from point A to point B, which has plagued travelers for decades. It will allow the user to specify the starting airport and destination airport (from large and medium hub airports in the United States). The program presents the cheapest flight given a maximum number of stops on a map to the user.

How I will solve this problem:

* **The GUI**

The primary portion of the user experience will be a map of the United States and airports in the United States. The user will specify the airport he or she wishes to depart from and go to as well as the maximum acceptable number of stops using a dialogue box in the upper-left corner of the screen. Another selection option the user will have is to click on the airport on interest on the map. The user will be able to move around the map by clicking and dragging as well as using arrow keys and zoom by using the designated zoom buttons and by using the “+” and “-“ keys.

If the user has trouble utilizing any of these features, he or she can select the “help” option from the upper-left dialogue box and be presented will a full tutorial on how to use the app.

* **The Flight Data**

One of the paramount tasks in this project will be the acquisition of flight cost data. This will come from the Bureau of Transportation Statistics, and will be average flight data for the year 2017. I will read this data in from a csv file and extract the necessary cost data from flights between various airports.

* **Finding the cheapest route**

I will find the cheapest route for flights between various airports by creating a graph with nodes that represent the airports and edges that represent the cost of flying between those airports (from the csv file). I will then use Dijkstra’s algorithm to minimize the cost of flying between two airports (nodes).

A recursive algorithm for Dijkstra’s:

I will represent the graph of airports and their connections as a dictionary with keys of various airports and values of dictionaries with the airports the key is connected with mapped to the cost of flying between the original airport and that airport. For example:

airportMap = {“LA”: {“NY”: 500, …}, …}

Then, I will use Dijkstra’s to minimize cost between flying from a start airport to a finish airport. My solution uses recursion to go through all possible paths to the finish, updating a list that represents the minimum cost way of going there (it doesn’t return anything and relies on list aliasing). The solution will look like this:

def Dijkstras(map, start, finish, nodeWeights, minimumCostPath, visitedList):

for possible flights from start:

add to visited

if pathCost < currentAirportCost:

update node weight

if at end and less than max stops:

update minCostList

otherwise:

apply Dijkstra’s with current node as the starting node

remove from visited

remove start from visited (for recursive calls)

Update 1:

I’m going to add the following:

* A swap button that switches the departure and arrival airports
* Webscraping faredetective.com to get my cost-of-flight between airports data
* Zooming on areas of the United States with many large airports
* The addition of airports with 500,000 to 1,000,000 departures
* An additional map for the user to choose between

I’m going to remove the following features:

* Continuous zoom
* A maximum number of stops selector

Update 2:

I successfully added a swap airports button and an option to zoom-in on the northeast to Shiyak. I also successfully scraped data from the web to populate my connections between airports. I then stored these connections as a dictionary in a .txt file in the project folder, so web-scraping doesn’t have to happen every time the user runs the program, saving the user time. I decided not to implement an additional map because it would mess-up all the pixel locations for my project. I also decided not to add all the airports with between 500,000 and 1,000,000 departures because that would take too much time to run.