Before coding plan:

* We will build a graph with a method that reads the data provided from the CSV file and make an adjacency matrix from it. The graph will show the connection between the stations and train lines. If the line exists in the station, it will contain a “1”, if the line doesn’t exist on the station, it will contain a”0” The columns will contain the station’s name, while the row will contain the train lines.
* The “fastest route” will be defined as how to get from destination A to destination B with the least number of transfers required.
* We will model the transfer by having the user go to the closest station from their starting point, and transfer in a station with a train line that requires the least number of transfers. This affects our fastest route since our goal is to have the least number of transfers required.
* Our program will implement Dijkstra’s algorithm to find the shortest route.
* We will be using a Hash Map for the adjacency matrix.
* The program will print to the user the number of trains or trains they need to take to reach their destination from their starting station.

Changes made during coding:

* Instead of using an adjacency matrix to represent the tree of the subway station and subway lines we chose to use an adjacency list.
* Dijkstra’s algorithm was also replaced by the Breadth first search algorithm to find a path amongst the two stations.
* We got rid of the program finding a path by using the least number of transfers.

Analyzing the program:

SubwaySytem:

* This method reads the CSV file provided and creates a Hash Map which then stores the subway connections and stations.
* The data taken from the CSV Files are, stationID, stationName, longitude, latitude, and the subwayLines in the stations.
* A while loop is used to read line by line of the CSV file and store the information in the Hash Map subwayConnections of each subway station until there are no more lines to read.
* The subway lines of each station are stored in the Hash Set: lines. The for loop iterates over each subway line in the subwayLines array and adds it to the lines Hash Set.
* The final for loop checks over each station in subwayConnections Hash Map and compres the subway lines to every other station’s subway lines. If two stations share a common line it creates an edge and adds it to the adjacencyList HashMap. The method addStation is called to add the edge.

addStation:

* The purpose of this method is to add the subway stations into a tree and link the stations with the common subway lines. The tree will be represented with an adjacency list.
* Firstly, the method checks to see if stationNames has edges already or not by using the getOrDeafult method. If the stationName doesn’t contain edges, then an empty list is created to hold the info for edges.
* Then a new edge object is created for stationName, satationName2 and subwayLine as its parameters.
* Then the edges are stored back in the adjacencyList data structures using the “put” method.
* This method works together with the SubwaySystem method in order to create the graph of the stations from the CSV file.

Bfs:

* The purpose of this method is to use the Breadth First Search algorithm to find a path from starting station to the ending station.
* This method takes in the two parameters startStation and endStation.
* Prev is used to represent the previous stations in the path, subnwayLineAtPrevStation will be used to keep track of the line that is used to get to prev and visited keeps track of the visited stations.
* A queue is initialized with the startStation and an Array list with line which are used to keep track of the subwaylines used to get from station to station.
* The while loop searches for the path from startStation to endStation. The stations visited will be added to visited, for each neighboring station that has not been visited, it adds the neighboring station to the queue and updates prev and subwayLineAtPrevStation to contain the current station as the previous station and the line used to get to the neighboring station.
* Once the path has been found, the method confirms the path beginning at the endStation and following back to the prevs then back to the startStation. Each station is added to the beginning of the path list until the startStation is reached.
* If a transfer is required between subway lines, it adds the transfer station(s) and the subway line(s) requires for the path.

Overall, this program used Breadth First Search algorithm to find a path between the two stations. We also used an adjacency list to represent the tree of the subway stations and subway lines. The goal of the program is to find a path from the start station to end station. If a transfer is required, the program adds the current station name and the subway line at the previous station to the HashSet “line”. It then increments the index i by one to skip the next station, since it is a transfer station. This continues until the end station is reached. The user should run the program and enter their starting station and end station for the program to execute.