Zara Zetlin

**Project Description:**

Name of the TP: Globetrotter.

Globetrotter displays a map of 15,000 cities throughout the world. Users can make an account and plan trips by typing in the cities they want to visit. Using the simplex algorithm, the shortest path between these cities will be calculated and the suggested route between the cities will be displayed. Users can zoom in, and recommendations of cities to add to certain trips will be generated.

**Competitive Analysis:**

Sygic Travel Maps app displays attractions, restaurants, shops, etc. on a map and allows users to explore them and add a location to their trip. Users can plan their day-by-day trip in the app and see estimated times between locations and distances. This is similar to my project as users will be selecting locations to visit and recommendations will be provided. Instead of using the user’s order of input to decide their route, my project will find the shortest route for them. Additionally, Sygic uses distances with respect to road infrastructure, my project uses absolute shortest distances (straight lines).

Route4Me Route Planner app is a paid service that allows users to input destinations and find the shortest route with respect to roads between the destinations. It is targeted for companies that drive to many locations, so it displays the total driving time and the location of the driver along the road. Road Warrior Route Planner acts similar to Route4Me Route Planner app. It has the additional feature of inputting scheduling windows, which allows users to arrive at a specified stop within a time frame. These two apps are similar to my project as users will be selecting locations to visit and an optimal route will be generated. However, Route4Me and Road Warrior uses distances with respect to road infrastructure, my project uses absolute shortest distances (straight lines). Additionally, they’re both targeted at companies and drivers, while my project is aimed at individuals and not specified for a mode of transportation.

**Structural Plan:**

Here is a list of the functions I plan on having in my project:

appStarted(*app*)

redrawAll(*app*, *canvas*)

drawAllCities(*app*, *canvas*)

playMaps()

getXYCoords(*app*, *city*, *lat*, *lng*)

drawCity(*app*, *canvas*, *city*, *lat*, *lng*, *fill*)

drawDot(*canvas*, *x*, *y*, *fill*)

getDist(*app*, *xCoord1*, *yCoord1*, *xCoord2*, *yCoord2*)

makeTripDict(*app*)

makeDictOfEdgeVars(*app*)

formulateAsIP(*app*)

makeIntoSEF(*app*)

constructAuxilaryProb(*app*)

firstCanonicalForm(*app*)

checkCNLessThan0(*app*,*xVarMatrix*, *newCMatrix*, *aMatrix*, *bMatrix*, *basis*)

pivot(*app*,*xVarMatrix*, *newCMatrix*, *aMatrix*, *bMatrix*, *basis*)

generalCanonicalForm(*app*,*xVarMatrix*, *newCMatrix*, *aMatrix*, *bMatrix*, *basis*, *leavingVar*, *enteringVar*, *indexOfEnteringVarInBasis*)

simplexSolution(*app*,*count*, *xVarMatrix*, *newCMatrix*, *aMatrix*, *bMatrix*, *basis*)

confirmSoln(*app*,*count*, *xVarMatrix*, *newCMatrix*, *aMatrix*, *bMatrix*, *basis*)

simplexAlgorithm(*app*)

dictOfTripDetails(*app*, *xVarMatrix*, *newCMatrix*, *aMatrix*, *bMatrix*, *basis*)

drawTrip(*app*, *canvas*)

zoomIn(*app, canvas*)

zoomOut(*app, canvas*)

definingRecommendations(*app*)

findingRecommendations(*app*)

drawRecommendations(*app*)

keyPressed(*app*, *canvas*)

drawScore(*app*, *canvas*)

main()

**Algorithmic Plan:**

The most complex parts of my project are to solve the trip’s integer program (IP) using the simplex algorithm, zooming in/out, and recommending cities to add to a given trip.

Using the simplex algorithm to solve the shortest route between the cities of trip has variables for every edge in the Hamilton cycle of the trip. There is an objective function to minimize the distance between these initialized edges. There is a constraint for each city that says there can be two edges connected to that city. The simplex algorithm solves this by solving the auxiliary problem: using pivots to change the basis and then rewriting the IP in canonical form. Bland’s rule is used for the pivots to ensure the algorithm terminates.

For zooming in/out, I plan on having two buttons (a plus/minus sign) that when clicked will update the formulation for integrating latitudes/longitudes on to the canvas. The latitude/longitude will still be centered (with value (0,0)) at the center of the canvas (at app.width//2, app.height//2), but the latitudes won’t go from -90 to 90 and the longitudes won’t go from -180 to 180 but will be decreased/increased respectively.

For recommending destinations once the shortest trip has been found, I will locate cities within a certain distance from the edges of the trip. The few cities falling closest to the edges of the trip will be generated as recommendations.

**Timeline Plan:**

For checkpoint 5 (this Monday) I plan on having trips display correctly and hopefully having started zooming in/out.

For TP2 I plan on having zoom in/out and maybe starting on recommendations.

For checkpoint 6 I plan on having recommendations generated.

Then for TP3 I will have the whole project completed and presentable where I can add some additional features.

**(continued on next page)**

**Version Control Plan:**

I am using google docs to save my code in my google drive.

A screenshot of a cell phone

Description automatically generated

**Module List:**

I am not planning on using any additional modules.

**TP2 Update:**

I’ve added new functions to the structural plan:

*def* drawLogin(*app*,*canvas*):

*def* usernameEntry(*app*,*canvas*):

*def* passwordEntry(*app*,*canvas*):

*def* continuePressed(*app*,*canvas*):

*def* makeDictOfUsernamesNPasswords(*app*):

*def* saveUserData(*app*):

*def* drawPlanTrip(*app*,*canvas*):

*def* continuePressedInPlannedTrip(*app*,*canvas*):

*def* tripNameEntry(*app*,*canvas*):

*def* tripEntry(*app*,*canvas*):

*def* yesPressedInPlannedTrip(*app*,*canvas*):

*def* noPressedInPlannedTrip(*app*,*canvas*):

*def* drawLogout(*app*,*canvas*):

*def* drawTripsVisited(*app*,*canvas*):

*def* drawTripsPlanned(*app*,*canvas*):

*def* dictOfCities(*app*):

*def* drawLine(*app*,*dTrip*,*dXVars*):

*def* drawButtons(*app*,*canvas*):

*def* mousePressed(*app*,*event*):

*def* timerFired(*app*):

I will not be using two functions I had planned on in TP1:

zoomIn(*app, canvas*)

zoomOut(*app, canvas*)

**TP3 Update:**

I will not be using the functions :

*def* yesPressedInPlannedTrip(*app*,*canvas*):

*def* noPressedInPlannedTrip(*app*,*canvas*):