CSSE230 – Final Project Write Up

5guys

Technical Features that we implemented:

For this project we have implemented the required features as well as a few extra features that we felt were necessary to improve the quality of our final submission. Through division of labor we have implemented a map with multiple cities displayed – each with various points of interest such as national monuments, museums and gardens. Each of these cities has its very own interest rating and they each have connections to other cities. While some cities have more connections (roads) to other cities, all cities are connected to one another in some way or another. We used hashmaps to store neighboring cities in the node representing any given city. The node also holds the interest rating of the city as an integer and holds an arraylist of landmarks present in the city. The GUI displays these landmarks when you click on a given city – although you must click out of the map after clicking any particular node to refresh the screen. Hence you can view the interest level of any given city. Cities are also easy to locate since their names are displayed on the map interface.

The system that we have designed is also capable of finding the shortest route between any 2 cities. It is able perform this task as each path between cities has an implicit distance rating that is passed in through a parsed document. This is to show that even if the linear distance on the map between 2 cities seems short, the distance may actually be quite long due to winding roads and mountainous regions. The user can also search for routes by both distance and interest cost. Since each city has its own interest level, users can plan their journey to take the most “interesting” route between their start and destination. Users can also view the interest rating on every city preset on the map since each city has its own interest level and landmarks present within itself. The system is able to find these paths through the use of the A\* search algorithm. It stores each of the routes as nodes in the priority queue and then sorts them to obtain the optimal route or the most desired route. We found that using the A\* algorithm was optimal here as it allowed us to find the route successfully. The reason the optimal distance and most interesting routes are the same is due to the small amount of cities we have in our map of India.

The GUI that we designed is also great since it goes beyond just seeing city nodes and paths on the map. The GUI we have designed overlays these cities on a map of India and each city is actually located in the approximate geographical location it can be found on a real map of India. The GUI is also colorful and displays the distance between cities as well as the names of cities on the map itself. The side panel allows for a variety of searches and also provides users with optimal routes and the routes that users themselves are searching by.

Our mapping system is also capable of scaling up quite well since all information that is passed into the system is stored in well formatted text documents. Hence for any additional cities, their details are simply stored in these text documents. We believe that this is a great feature since the system is infinitely scalable and the actual code in the system will never be disturbed. Methods in the system have also been split up to follow strong object oriented design. Although we have left some of our hashmaps with default modifiers since they are used all over the place and using setters became something of a nuisance. This data is never really modified outside a single class so we believed it was a good idea to do so.

There is also only a single method in reader is also presented as one huge method since it parses multiple text documents. We felt it was necessary to do this so that in any other given class multiple methods would not need to be called – which could lead to confusion. This class is only used when loading the data into the system and hence this method only needs to be called on the reader object a single time to preload all the data.

Our system also made use of some extra features not listed in the project requirements. All the data that was loaded into the system was read from text documents and then put into objects present in the system. The GUI was also beyond simple since it offered a cool map to present the data on rather than a boring background with a few nodes on it.