COP3530

Project 3

Group 68

**Project 3 Report: Shortest Path Between Cities Road Trip Planner**

**Administrative**

* Team Name
* Metaverse
* Team Member
* Lawson Hutter, Ryan Hudson, Yang Zheng
* GitHub URL
* <https://github.com/ryanh6900/FinalProjectHoustonHutterZheng>
* Link to Video
* <https://www.dropbox.com/sh/qp8ixoa3x9ysxtp/AABGjfywTdAjJnVYIAaiYFYoa?dl=0>

**Extended and Refined Proposal**

* Problem: What problem are we trying to solve?
* In this project, we try to find the shortest path to the destination with stops between adjacent cities. We also will create a UI page that allows the user to explore the shortest path between cities by typing in two cities’ names.
* Motivation: Why is this a problem?
* As for our personal experience, it is hard to make a road trip plan by simply looking over the map. It is also impossible to plan out a perfect road trip schedule without extensive research. Especially if you don’t want to miss out on the cities and stops along the road towards your destination. Our program is designed to help people plan out their trip by simply typing in the names of two cities, and it will show you the suggested shortest path and stops between cities.
* Features implemented
* A UI page will be generated that allows the user to put in two cities and then displays the output.
* Description of data
* Using some functions and data from <https://github.com/pr4296/US-Map>, I read through a list of over 25,000 cities. Using this data set we pulled the longitude and latitude of each city and calculated the distance between cities, not including elevation, with the haversine formula.
* The data took over 20 minutes to process so we generated the data in advance and stored an edge list to a CSV file. To make the data set easier to use, we store several city edges within 30km, 100km, 720km. At most, within the edge list of a given city is about the distance of a full tank of gas. Additionally we put a cap of 1700 on the population to disregard smaller cities.
* The parameters had to be fine tuned for hours to ensure the map is connected.
* Once the map was connected, the final data set is stored in the format of: citynameA, statenameA, citynameB, statenameB, distance
* The final edge list contains over 700k edges and 12k vertices.
* Tools/Languages/APIs/Libraries used
* C++
* <https://github.com/pr4296/US-Map>
* SFML
* Graph Algorithms implemented
* **Graph:** The graph is implemented with an unordered map for time complexity and also to allow iteration by the name of a city. The key of the map is a city name followed by a state initial. The value of the map is a list of edge structs that contain the edge data.
* **Dijkstra's Algorithm:** Several iterations of this code were used to try and decrease the time the algorithm took to run. The final version implements a min heap via a priority queue. This improved the time needed to find the next unvisited minimum weight vertex.
* **Bellman Ford Algorithm:** This algorithm was very straightforward because we did not require the portion of the algorithm for negative weight. This algorithm worked very well upon the first run.
* Both algorithms take in a reference to a vector of strings and populates them with the shortest pathway of vertices from point A to point B.
* Additional Data Structures/Algorithms used
* Unordered\_map for graph
* Unordered\_set, map, linked list, and more for refining data. Some implemented by the modified code from the github project and some implemented by our team.
* Priority queue for dijkstras.
* Graph object to contain the entire graph.
* Edge struct to contain data of to, from, and weight for each edge.
* City object to contain name, state, and other data of each state.
* Distribution of Responsibility and Roles: Who did what?
* Lawson Hutter: Read and process all the datas and code the shortest path functions
* Ryan Hudson: Design the UI page and make the video
* Yang Zheng: Report

**Analysis**

* Any changes the group made after the proposal? The rationale behind the changes.
* We changed our proposal into a new topic after the first draft of the proposal. Originally, we wanted to write a project that can create recommendations for accounts based on users’ interests. This algorithm has already been implemented by different social media platforms to make higher user engagement and activities. However, when we tried to create a similar algorithm, we found it was difficult to only utilize the graph algorithm to implement our thoughts. Therefore, we tried to discover more topics that can use shortest path algorithms. Inspired by the RTS bus route map, we came out with the idea of finding the shortest path between cities.
* Complexity analysis of the major functions/features you implemented in terms of Big O for the worst case
* BellmanFord time complexity = O(VE). Bellman ford accesses the list of each vertex and the vertex itself. Hence V\*E.
* Dijkstras time complexity with priority queue = O(V + E)logV. Each edge and each vertex is analyed but each pass along the list omits vertices that have already been visited.
* However, our results did not reflect a massive change in time for either algorithm. Additionally, the two algorithms result in very slightly different solutions. We read online that this is possible if there is not one unique solution. The graph has thousands of different pathways that are potentially optimal and because of this we think the performances of the algorithms were not ideal. However, Dijkstras algorithm should have been faster.
* My dijkstras algorithm could have been more efficient. There may be issues with how it handles visited vertices. The issues with this stemmed from the methods I used for the graph structure.

**Reflection**

* As a group, how was the overall experience for the project?
* The project is an innovative way to implement learned knowledge into a real-world problem. Without the specific requirements and test cases, this project allows us to think outside of the box.
* Did you have any challenges? If so, describe.
* At the beginning of this project, brainstorming the idea that we can implement is the first challenge. Unlike the past projects we’ve done, creating our project needs us to comprehend the current knowledge fully while being creative. The second challenge is to figure out the logistics of the project. After we set our first proposal, we tried to figure out the best way to work on it, but we found out that our idea needs a more advanced level of data structure that hasn’t been covered in this class.
* The algorithms might have been able to run more effectively and faster if the graph was implemented differently. There is a really big tradeoff in what is efficient in the data structure vs what is efficient in the algorithm. It was very difficult determining what was the right strategy for the assignment.
* If you were to start once again as a group, any changes you would make to the project and/or workflow?
* If we can start again as a group, we should start as early as we can so it will allow us to have enough time to try out different ideas to decide which is the best fit for our group project. On top of that, as a group, we need to work on our time management skills. In order to keep everyone on the same page, assigned tasks with multiple deadlines can be set to track the progress of each group member. The distribution of workloads should be more evenly so that every group member can have hands-on experience on different aspects of the project.
* Comment on what each of the members learned through this process.
* Communication skill is the crucial element of teamwork. I need to be more responsive and make time to work on the project instead of working on it within a very limited time frame. — Yang Zheng
* I learned more about the tradeoff between data structure design vs algorithm design. It appeared that there were tons of different ways to implement one over the other and I currently did not know enough to find the most efficient method. — Lawson Hutter
* I learned how to work as a team and about graphs. I learned a lot about the SFML library. The library took way more work than other API’s that I know. Got more experience working on a programming project as a group.

— Ryan Houston