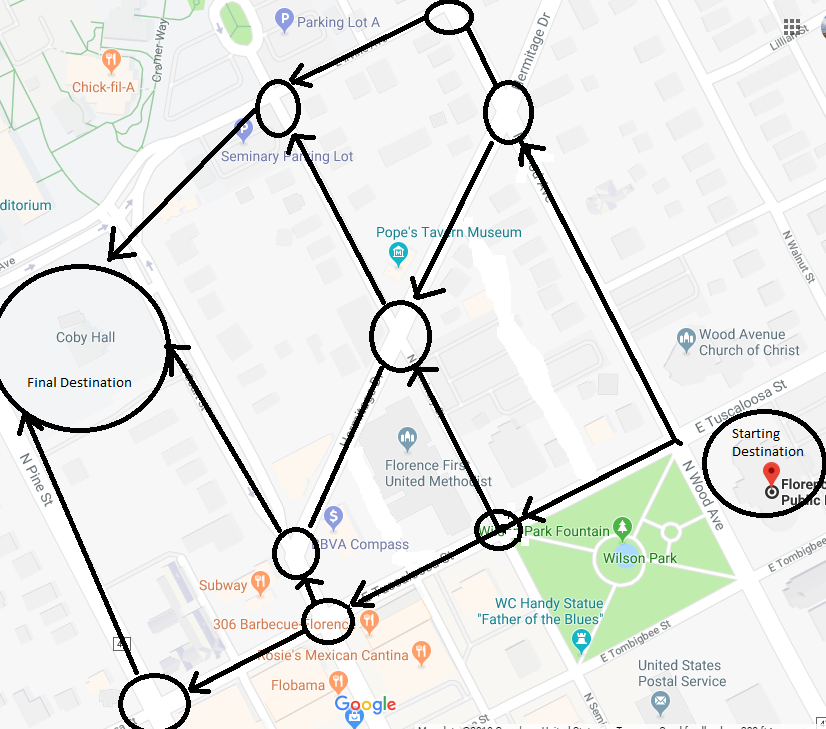
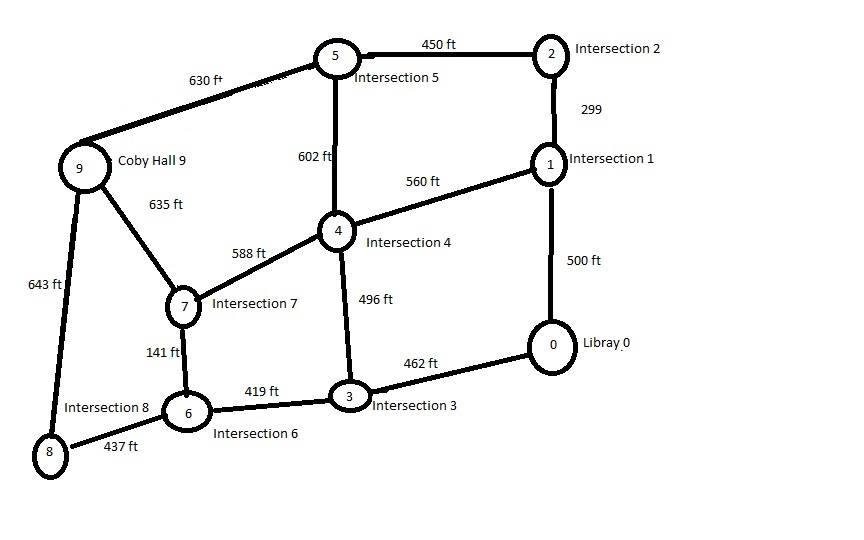
**Initial Idea Description:**

The initial idea for this program is to find the shortest path from Florence-Lauderdale Public Library to Coby Hall. There are 8 road intersections on the route from the Library to Coby Hall. There are 10 vertices that are in the graph/map. These vertices represent all the intersections, the public library, and the Coby hall in the real world. The edges represent the roads between the intersections and the two buildings. The weight of the edges will be measured by feet. The program will find the shortest path from the public library to every intersection and to Colby Hall. The data structure is a graph that is represented by the adjacency matrix. 

Mock Graph:



**Functional Requirements:**

To calculate the minimum distance and keep track of the vertices that have been visited:

int minDistance(int\* distance, bool\* visited)

To display the distance/edges cost/weight from the source:

void Display(int\* distance, int n)

To calculate and determine the shortest path in the graph:

void ShortestPath(int graph[MAX][MAX], int src)

**Non-Functional Requirements:**

Readability

Documentation

Usability

Reliability

**Mock-Up of the User Interface:**

Path Shortest Distance from Source in Feet

Library 0

Library to Intersection 1 500

Library to Intersection 2 799

Library to Intersection 3 462

Library to Intersection 4 958

Library to Intersection 5 1249

Library to Intersection 6 881

Library to Intersection 7 1022

Library to Intersection 8 1318

Library to Colby Hall 1657

**Back End Data Structures Description:**

The map has a dense graph which will be represented using adjacency matrix representation, graph[MAX][MAX].

**Inputs & Outputs Descriptions:**

The output will display the total distance or weight in integers of the shortest path from the source (public library) to all vertices (Intersections and Coby Hall).

**Detailed, Predicted Team Schedule:**

|  |  |
| --- | --- |
| **Date & Time** | **Detailed** |
| April 2, 2019/2:00 PM | Real-World Proposal |
| April 3, 2019/3:00 PM | Design Document |
| April 4, 2019/2:00 PM | Source Code Implementation/Design Documentation Update |
| April 5, 2019/12:00 PM | Source Code Implementation/ Design Documentation Update |
| April 8, 2019/3:00 PM | Source Code Implementation/ Design Documentation Update |
| April 9, 2019/2:00 PM | Source Code Implementation/ Design Documentation Update |
| April 10, 2019/3:00 PM | Source Code Implementation/ Design Documentation Update/Presentation Materials |
| April 11, 2019/2:00 PM | Source Code Implementation/ Design Documentation Update/Presentation Materials |
| April 12, 2019/12:00 PM | Presentation Materials/Portfolio |
| April 15, 2019/3:00 PM | Presentation Materials/Final Portfolio |

**Test Plan:**

Initialize all the vertices distance to infinite.

Initialize all visited[i] to false, except the source.

Create a 2D matrix, graph[MAX][MAX]:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **0** | 0 | 500 |  | 462 |  |  |  |  |  |  |
| **1** |  | 0 | 299 |  | 560 |  |  |  |  |  |
| **2** |  |  | 0 |  |  | 450 |  |  |  |  |
| **3** |  |  |  | 0 | 496 |  | 419 |  |  |  |
| **4** |  |  |  |  | 0 | 602 |  | 588 |  |  |
| **5** |  |  |  |  |  | 0 |  |  |  | 630 |
| **6** |  |  |  |  |  |  | 0 | 141 | 437 |  |
| **7** |  |  |  |  |  |  |  | 0 |  | 635 |
| **8** |  |  |  |  |  |  |  |  | 0 | 643 |
| **9** |  |  |  |  |  |  |  |  |  | 0 |

Result:

Path Shortest Distance from Source in Feet

Library 0

Library to Intersection 1 500

Library to Intersection 2 799

Library to Intersection 3 462

Library to Intersection 4 958

Library to Intersection 5 1249

Library to Intersection 6 881

Library to Intersection 7 1022

Library to Intersection 8 1318

Library to Colby Hall 1657

**Algorithm/Solution Overview:**

* Set all vertices distances = infinity except for the source vertex, set the source distance = 0.
* Push the source vertex in a min-priority queue in the form (distance, vertex), as the comparison in the min-priority queue will be according to vertices distances.
* Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source).
* Update the distances of the connected vertices to the popped vertex in case of "current vertex distance + edge weight < next vertex distance", then push the vertex  
  with the new distance to the priority queue.
* If the popped vertex is visited before, just continue without using it.
* Apply the same algorithm again until the priority queue is empty.