Brendan Jang
CS 325
Homework 5

1. a) I believe that the best algorithm to find the fastest route from the fire station to each of the intersections is Dijkstra's algorithm. Placing the fire station at G, the shortest path from G to all the other towns is shown below.

Vertex	Distance	Parent of Vertex	Shortest Path
G	0	Start	G
A	12	C	G -> E -> D -> C -> A
B	6	H	G -> H -> B
C	8	D	G -> E -> D -> C
D	5	E	G -> E -> D
E	2	G	G -> E
F	8	G	G -> F
H	3	G	G -> H

b) In order to find an optimal solution, the location for the fire station must be selected so that it minimizes the distance to the farthest intersection. An algorithm for the optimal solution is below.

```
findOptimal(city[][], f)
        // input graph of city, f number of towns or possible locations
        // return optimal location
        // Initialize optimalLocation to -1
        optimalLocation = -1
        Intialize all the distances
        sum = 0
        // find optimal location
        For I = 0 to f -1
                // initialize the distances to 0;
                tempSum = 0
                // use Dijkstra algorithm using source vertex as I and store in distances
                distances[] = Dijkstra(city[][], f, i)
                // number of roads
                roads = length(distances)
                // find sum of all the roads
                For j = 0 to roads -1
                        tempSum = tempSum + distances[j]
                if sum > tempSum
                        sum = tempSum
                        optimalLocation = i
        return optimalLocation
```

The time complexity for this algorithm is as follows. If we run the algorithm f times for f possible with r locations with r roads, is f * Θ(r + f(log f)). Which would equal Θ(f*r + f^2(log f)).

c) The most optimal location to start a fire station is D. The cost of spanning the graph is 20. Using the table below, we can see the shortest path from D to all other towns on the graph.

Vertex	Distance	Parent of Vertex	Shortest Path
D	0	Start	D
A	7	C	D -> C -> A
B	11	H	D -> E -> G -> H -> B
C	3	D	D -> C
E	3	D	D -> E
F	5	C	D -> C -> F
G	5	E	D -> E -> G
H	8	G	D -> E -> G -> H

d) An algorithm to find the optimal location for two stations would be as follows. We would place each fire station location in a list with all the potential places that would need to be reached. We would then loop over each potential place from the first location and compare it to the second location using r roads. The time complexity for this algorithm would be O(f^3) because of three loops. One loop to get the distance to every location from the first location, another to get the distance to every location from the second location, and one to return what the min distance is for each location.

e) Using this method, the optimal locations for two stations would be H and C. From H locations B, G, and H are easily accessible. From C, the paths to A, E, D, and F are short as well. The maximum distance would be from C to E which is 6.

2. a) I would use a Breadth First Search algorithm (BFS). Using this method, the weight of the edges have to be at least W. Therefore, we can ignore the edges whose weights are less than W. The algorithm would start at s and check the near edges. If an edge has a weight of less than W, then it can be ignored as a possible path. This process will repeat until there are no more edges or when we find our way to t.

   b) The running time for this algorithm would be based on the algorithm traversing the graph. Therefore, running time would be Θ(V + E) using BFS. This run time is dependent on the amount of V vertices and V – 1 edges.

3. a) First we would need to show the rivalries as a graph G(V, E) where each vertex represents a wrestler and each edge represents a rivalry. The graph will have n vertices and r edges. Therefore, |V| = n and |E | = r. From here, we would perform BFS as many times as necessary to visit all vertices. We would then assign all wrestlers whose distance is even to be babyfaces and those whose distance is odd to be heels. Then we would verify each edge to see if it goes between a babyface and a heel.

b) Because I am utilizing the BFS algorithm, the expected runtime of would be $\Theta(V + E)$.

c) Implementation of program wrestler.cpp