George Lenz
HW 5
7/29/2018

Question 1:

a) The minimum spanning tree will be the same because the edges are only changing by a constant all the way around so the smallest edge is still the smallest and the biggest is still the biggest so it will not change.

b) the shortest path may change because depending on the number of edges, the total may increase by 1 per edge, so if a path had a less edges it may increase by a total less than a path with a longer edge. For example if one path has goes from a to d and as a->b = 1 b->c-1 and c->d = 1 but another path goes from a->d = 4 ,then the shortest path at first is the one with three edges as it is equal to 3, but if you increase them all by 1, then a->d = 5 and a->b->c->d now is equal to 6 and the shortest path has changed.

Question 2
a) You can run a depth first search from s and end when it gets to t but instead of always taking the path with the lowest weight you can change it to take a path only with weights that fit the criteria, or in this case if the weight is less than or equal to W. If it can not find a path that fits the criteria you can have the algorithm return something to say so.

b) the running time of the algorithm is upwardly bound by the running time of a DFS on the whole graph so it is O(V + E), it is not theta because you may not have to go through the whole graph, for example if the path from s->t is direct and the edge weight fits the criteria.

Question 3
a) You can run Dijkstra's as a shortest path algorithm to that will give you the shortest path to each location from the firehouse. I would run as a typical Dijkstra's algorithm in which you start at G, and then you will find he distance the adjacent vertices. From there you will choose the one with the smallest distance and then find it's adjacency list, and add their edge distances. Then you take the next vertices that haven't been examined and choose the next smallest one. Eventually you will have a list of all the vertices with the smallest distance to each one. The only difference is that you will have to treat each edge as bidirectional.

	S	D_v	P_v
A	T	12	C
B	T	6	H
C	T	8	D
D	T	5	
E	T	2	
F	T	8	
G	T	0	-
H	T	3	

b) You can run Dijkstra's algorithm on all of the vertices and find where the longest intersection lies. From there you can take each one of the longest intersections and find which one is the minimal and choose the source which produced that minimal longest intersection. The running time depends on what type of priority cue you are using, but essentially you will run dijkstra's algorithm f times, so if you're using a binary heap the algorithm comes out to theta(f*(f lg f + r)) = theta(f^2 lg f + f*r).

c) The "optimal" location would b E

Question 4:

a)

```
initialize wresters
Q = {first wrestler in list}
first wrestler team = babyface
while (Q is not empty)
    u = dequeuer(Q)
    for each wrestler adjacent to u
        if already discovered
            if wrestling team is same as parent
                end program, result is impossible
        if not already discovered
        set wrestler team to opposite of parent
        set parent to u
        ENQUEUE wrestler
    set wrestler u to discovered
```

b) The running time of the algorithm is the same as the BFS which is theta(V+E)