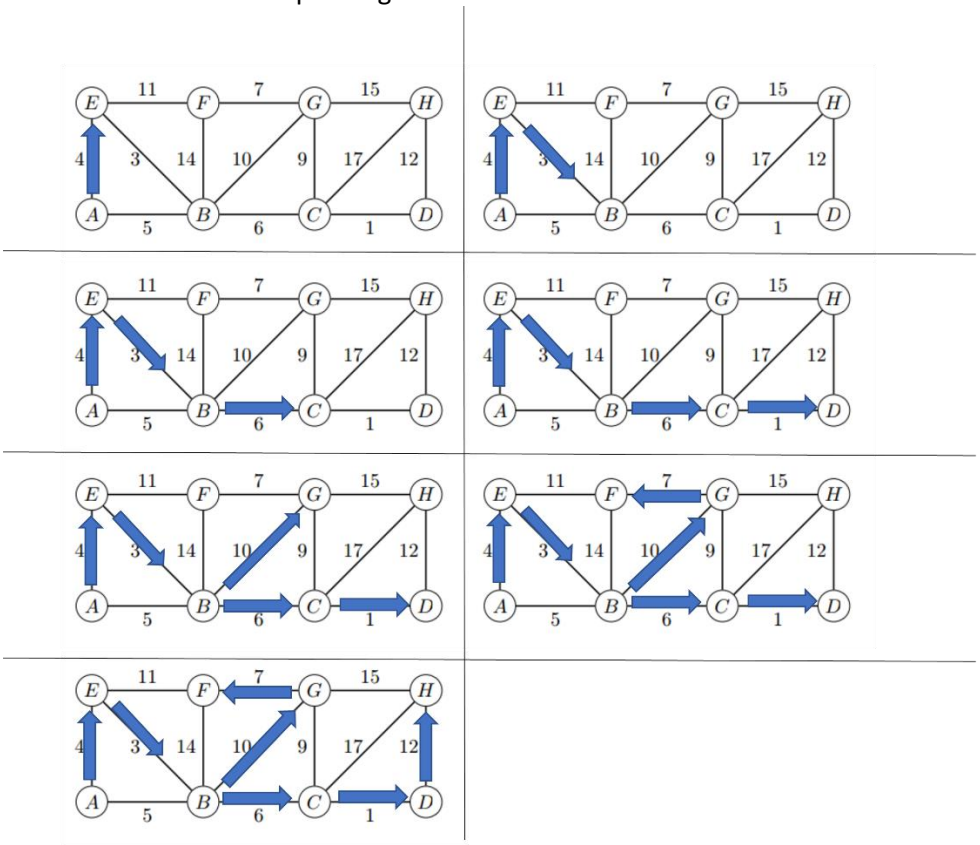
1). A) Demonstrate Prim's algorithm starting from vertex A. Write the edges in the order they were added to the minimum spanning tree.



The total weight of this MST is 43.

B) If each edge weight is increased by 1 will this change the minimum spanning tree? Explain.

If each edge weight is increased by 1, this will not change the minimum spanning tree. Considering that all the edges are increased by the same amount of 1, the graph structure itself does not change. Though the total weight of the MST might change due to the increment, but the overall structure of the MST remains the same.

2). A) What algorithm would you recommend be used to find the fastest route from the fire station to each of the intersections? Demonstrate how it would work on the example above if the fire station is placed at G. Show the resulting routes and times.

I would use the Dijkstra's shortest path algorithm to find the fastest route. Here are the results:

Vertex	Time (Minutes)	Parent	Shortest Path/Route
G	0	NONE	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) Suppose one "optimal" location (maybe instead of G) must be selected for the fire station such that it minimizes the time to the farthest intersection. Devise an algorithm to solve this problem given an arbitrary road map. Analyze the running time complexity of your algorithm when there are f possible locations for the fire station (which must be at one of the intersections) and r possible roads.

Let's run the Dijkstra's algorithm with each single vertex as the source. For each vertex, remembers the maximum distance after extracting all the paths from that vertex, and then select the smallest maximum distance. Given that we just need to run Dijkstra's algorithm for f times, since we can write the Dijkstra's algorithm in a lot of different ways with different time complexity, we can pick one of the most efficient algorithms by using the Fibonacci heaps which the time complexity is just $f * O(r + f \log f)$. Therefore, the overall time complexity would be $O(rf + f^2 \log f)$.

C) In the above graph what is the "optimal" location to place the fire station? Explain.

By using the algorithm above, we can place the fire station at location "E" because the longest distance is 10, which it is from E -> D -> C -> A, and it is also the shortest "maximum" distance compare to all other locations.

3). A) Give pseudocode for an efficient algorithm that determines whether it is possible to designate some of the wrestlers as Babyfaces and the remainder as Heels such that each rivalry is between a Babyface and a Heel. If it is possible to perform such a designation, your algorithm should produce it.

```
Create a map<string, string> container
Initialize a wrestler array to take the wrestlers
Initialize a rivalries 2d array to take a pair of wrestlers at a time
        Initialize the first wrestler in the pair as a "Babyface" for consistence assignment
Assign "Empty" values to all wrestlers in the container

While(1){
        Check = 0
        For i<- 0 where i<# of pairs
                If container[rivalries[i][first wrestler]] == Babyface
                        If container[rivalries[i][second wrestler]] == Babyface
                                Quist the program //since they are the same type
                        else If container[rivalries[i][second wrestler]] != Heel
                                change the current container value to Heel
                                sets check to 1

                If container[rivalries[i][first wrestler]] == Heel
                        If container[rivalries[i][second wrestler]] == Heel
                                Quits the program //since they are the same type
                        else If container[rivalries[i][second wrestler]] != Babyface
                                change the current container value to Babyface
                                sets check to 1

                if container's second wrestler at i == Babyface but the first wrestler is not
                Heel
                        set container's first wrestler at i to be Heel
                        set check to 1

                if container's second wrestler at i == Heel but the first wrestler is not
                Babyface
                        set container's first wrestler at i to be Babyface
                        set check to 1
        For Loop Ends

                if check equals to 0
                        For j<-0 where j<# of wrestlers
                                If the wrestler in container at j is "Empty"
                                Set its value to Babyface
                                Sets check to 1
                                Break the loop
                        For Loop Ends
```

If check still equals 0 //which means that we are done
Then break the while loop

While Loop Ends

B) What is the running time of your algorithm?

The time complexity of my algorithm is O(r*n) as r is the number of rivalries and n is the number of wrestlers.

C) Implementation

Has been turned in to Teach as a Zip file.