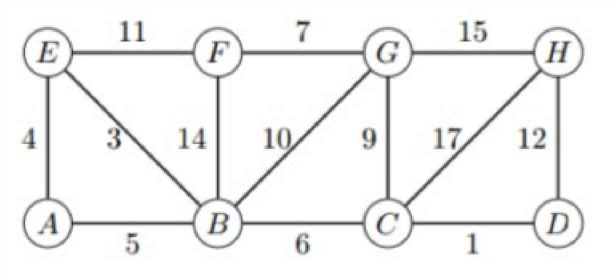
1 – TREVOR STAHL STAHLTR@OREGONSTATE.EDU

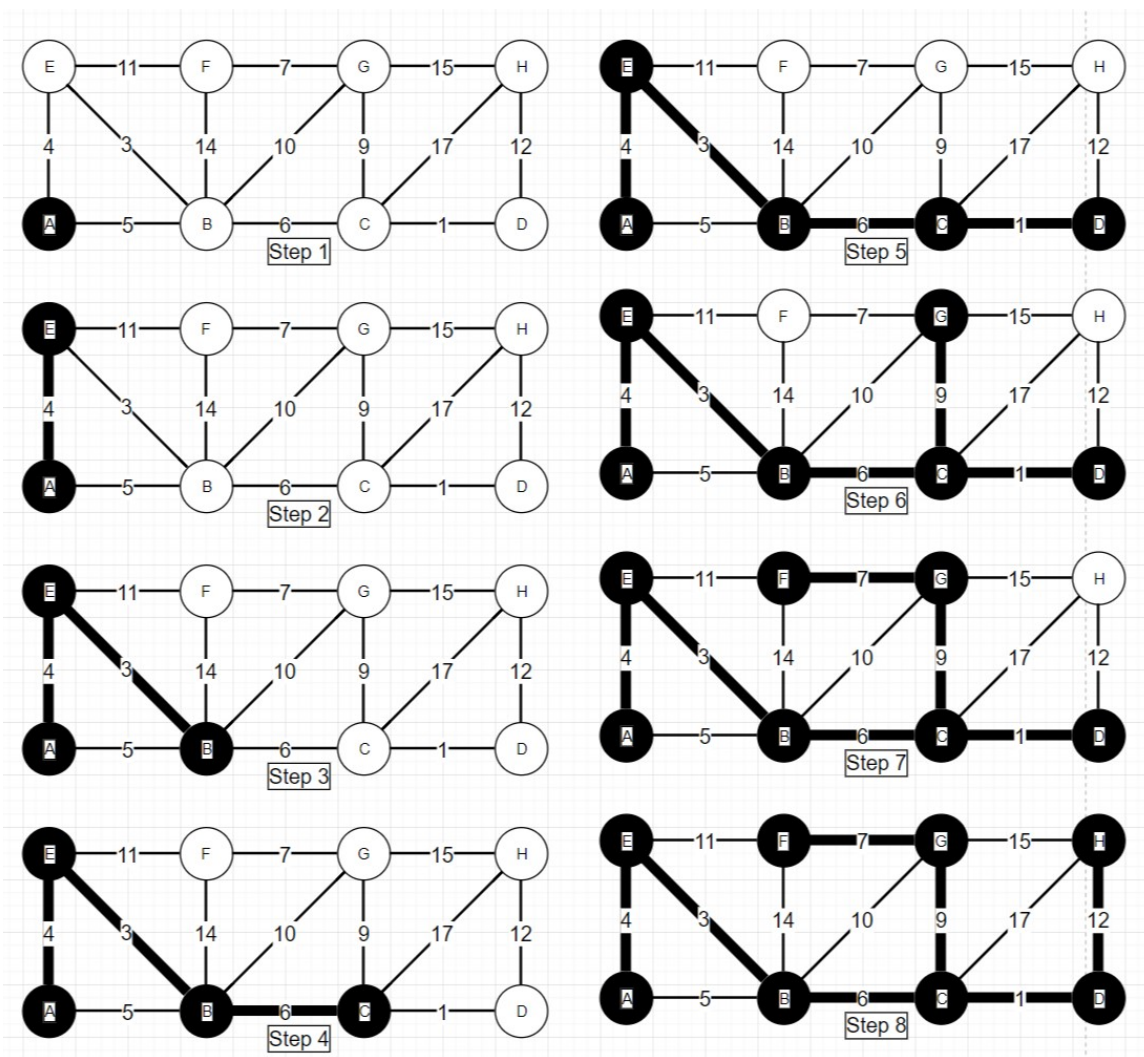
Consider the weighted graph below:



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

Prim – Extend a tree by including cheapest outgoing edge.

Root vertex is A.



List of edges as they were added:

(A,E), (E,B), (B,C), (C,D), (C,G), (G,F), (D,H)

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

No, because the differences between the weights will be the same. So, the order to take them will be the same.

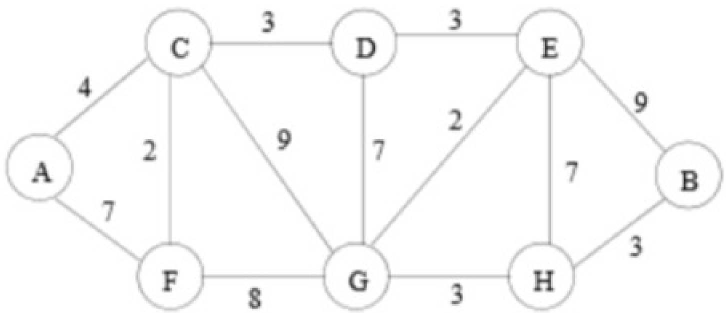
2

A region contains a number of towns connected by roads. Each road is labeled by the

average number of minutes required for a fire engine to travel to it. Each intersection is labeled with a

circle. Suppose that you work for a city that has decided to place a fire station at location G. (While this

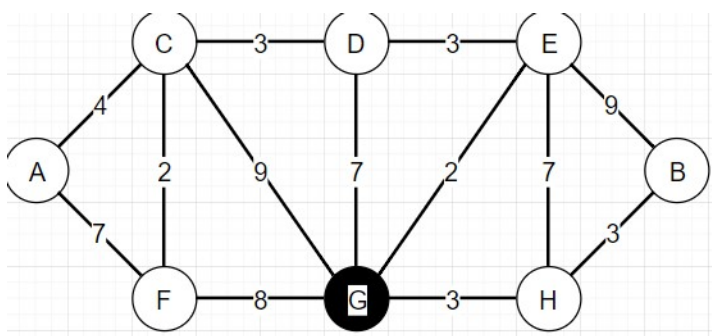
problem is small, you want to devise a method to solve much larger problems).



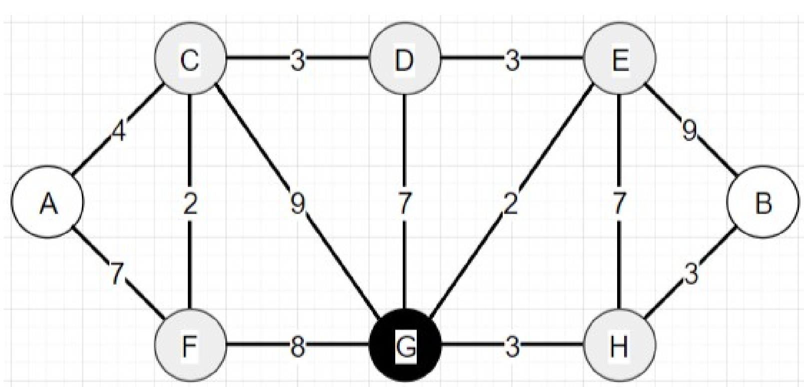
(a) What algorithm would you recommend be used to find the fastest route from the fire station to each of the intersections? Demonstrate how it would work on the example above if the fire station is placed at G. Show the resulting routes and times.

Dijkstra's algorithm with source/root of G for undirected non-negative cyclic graph.

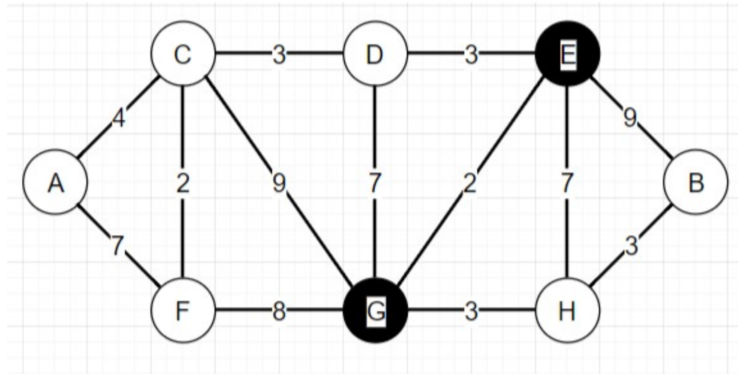
	S	Distance	Parent
A	F	INF	-
B	F	INF	-
C	F	INF	-
D	F	INF	-
E	F	INF	-
F	F	INF	-
G	T	0	NIL-Source
H	F	INF	-



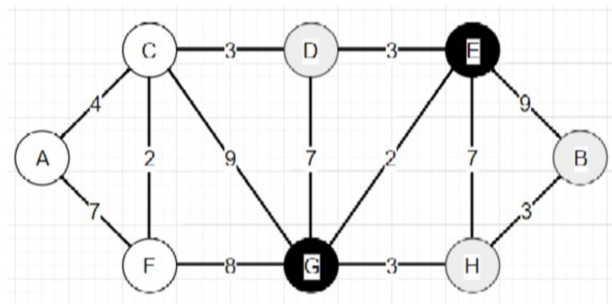
	S	Distance	Parent
A	F	INF	-
B	F	INF	-
C	F	9	G
D	F	7	G
E	F	2	G
F	F	8	G
G	T	0	NIL-Source
H	F	3	G



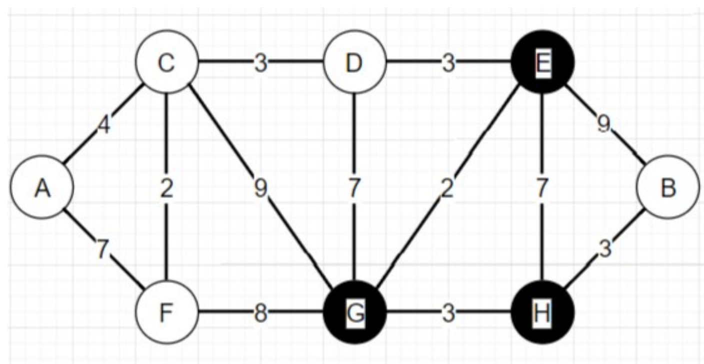
	S	Distance	Parent
A	F	INF	-
B	F	INF	-
C	F	9	G
D	F	7	G
E	T	2	G
F	F	8	G
G	T	0	NIL-Source
H	F	3	G



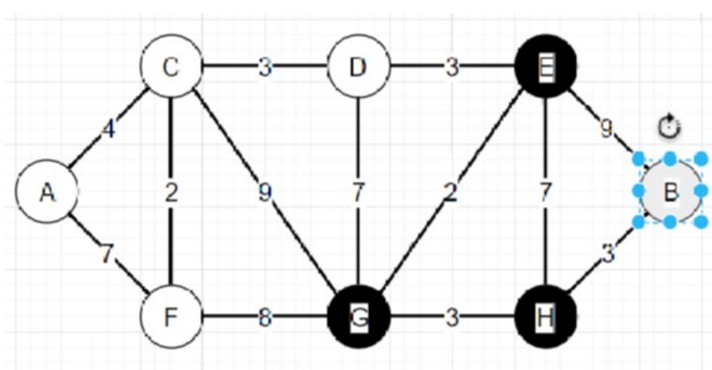
	S	Distance	Parent
A	F	INF	-
B	F	11	E
C	F	9	G
D	F	5	E
E	T	2	G
F	F	8	G
G	T	0	NIL-Source
H	F	3	G



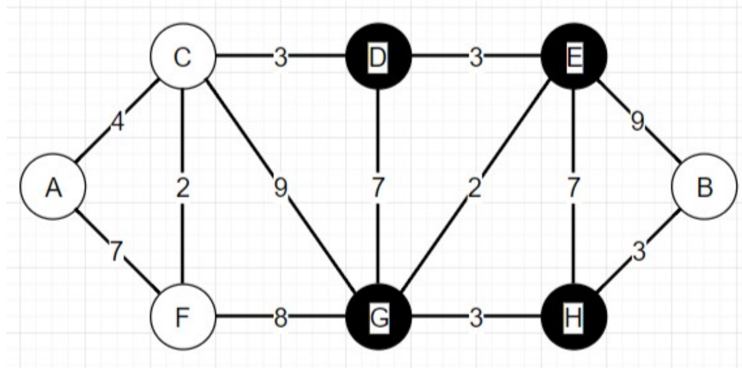
	S	Distance	Parent
A	F	INF	-
B	F	11	E
C	F	9	G
D	F	5	E
E	T	2	G
F	F	8	G
G	T	0	NIL-Source
H	T	3	G



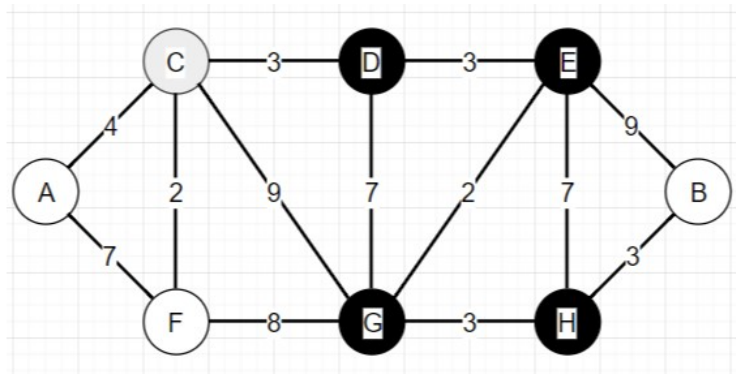
	S	Distance	Parent
A	F	INF	-
B	F	6	H
C	F	9	G
D	F	5	E
E	T	2	G
F	F	8	G
G	T	0	NIL-Source
H	T	3	G



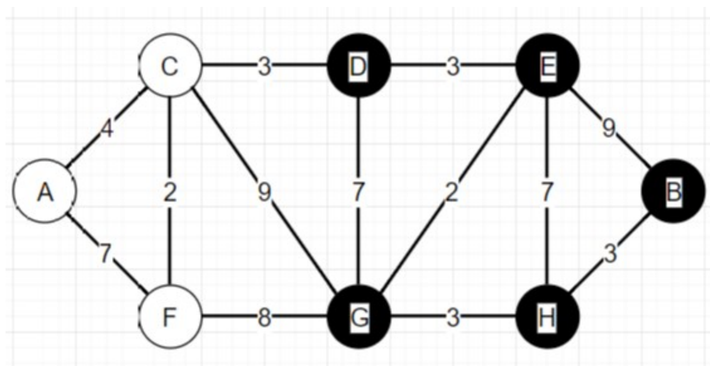
	S	Distance	Parent
A	F	INF	-
B	F	6	H
C	F	9	G
D	T	5	E
E	T	2	G
F	F	8	G
G	T	0	NIL-Source
H	T	3	G



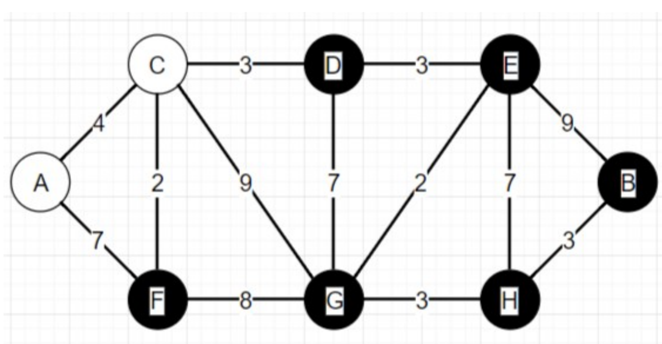
	S	Distance	Parent
A	F	INF	-
B	F	6	H
C	F	8	D
D	T	5	E
E	T	2	G
F	F	8	G
G	T	0	NIL-Source
H	T	3	G



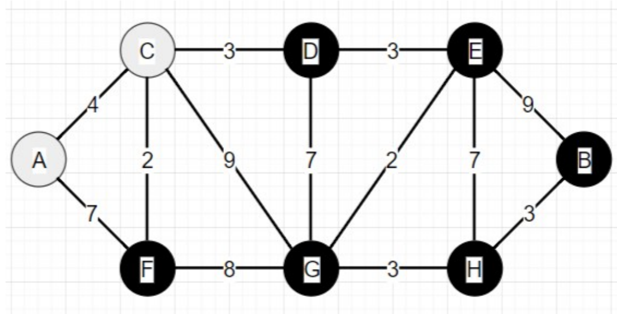
	S	Distance	Parent
A	F	INF	-
B	T	6	H
C	F	8	D
D	T	5	E
E	T	2	G
F	F	8	G
G	T	0	NIL-Source
H	T	3	G



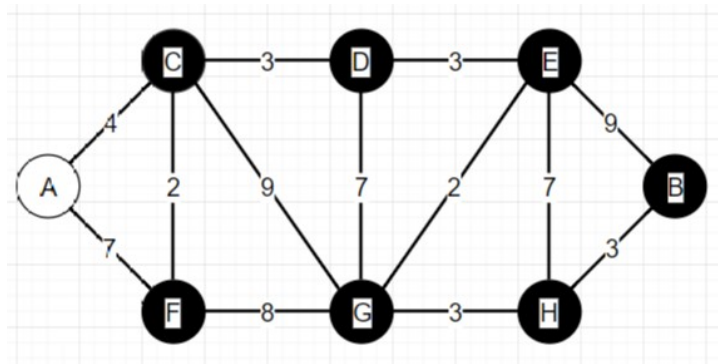
	S	Distance	Parent
A	F	INF	-
B	T	6	H
C	F	8	D
D	T	5	E
E	T	2	G
F	T	8	G
G	T	0	NIL-Source
H	T	3	G



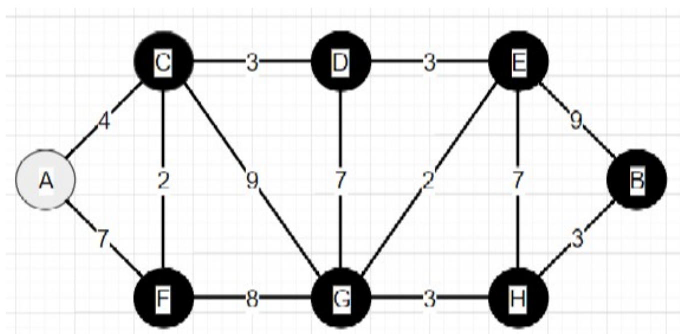
	S	Distance	Parent
A	F	15	F
B	T	6	H
C	F	8	D
D	T	5	E
E	T	2	G
F	T	8	G
G	T	0	NIL-Source
H	T	3	G



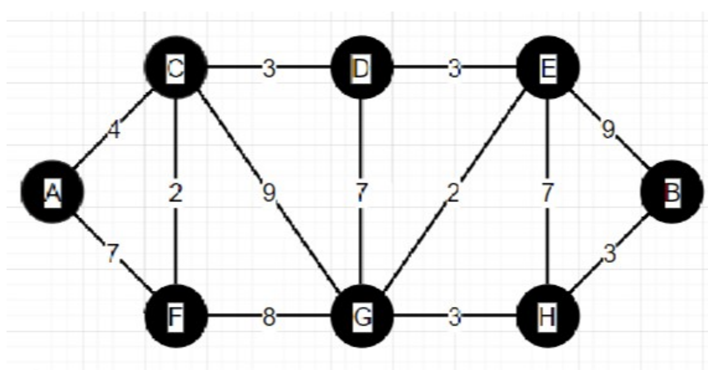
	S	Distance	Parent
A	F	15	F
B	T	6	H
C	T	8	D
D	T	5	E
E	T	2	G
F	T	8	G
G	T	0	NIL-Source
H	T	3	G



	S	Distance	Parent
A	F	12	C
B	T	6	H
C	T	8	D
D	T	5	E
E	T	2	G
F	T	8	G
G	T	0	NIL-Source
H	T	3	G



	S	Distance	Parent
A	T	12	C
B	T	6	H
C	T	8	D
D	T	5	E
E	T	2	G
F	T	8	G
G	T	0	NIL-Source
H	T	3	G



Answer Table - ideal routes and best times

	S	Distance	Parent
A	T	12	C
B	T	6	H
C	T	8	D
D	T	5	E
E	T	2	G
F	T	8	G
G	T	0	NIL-Source
H	T	3	G

(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.

Algorithm:

One algorithm would be to run Dijkstra’s SSSP for an undirected graph on every vertex. Then with these results for each vertex choose the vertex that had the smallest max value in their distance table. This is correct because Dijkstra’s algorithm is correct and find the best routes given a source to start from. The source that returns a table of distances with the smallest max distance out of all the other vertices would be the ideal vertex to choose because it has the smallest distance to its furthest vertex as desired.

Running time:

Part of the running time would be the running time of Dijkstra’s algorithm times the number of vertices. So using the ideal Dijkstra implementation it would be Theta(|V|) * O(V lg V + E)

which is O(V^2 lg V + EV)

The rest of algorithm would be finding the max distance value for each table, and there are V tables so V^2 because for each V we must find the max from V values which is linear. And this is dropped in aggregate analysis for asymptotic behavior. Returning the value is Theta(1) and would also be dropped in the final analysis.

(c) In the above graph what is the “optimal” location to place the fire station? Explain.

E is optimal, it has max distance to A of 10. I found this because first the shortest path between A and B which are the farthest apart visually and by numbers of edges is through C D and E. So it must be C D or E. Going through each showed that E was the ideal with total weight of edges on each side being as balanced as possible.

3

Suppose there are two types of professional wrestlers: “Babyfaces” (“good guys”) and

“Heels” (“bad guys”). Between any pair of professional wrestlers, there may or may not be a rivalry.

Suppose we have n wrestlers and we have a list of r pairs of rivalries.

Input is read in from a file specified in the command line at run time. The file contains the number of wrestlers, n, followed by their names, the number of rivalries r and rivalries listed in pairs. Note: The file only contains one list of rivalries

Results are outputted to the terminal.

- Yes, if possible followed by a list of the Babyface wrestlers and a list of the Heels.
- No, if impossible.

(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.

```
Wrestler (list of wrestlers, list of rivalries){
    Assign first wrestler in list to be baby face
    While not all wrestlers are assigned:
        For each rivalry
            Check if one of wreslters is assigned a group yet
            If so assign opposite group to other wrestler
            If they are assigned the same group already return No, Impossible
        If no changes made since last for loop then assign first wrestler in the list who is still not
        assigned a group to be baby faced.
    Print out necessary text to show results.
}
```

(b) What is the running time of your algorithm?

Assigning first wrestler is theta(1). Each pass through the while loop will assign at least one wrestler a group. So this will happen at most N times, where is N is number of wrestlers so O(N). Printing the necessary text if the algorithm found a full set of designations will take O(N). Each loop through the while loop also takes O(R) where R is number of rivalries. Thus, we have O(N*R).

(c) Implement: Babyfaces vs Heels in C, C++ or Python. Name your program wrestler and include compile and executions instructions in the README file.