Nicole Goodman CSE 100 Final

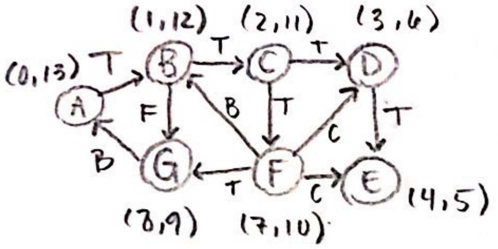
1. True Dijkstra's algorithm is a greedy algorithm meaning it finds the shortest path between all pairs of vertices. The vertices will be added to the set S from the least to greatest weight. Therefore, the given claim $\delta(s,v_1) \leq \delta(s,v_2) \leq \dots \leq \delta(s,v_n)$ is true.

2. True MST algorithms chooses the smallest weight that does not cause a cycle. MST cannot have cycles b/c there will be a never ending connection in that given cut. In the choosing process of edges, I assume that edge e would be chosen last as it has the maximum amount of weight (not a light edge). In both MST algorithms (Prim and Krushal) the safe edge is chosen by the lightest weighted edge, therefore edge e is not safe b/c it is the heaviest edge in the cycle. Every vertex in cycle C will be apart of the tree, so edge 'e' will not be able to cross any cuts.

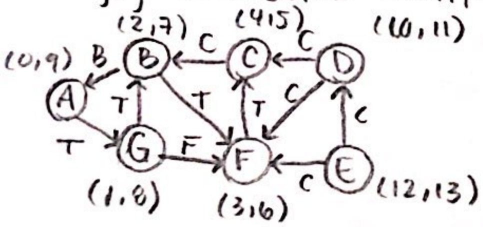
3. The Bellman ford algorithm takes the source vertex (s) and finds the shortest path from S to V. Since the graph has no negative weight, the runtime is O(VE) as given by the book (pg. 651). In order to find the shortest path from S to a given vertex in time O(E+V), we must use the DFS algorithm on the graph. Using the pseudocode from the book on pg. 604. Using an adjaceny list the algorithm will have to visit all vertices and edges. Every edge is seen at most twice which is O(E) in an undirected graph and once in a directed graph O(V). Therefore the runtime is O(V+E).

4. In order to test Dr. Sponge's theorem, I would use BFS. Given that the edges have non negative weight and we know that BFS has a runtime of O(V+E) according to the book, this is a plausible option. The algorithm will compute the smallest number of edges from the source vertex (s). In the Enqueuing and Dequeuing process the runtime is O(1) where the time spent on the queuing process is O(V). The algorithm will then scan the adjacency list which will take O(E) time. The overall runtime of the BFS algorithm is O(V+E) which can be use to test Dr. Sponge.

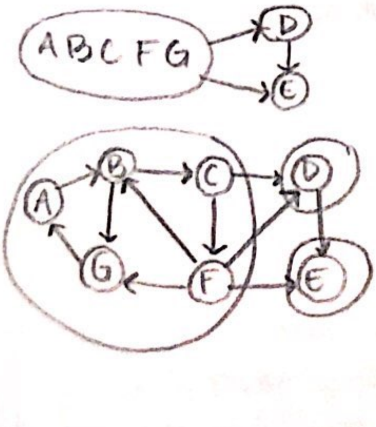
5. in this case I woud use BFS. since we are limited to k edges we must mark each vertex with the number of edges followed. in order to do this each visited edge would be marked black. As the algorithm continues to Enqueue and dequeue each vertex, it will stop once H reaches k's limit. The time devoted to queue operations is O(V) time. since the algorithm scans the adjacency lists of each vertex when the vertex it dequeued, it take O(E) time. Therefore, the total time of BFS is O(V+E).

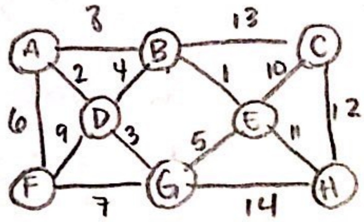
6. 

b) strongly connected components of G



c)



7. 

a) (B, D) is a safe edge because it has the smallest weight amongst the connecting vertices. There fore it will be the cheapest cut to make.

b) (B, E) = 1
(A, D) = 2
(D, G) = 3
(B, D) = 4
(A, F) = 6
(C, E) = 10
(E, H) = 11

c) (A, D) = 2
(D, G) = 3
(B, D) = 4
(B, E) = 1
(A, F) = 6
(C, E) = 10
(E, H) = 11