HW4 - S16

Handed out 4-20 — Chris Troiano — 4-26, beg. of class

1. (a) Run Dijkstra's algorithm on the following graph (showing all intermediate steps).
   (b) Show the shortest path tree corresponding to running Dijkstra on this graph.

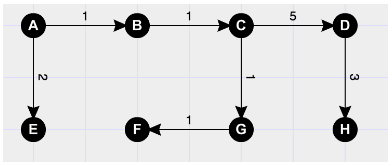
Solution:

(a) Table:

Table 1: Dijkstra's Shortest Path

iteration	A	B	C	D	E	F	G	H
0	0	∞	∞	∞	∞	∞	∞	∞
1	0	1	∞	∞	2	10	∞	∞
2	0	1	2	∞	2	6	4	∞
3	0	1	2	7	2	4	3	∞
4	0	1	2	7	2	4	3	10

(b) Graph:



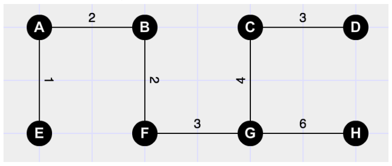
2. Run Kruskal's algorithm on the following graph (showing all intermediate steps).

Solution:
Edges ordered by weight:

$A-E:1$
$A-B:2$
$B-F:2$
$C-D:3$
$F-G:3$
$C-G:4$

~~$E-F:4$~~
~~$B-E:5$~~
~~$C-F:5$~~
~~$D-G:5$~~
$G-H:6$
~~$B-C:7$~~

~~$D-H:7$~~

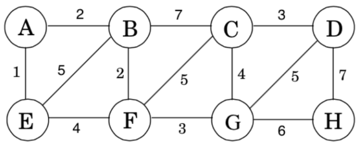


3. (a) Consider the edge (B, E) in the graph of Problem 2. Either show that there is a minimum cost spanning tree using this edge by applying the cut property, or prove that this edge is not used in any minimum cost spanning tree using the cycle property.

(b) Ditto for edge (C, G).

Solution:

(a) Observe cycle A, B, E

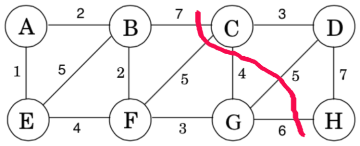


$A - E : 1$
$A - B : 2$
$B - E : 5$

By the cycle property, $B - E$ is not in the MST.

(b) Let S be the subset of nodes C, D, H. The cutset contains edges:

$B - C : 7$
$C - F : 5$
$C - G : 4$
$D - G : 5$
$G - H : 6$



$C - G$ is the min cost edge with exactly one node (C), in S. Therefore the MST contains the $C - G$ edge.

4. Prove or disprove the following statements about an arbitrary undirected graph $G = (V, E)$:

(a) If e is part of some MST of G, then it must be a lightest edge in some cutset of G.

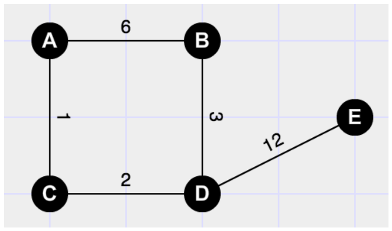
(b) If graph G has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree.

(c) The shortest path between two nodes is necessarily part of some MST.

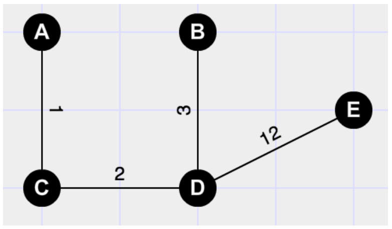
Solution:

(a) Let S be a subset of nodes in G where edge e has exactly one endpoint in S. Let T be a MST of G that does not contain e. When we add e to T, it creates a cycle C. Edge e is in the cycle C & the cutset X. There exists another edge l, that is in both C & X.
$T' = T \cup \{e\} - \{l\}$ is also a spanning tree.
$e < l, cost(T') < cost(T)$, therefore, e is the lightest edge in cutset X.

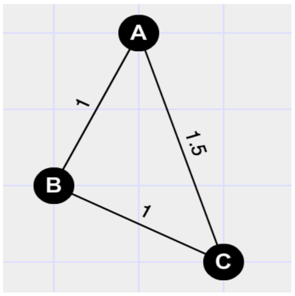
(b) This is false because the longest edge may be the only edge connecting some node to the MST and thus, must be included.



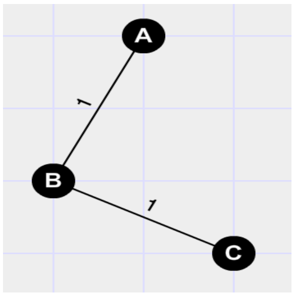
$\Rightarrow$



(c) This is false, consider the undirected graph below. The shortest path between A & C is 1.5, but if Kruskal's algorithm is used to find the MST, it will be composed with the two lighter edges.



$\Rightarrow$



5. Consider an undirected graph $G = (V, E)$ with non-negative edge weights $w_e \geq 0$. Suppose that you have computed a minimum spanning tree of G, and that you have also computed shortest paths to all nodes from a particular $s \in V$.

   Now suppose each edge weight is replaced by $w'_e = f(w_e)$, where f is a strictly increasing function.

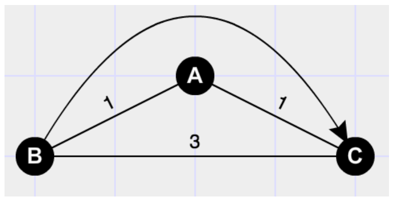
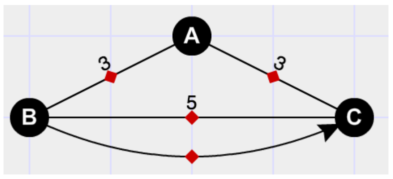
   (a) Does the minimum spanning tree change? Give an example where it changes or prove that it cannot change.

   (b) Do the shortest paths change? Give an example where they change or prove they cannot change.

   Solution:

   (a) If each edge weight is replaced by $w'_e = f(w_e)$, the MST will not change. This is best illustrated with Kruskal's algorithm. If we order the all the edges, w_e, they will stay in the same order when we apply the strictly increasing function $w'_e = f(w_e)$, we know this because the function is strictly increasing and not sinusoidal or decreasing. Therefore, the algorithm will produce the same MST.

   (b) The shortest paths can change. Consider the graphs below:

 $f(w_e) = w_e + 2 \Rightarrow$ 

Initially the shortest path from $B - C$ is $B - A$ then $A - C$. After applying the strictly increasing function to each node, the shortest path is now $B - C$.

6. **Extra Credit**. Let $G = (V, E)$ be a directed graph. A path $(v_1, v_2, \ldots, v_n)$ in G is called a **simple path** if it does not have cycles, meaning that no node repeats twice in the path. Note, that for the shortest path problem with non-negative weights, any shortest path will always be a simple path. This may not be the case when we allow negative weights. For the purpose of this problem however, we will consider the problem of finding the **shortest simple paths** from a given source.

   It is known that Dijkstra's algorithm may not return the correct result to this problem if a graph contains edges with negative weights. What if we only allow negative weights on the edges leaving the source node (and all other edges are guaranteed to have non-negative weights)? Will Dijkstra's algorithm be always correct in this case (for finding shortest simple paths)? If yes, prove it, if no, provide a counter-example.

Solution:
Dijkstra's algorithm will not always be correct. Consider a graph where there is a relatively large negative edge weight involved in a cycle. The algorithm will find that every time around the cycle, the cost of the path gets smaller.