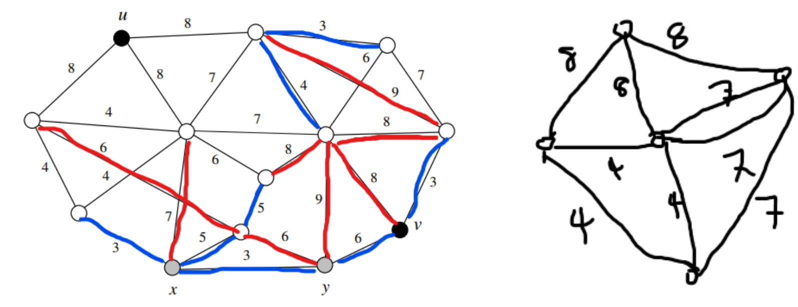
1. (a) 27. Apply blue rule and red rule when there is an unique edge to color, then the blue edges are in any MST and the red edges are not. We regard each subtree as a mega node, then we get the simplified graph (the right one).



(b) No. The edge xy is colored blue and it must be in any MST of G.

(c) 7. If $w(uv) = 7$, then uv is the lightest edge in the cut $[\{u\}, V - u]$ and this will reduce mst(G) by 1. When $w(uv) = 8$, show $mst(G) = mst(G+uv)$.

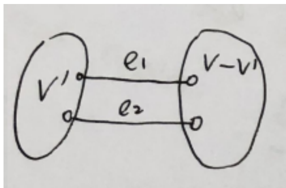
2. Continue from Case 2 in the notes. When e is colored red. If e is not contained in T, then we are done. Otherwise, $e \in T$ amd we will construct a new MST T' that satisfies the color invariant.

Let u, v be two ends of e. The removal of e vertices of G into two subtrees and we denote the subtree containing u as T_1 and the other containing v as T_2.

Consider the cycle to which the red rule is used to color e, there is another edge e' on this cycle such that its two ends u' and v' belongs to T_1 and T_2 respectively.

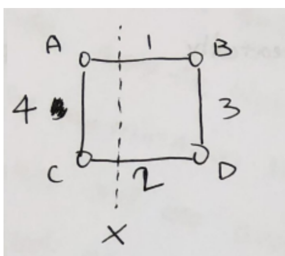
Note that e' is neither blue (every blue edge is in T) nor red (red rule is used for cycles without red edge), and $w(e) \geq w(e')$. Therefore, $T' = T - e + e'$ is an MST satisfying the color invariant.

3. (a) Incorrect. If a cut $[V', V - V']$ has two edges e_1, e_2, the algorithm may add both edges into T which violates the blue rule.



(b) Correct. Contracting an edge $e = \{u, v\}$ removes every cut $[V', V - V']$ with $u \in V'$ and $v \in V - V'$ and keeps all cuts with both u and v on the same side. Blue rule does the same.

(c) Incorrect. In the following example, the algorithm may delete edge cd for the cut X. But observe that $6 = mst(G) < mst(G - cd) = 8$.



(d) Incorrect. Consider the same example in (c), the algorithm may first pick a and add ab into T, and then pick c and add cd into T. Now every vertex incident to some blue edges, the algorithm output a disconnected T.

4. (a) Addition of edge uv: the edge uv and the $u - v$ path in T forms a cycle without red edge, we can apply red rule. This takes $O(m + n)$ time.

(b) Deletion of an edge uv: if $uv \notin T$, we are done; otherwise, the deletion of uv break T into two trees T_1, T_2 where $u \in T_1$ and $v \in T_2$. There exists exactly one cut $[V(T_1), V(T_2)]$ without blue edges, and we apply Blue Rule to it. This takes $O(m + n)$ time.

(c) Change weight of edge uv: delete uv and then add it back.

5. Constructing G^* by adding a new vertex s' and for each source $s \in G$, add a new edge $s's$ with weight 0. Clearly there is no negative cycle in dag G^* and we can set $h(v) = d_{G^*}(s', v)$ for all $v \in V$. The single source shortest path of G^* from s' can be calculated in $O(m + n)$ time for dag G^*

as follows. First, topologically sort the vertices and then in this order set $dist(v) = min(dist(u) + w(uv))$ for each $uv \in E(G^*)$ and $dist(s') = 0$.

6. Yes. First, Dijkstra's algorithm still terminates. When calculatiug distance via a negative edge with tail s, because there is no negative cycle $d(s) = 0$ will not change.

Next, we show that $d(u) = \delta(s, u)$ for each u added to S. Denote the distance of the shorteste distance from s to u as $\delta(s, u)$. When running Disjkstra's algorithm, once u as added to S, $d(u)$ is unchanged and should be $\delta(s, u)$.

Suppose u is the first vertex added to S for which $\delta(s, u) \neq d(u)$. Let p be the shortest (s, u)-path, p must contains a negative edge e. Since every negative edge has s as tail, we partition the path p into p_1 and p_2 where p_1 starts with the first vertex of p and ends with the tail of e, and p_2 contains the rest. Since G has no negative cycle, we can see p_2 is a shorter (s, u)-path than p. This is a contradiction, so u doesn't exist.