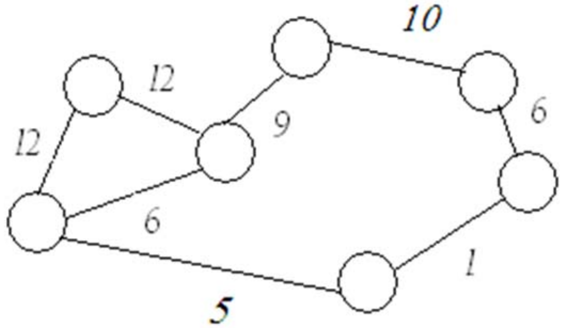
1 ▸ For weighted graph $G=(V,E)$, let $\mu(G)=\max_{e\in E}\{w(e)\}$. For example, if G is



then $\mu(G)=12$.

a Describe an efficient algorithm to accept as input a connected weighted graph *G* and yield as output a spanning subgraph *G'* of *G* to minimize $\mu(G')$. That is, *G'* should satisfy

$$\mu(G')=\min_{\text{spanning subgraph } H \text{ of } G}\{\mu(H)\}.$$

b Prove that your algorithm works correctly.

c Describe an algorithm which accepts as input connected weighted graph $G=(V,E)$ and $\omega\in\mathbb{R}^+$ and tests whether *G* admits a spanning tree *G'* such that $\mu(G')\le\omega$. Your algorithm should work in worst-case time in $O(m)=O(|E|)$.

<u>SOLUTION</u>: ***a*** A MST *G'* of *G* also minimizes $\mu(G')$ over all spanning trees of *G*. So Prim's algorithm or Kruskal's algorithm or any algorithm to compute a MST of *G* would solve our problem.

b Assume $G'=(V,E')$ is a MST of *G* but $G''=(V,E'')$ is a spanning tree of *G* with $\mu(G')>\mu(G'')$. Choose some $e\in E'$ satisfying $w(e)>\mu(G'')$. There must be some edge *e'* from E'' in the principal cut of G' and *e*. Because $w(e)>\mu(G'')$, replacing *e* in E' yields a spanning tree of lower weight than *G'*, contradicting that *G'* is a MST. Hence, *G'* must minimize $\mu(G')$ over all spanning trees of *G*.

c	$E'\leftarrow\{e\in E \mid w(e)\le\omega\}$	$O(m)$
	Test if (V,E') is connected	$O(m)$

****

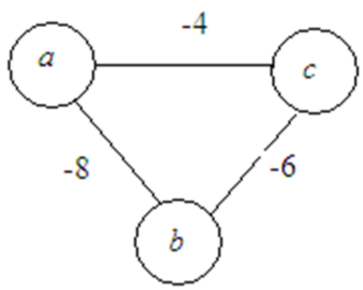
2 ▸ A $(|V|+15)$-*tree* of a weighted graph is a subgraph with $(|V|+15)$ edges which spans the graph. Note that it's not a tree. Describe an efficient algorithm construct a minimum weight $(|V|+15)$-tree of a connected graph.

SOLUTION: We construct an MST of the graph and add the 16 lightest edges of the graph which do not belong to the MST.

****

3 ▸ As we presented the minimum spanning tree problem, the Blue Rule and the Red Rule assumed that all edge weights are positive, $w:E\to\Re^+$. What happens if we relax this assumption?

In class we showed that if G is connected and $w: E \to \Re^+$ and we apply the Blue Rule and the Red Rule until they can't be applied anymore, then the red edges and the blue edges partition E and the blue edges are a minimum weight spanning subgraph of G.
For the rest of this problem, assume you are given a connected weighted graph $G = (V, E)$ with $w: E \to \Re$.
a Show that applying the Blue Rule and the Red Rule until they can't be applied any more does not necessarily construct a set of blue edges which comprise a minimum weight spanning subgraph of G.
b Modify the Blue Rule and the Red Rule such that when we apply them until they can't be applied anymore, then the red edges and the blue edges partition E and the blue edges are a minimum weight spanning subgraph of G.
Solution: Application of the rules to the graph



yields the tree with edges $\{ab, bc\}$ of weight -14, but the minimum weight spanning subgraph has edges $\{ab, bc, ac\}$ of weight -18.

b. **Blue Rule**: For any edge *e*, color it *blue* if its weight is negative or if there is a cut for which every edge in the cut is *red* or uncolored and *e* has minimum weight of all edges in the cut.
Red Rule: Color any edge *red* if its weight is positive and it belongs to a cycle with no *red* edges and it is the *uncolored* edge in the cycle of maximum weight.
****

4 ▸ Given a connected weighted graph $G = (V, E)$ and edge $uv \in E$, describe an algorithm to test if there exists a MST of G which does not contain *uv*. The execution time of your algorithm should be in $O(n+m) = O(|V|+|E|)$. Prove that your algorithm works.

SOLUTION:

for each $e \in E$ **if** $w(e) > w(uv)$ **then** remove *e* from *E*	$O(m)$
$E \leftarrow E - \{uv\}$	$O(1)$
if *u* and *v* belong to the same component of *G*	$O(m)$
then return "There exists an MST which doesn't contain *uv*"	$O(1)$

PROOF: If the new *E* spans a connected graph, then there exists a path from *u* to *v* of edges with weights $\leq w(uv)$. Thus if we adds *uv* to this path, we can apply the Red Rule to eliminate *uv* from consideration for inclusion in an MST of *G*.