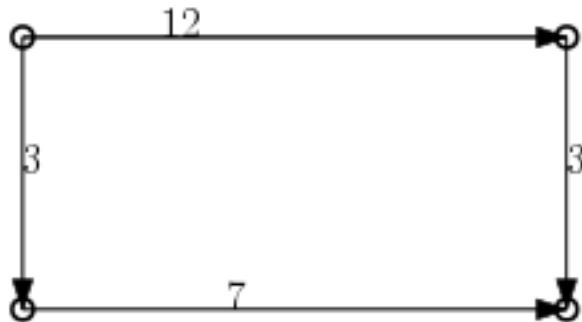


**Problem Set 7 — Minimum Spanning Trees and Single Source Shortest Paths Due by 4:30pm Friday, March 30, 2018 as a single pdf via Moodle (either generated via L<sup>A</sup>T<sub>E</sub>X, or concatenated photos of your work). Late assignments are not accepted.**

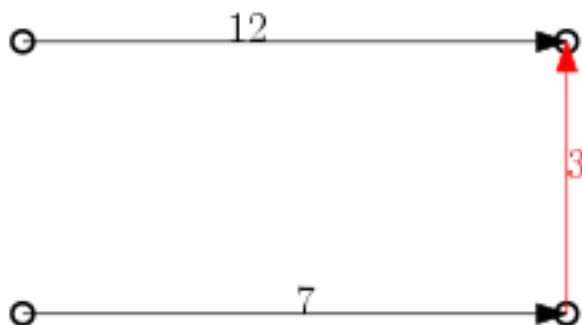
This is an *individual* assignment: collaboration (such as discussing problems and brainstorming ideas for solving them) on this assignment is highly encouraged, but the work you submit must be your own. Give information only as a tutor would: ask questions so that your classmate is able to figure out the answer for themselves. It is unacceptable to share any artifacts, such as code and/or write-ups for this assignment. If you work with someone in close collaboration, you must mention your collaborator on your assignment.

1. Problem 23.1-1 from CLRS. Let  $G = (V, E)$  be a connected undirected graph. Let  $(u, v)$  be a minimum weight edge. Let there be  $T$  which is some subset of  $E$  that does not contain a vertex  $u$ . Since  $(u, v)$  is a minimum weight edge, it is a safe edge. And a minimum spanning tree must contain all vertices including  $v$  the edge  $(u, v)$  will be added.
2. Problem 23.2-8 from CLRS. With the valid input where a tree that has lower weight exists.

We start with this graph given as input.

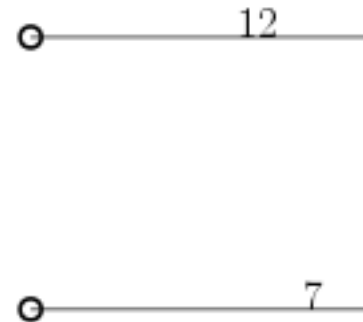


Based on the algorithm we then search the edges that go between the two graphs to find a light edge, and then we add that edge.

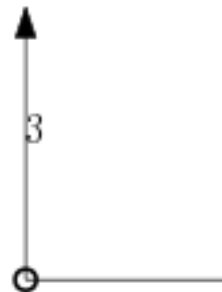


Weight = 22

This is the graph during with  
tion. The third call was a ne  
only two vertices and one ed  
G2 are returned as shown.



Based on the algo  
mum span tree, bu  
can find another  
which is shown be



### 3. Problem 24.1-6 from CLRS.

Invariant: The negative weight cycle is intact right before the start of every loop.

Initialization: At the start of the loop, no edges have been removed nor recorded.

Maintenance: At the start of the loop we first check if Bellman-Ford function returns true. If it returns false we can remove an edge from the graph without changing the cycle and thus passes the invariant. Then when Bellman-Ford returns true, it means that the edge  $u, v$  was a part of the cycle, thus we add back the edge and using the adjacency list do a breadth first search to find the remaining parts of the cycle. If we find another edge  $(v, x)$  when removed causes the BF return true then we change  $u$  to equal  $v$  and  $v$  to equal the current  $x$  so that we search it's neighbors for another edge in the cycle.

Termination: The function terminates if there was no cycle found. It also terminates when  $x = u_1$  with  $u_1$  being the first vertex found to the part of the cycle.

---

```

1: function NEGATIVE-CYCLE( $G$ )
2:    $Q = \emptyset$ 
3:   for all  $(u, v) \in E$  do
4:      $E = E - (u, v)$ 
5:     if BELLMAN-FORD( $G$ ) == true then
6:        $Q = Q + u_1$ 
7:        $u = u_1$ 
8:        $E = E + (u, v)$ 
9:       for all  $x \in \text{adj}[v]$  do
10:         $E = E - (v, x)$ 
11:        if BELLMAN-FORD( $G$ ) == true then
12:           $Q = Q + u$ 
13:           $E = E + (v, x)$ 
14:           $u = v$ 
15:           $v = x$ 
16:          if  $x = u_1$  then
17:            return  $Q$ 
18:   return  $Q$ 

```

---

4. Problem 24.3-4 from CLRS.

- (a) We first check if there is one vertex  $s$  which  $\pi[s] = \text{NIL}$  and  $\text{dis}[s] = 0$ .
- (b) We then check if the information given is a tree on the graph given. We first check if  $\text{dis.length} = \pi.\text{length} - 1$ . We then do a BFS and check if it visits all and  $\pi[v] \neq \infty$ . Then, we check if for each pair  $(u, v)$ ,  $\text{dis}[u] + w(u, v) \geq \text{dis}[v]$
- (c) We check for each  $v \in \pi - s$ , if  $\text{dis}[\pi[v]] + w(\pi[v], v) = \text{dis}[v]$