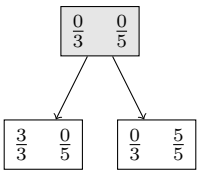
DSC 40B - Discussion 08

Problem 1.

Two containers A and B have capacities of 3 liters and 5 liters, respectively. Describe how you can measure exactly one liter of water using only these two containers. You may 1) fill each container to the top using a faucet; 2) completely empty either container into a drain; and 3) pour one container into the other until it is empty or the other container is full. Both containers are empty to start.

Solution:

While this may not seem like it, we can think of this as a graph problem. Starting with both containers empty, there are two things we can do: we can fill container A or fill container B. We can depict this situation as follows:



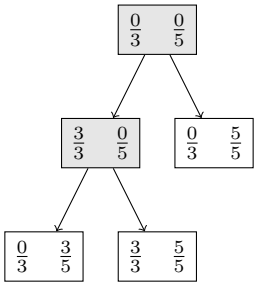
In other words, going from the situation where both containers are empty, we can go to either of two new situations: that in which container A is full and B is empty, or that where A is empty and B is full. In other words, we have a graph in which each node represents a state of the world, and we have an edge between two nodes if we can get from one state to the other. We are in search of a node where one of the containers contains exactly 1 liter.

We have colored the starting node gray in the above depiction to denote that we have already enumerated all of the situations that can be reached in one step, starting at that node. The other nodes are still uncolored: we need to enumerate their successors.

Our next step is to further expand the graph. Suppose we are at the left node in which A is full and B is empty. There are

1. Pour out A. This takes us back to the starting node in which both containers are empty. We could draw an edge back to it, but let's not, since returning to the start cannot get us closer to the solution, and drawing the edge will only clutter the picture.
2. Pour A into B. This results in A being empty and B having 3 liters.
3. Fill B from the faucet. This results in A having 3 liters and B having 5.

We can depict these steps with the following graph:

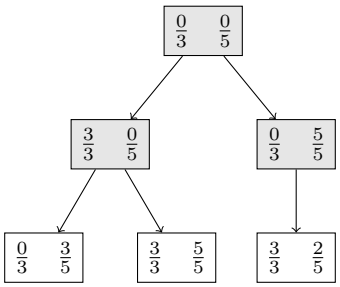


We still haven't found a solution node yet. Let's expand the  node. There are several things we

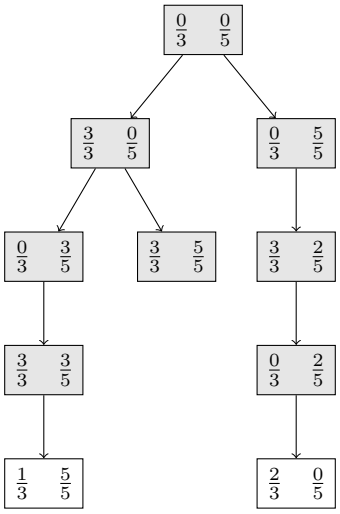
can do:

1. Pour out B. This takes us back to the starting node in which both containers are empty. We could draw an edge back to it, but we won't for the same reasons given above.
2. Pour B into A until A is full. This results in A having 3 liters and B having 2 liters.
3. Fill A from the faucet. This results in A having 3 liters and B having 5. We already have such a node, and we could draw an edge to it, but we won't for the reasons given above.

Our graph becomes:



Continuing this process for several more steps, we will arrive at:



We have found a node in which container A contains 1 liter. We can describe the process followed by taking the path from the root node to this node:

1. Fill A from the faucet: $\frac{3}{3}$ $\frac{0}{5}$
2. Pour A into B: $\frac{0}{3}$ $\frac{3}{5}$
3. Fill A from the faucet: $\frac{3}{3}$ $\frac{3}{5}$
4. Pour A into B: $\frac{1}{3}$ $\frac{5}{5}$

Problem 2.

What is the result of updating the edge (u,v) when the est[u], est[v] and weight(u,v) are given as follows?

Figure 1: Bellman Ford update subroutine

```
def update(u, v, weights, est, predecessor):
    if est[v] > est[u] + weights(u,v):
       est[v]=est[u]+weights(u,v)
       predecessor[v]=u
       return True
    else:
       return False
```

a) est[u] = 7, est[v] = 11, weight(u,v) = 3

Solution: est[v] is updated to 10

b) est[u] = 15, est[v] = 12, weight(u,v) = -3

Solution: est[v] is not updated

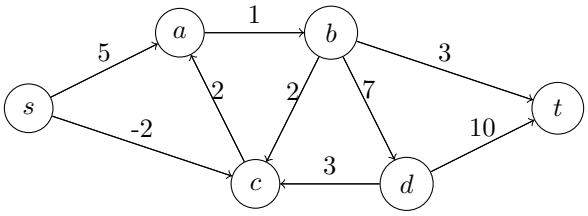
c) est[u] = 12, est[v] = 14, weight(u,v) = 3

Solution: est[v] is not updated

Problem 3.

Run Bellman-Ford on the following graph using node s as the source. Below each node u, write the shortest path length from s to u. Mark the predecessor of u by highlighting it or making a bold arrow.

```
def bellman_ford(graph, weights, source):
    est={node:float('inf') for node in graph.nodes}
    est[source]=0
    predecessor={node: None for node in graph.nodes}
    for i in range(len(graph.nodes)-1):
        for(u, v) in graph.edges:
            update(u, v, weights, est, predecessor)
    return est, predecessor
```



Solution:
predecessor : {'s':None, 'a':'c', 'b':'a', 'c':'s','d':'b', 't':'b'}
est : {'s':0, 'a':0, 'b':1, 'c':-2, 'd':8', 't':4}

Problem 4.

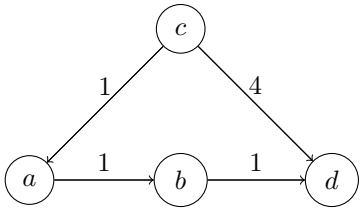
State TRUE or FALSE for the following statements:

a) If (s, v_1, v_2, v_3, v_4) is a shortest path from s to v_4 in a weighted graph, then (s, v_1, v_2, v_3) is a shortest path from s to v_3

Solution: True. Assume for the sake of contradiction that there is a path P from s to v_3 whose weight is lesser than (s, v_1, v_2, v_3). Then we can find a path from s to v_4 by combining P with the edge (v_3, v_4) whose weight is lesser than (s, v_1, v_2, v_3, v_4). This contradicts the fact that (s, v_1, v_2, v_3, v_4) is a shortest path from s to v_4.

b) Let P be a shortest path from some vertex s to some other vertex t in a directed graph. If the weight of each edge in the graph is increased by one, P will still be a shortest path from s to t.

Solution: False.



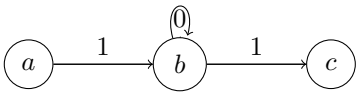
Consider the graph given above. The shortest path from c to d is (c,a,b,d) which is of weight 3. However, if the weight of each edge is increased by 1, the shortest path from c to d would be (c,d) of weight 5.

c) Suppose the update function is modified such the est[v] is updated when est[v] $\geq$ est[u] + weight(u,v) instead of strictly greater than. The est values of all nodes at the end of the algorithm would still give the shortest distance from the source.

Solution: True.

d) Suppose the update function is modified such the est[v] is updated when est[v] $\geq$ est[u] + weight(u,v) instead of strictly greater than. We can still find the shortest path from the source to any node using the predecessors using the new algorithm.

Solution: False.



Consider the graph given above. Let a be the source in the execution of Bellman Ford algorithm. Updating the edge will result in est[b] = 1 and predecessor[b] = a. However, updating the edge (b,b) will result in predecessor[b] = b according to the new update algorithm. Therefore, it would not be possible to find the shortest path from a to b using the predecessors.