DSC 40B - Discussion 07

Problem 1.

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 2.

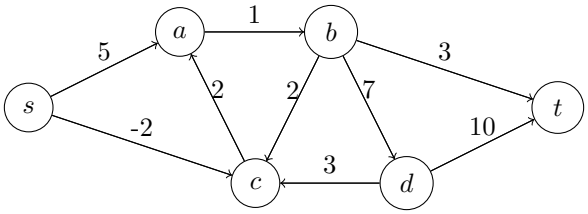
Run Bellman-Ford on the following graph using node s as the source. Below each node u, write the shortest path length from s to u. Mark the predecessor of u by highlighting it or making a bold arrow.

```
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
```

Edge ordering convention. In each iteration (pass), perform `update(u, v)` exactly as listed below. Iteration pass uses this exact edge order:

$$(s,a),(s,c),(c,a),(a,b),(b,c),(b,d),(d,c),(b,t),(d,t).$$

Use this same edge order for every pass.



Solution:

Solution: Using the required edge order:

$$(s,a),(s,c),(c,a),(a,b),(b,c),(b,d),(d,c),(b,t),(d,t).$$

Pass 1 updates:

$$a:\infty\to 5\to 0,\ c:\infty\to -2,\ b:\infty\to 1,\ d:\infty\to 8,\ t:\infty\to 4.$$

Pass 2: no values change, so the algorithm has converged.

Final values:

- 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

Notes:

- The edge ordering does not affect Bellman-Ford correctness.
- The edge ordering can affect how many passes are needed before convergence.
- If multiple shortest paths exist, different orderings can yield different predecessor trees.

Problem 3.

Follow-up: Bellman-Ford example from Lecture 14 slide 46.

Consider the directed weighted graph from Lecture 14 (slide 46) with source node v_1. The weighted edges are:

$$(v_1,v_2):2,\ (v_2,v_3):-1,\ (v_3,v_4):1,\ (v_4,v_5):3,\ (v_5,v_6):2,\ (v_1,v_7):7,\ (v_7,v_6):0,\ (v_7,v_5):3,\ (v_5,v_7):-1.$$

Edge ordering convention. In each iteration (pass), perform `update(u, v)` exactly as listed below. Iteration pass uses this exact edge order:

$$(v_3,v_4),\ (v_1,v_2),\ (v_2,v_3),\ (v_7,v_6),\ (v_5,v_7),\ (v_7,v_5),\ (v_4,v_5),\ (v_5,v_6),\ (v_1,v_7).$$

Answer the following:

a) How many full passes are needed until distances no longer change?

b) What are the final `est` and `predecessor` dictionaries?

c) Briefly explain why this ordering is slower.

Solution:

Solution: Pass-by-pass updates under the required order:

- Pass 1: $v_2 \leftarrow 2$, $v_3 \leftarrow 1$, $v_7 \leftarrow 7$.
- Pass 2: $v_4 \leftarrow 2$, $v_6 \leftarrow 7$, $v_5 \leftarrow 5$.
- Pass 3: $v_7 \leftarrow 4$ (via $v_5 \rightarrow v_7$).
- Pass 4: $v_6 \leftarrow 4$ (via updated $v_7 \rightarrow v_6$).
- Pass 5: no values change (converged).

So the first no-change pass is pass 5 (equivalently, all final values are already fixed by the end of pass 4).

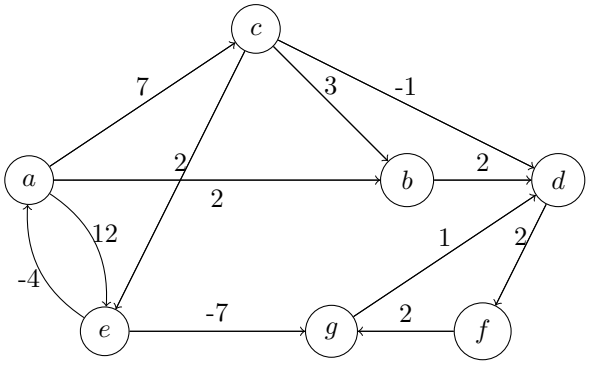
Final values:

- est: $v_1 : 0,\ v_2 : 2,\ v_3 : 1,\ v_4 : 2,\ v_5 : 5,\ v_6 : 4,\ v_7 : 4$
- predecessor: $v_1 : \texttt{None},\ v_2 : v_1,\ v_3 : v_2,\ v_4 : v_3,\ v_5 : v_4,\ v_6 : v_7,\ v_7 : v_5$

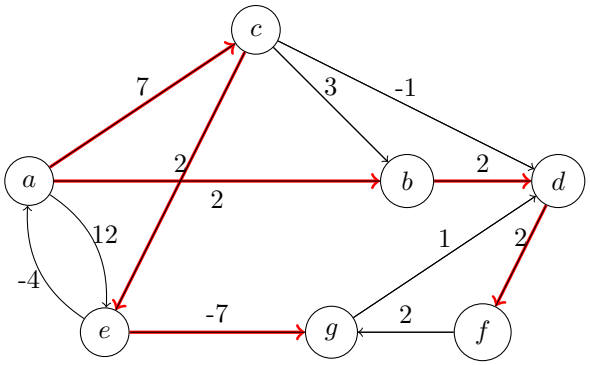
Why slower: this edge ordering is not aligned with the long shortest-path chain from v_1 to v_6. Improvements found later in a pass often cannot propagate until the next pass, so information moves more slowly. If we used an order that follows the chain direction (topologically along that path), the same information could propagate in far fewer passes (often around 1–2 passes for this type of example).

Problem 4.

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



Solution:



est = { 'a': 0, 'b': 2, 'c': 7, 'd': 4. 'e': 9, 'f': 6, 'g': 2 }

b) Dijkstra's algorithm found the wrong path to some of the vertices. For just the vertices wherethe wrong path was computed, indicate both the path that was computed and the correct path.

Solution:
Computed path to d is a,b,d but shortest path is a,c,e,g,d.
Computed path to f is a,b,d,f but shortest path is a,c,e,g,d,f.

c) What single edge could be removed from the graph such that Dijkstra's algorithm would happen to compute correct answers for all vertices in the remaining graph?

Solution: (e,g)