DSC 40B - Homework 08

Due: Monday, November 21

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer. Homeworks are due via Gradescope at 11:59 p.m.

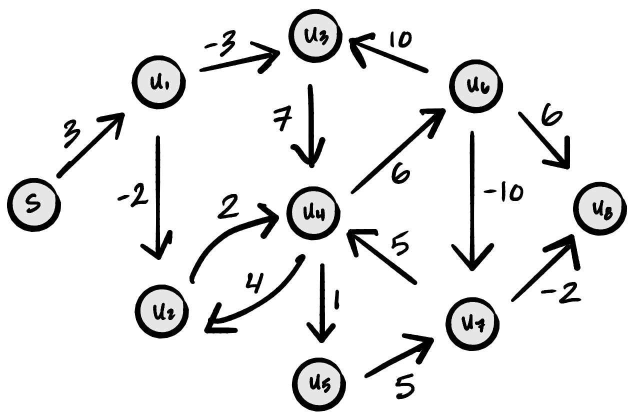
This homework is somewhat shorter than a typical homework due to the exam.

Note: because of the ongoing strike, this homework is graded by *completion*. This means that if you make a reasonable effort on a problem (and show your work where required), you'll get full credit even if your answer is incorrect.

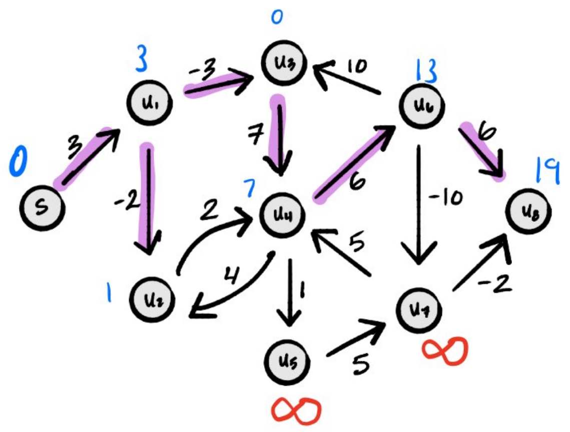
Problem 1.

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. You can assume that `graph.edges` produces the graph's edges in the following order:

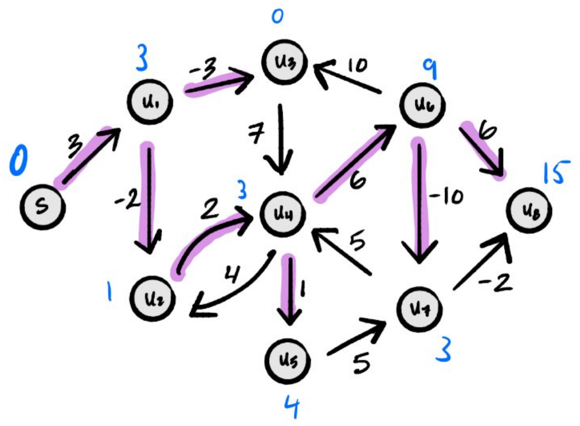
$$(u_2, u_4), (u_7, u_8), (u_6, u_3), (s, u_1), (u_1, u_2), (u_4, u_5), (u_1, u_3),$$
$$(u_3, u_4), (u_6, u_7), (u_4, u_6), (u_5, u_7), (u_7, u_4), (u_4, u_2), (u_6, u_8)$$



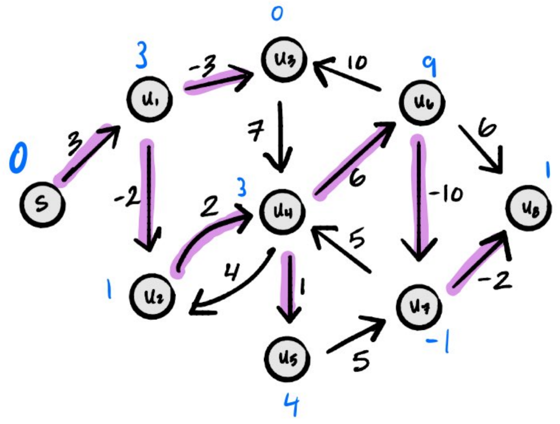
First Iteration of outer loop



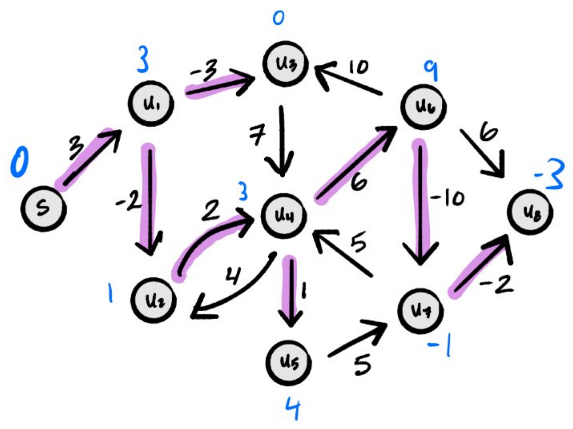
Second Iteration of outer loop



Third Iteration of outer loop



Fourth Iteration of outer loop



Ends there as in each iteration no points are updated anymore.

Problem 2.

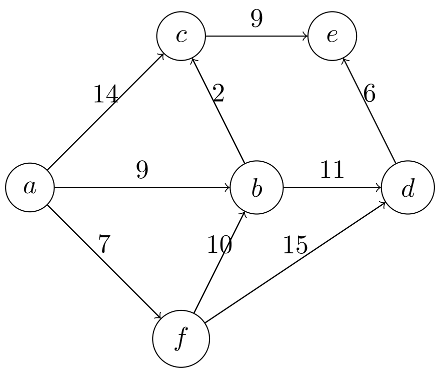
Recall that the Bellman-Ford algorithm (with early stopping) will terminate early if, after updating every edge, no predecessors have changed. Suppose it is known that the greatest shortest path distance in a graph $G = (V, E)$ has $\Theta(\sqrt{V})$ edges. What is the worst case time complexity of Bellman-Ford when run on this graph? State your answer using Θ notation.

Soln

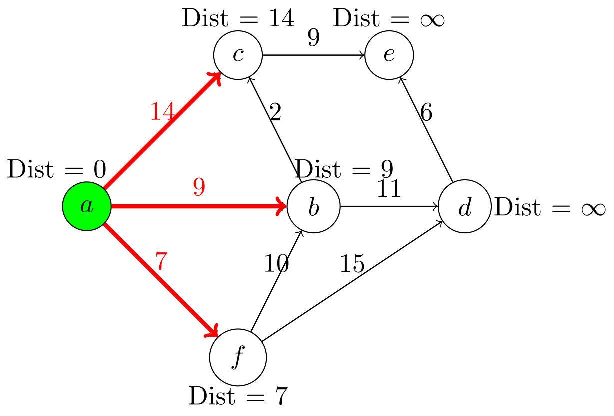
The worst case time complexity of Bellman ford in this case is still $\theta(VE)$ this is because, even though the length of the shortest path is known as $\sqrt{V}$. Bellman Ford looks at each edges arbitrarily, which means in the worst case scenario, where the edges appear in an order which causes very little updates each iteration of the outer loop. It can still take up to the maximum number of required updates which is exactly $(V - 1)E$ total updates. Therefore giving us the still worst case time complexity of $\theta(VE)$.

Problem 3.

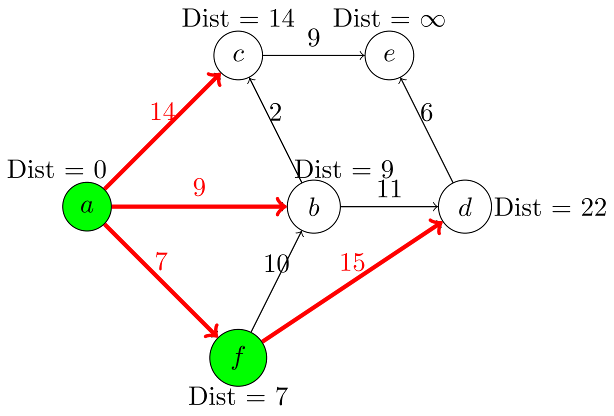
Run Dijkstra's Algorithm on the following graph using node a as the source. Below each node u, write the shortest path length from a to u. Mark the predecessor of u by highlighting it or making a bold arrow.



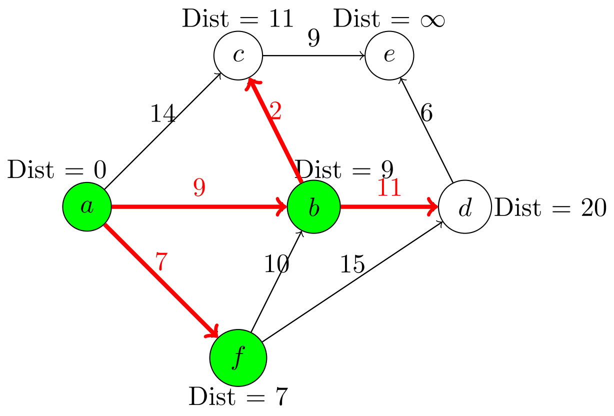
Iteration 1



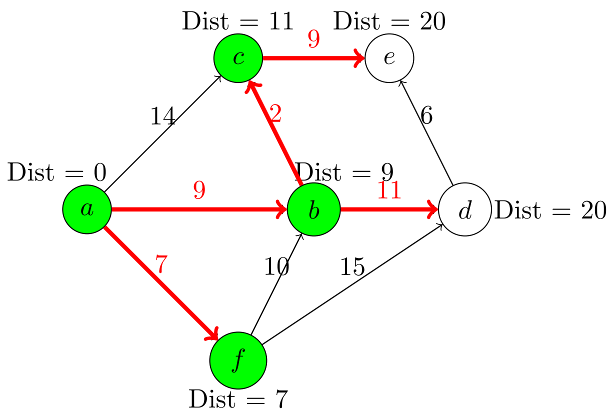
Iteration 2



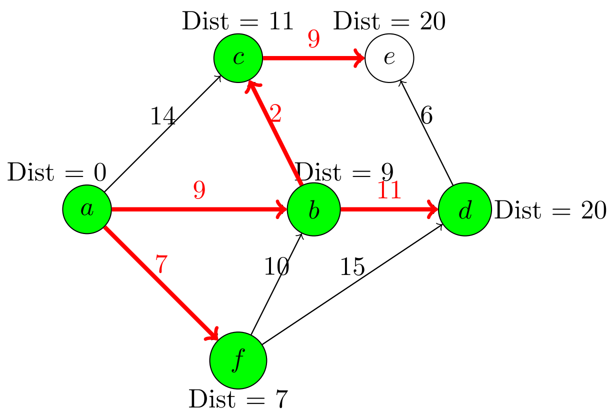
Iteration 3



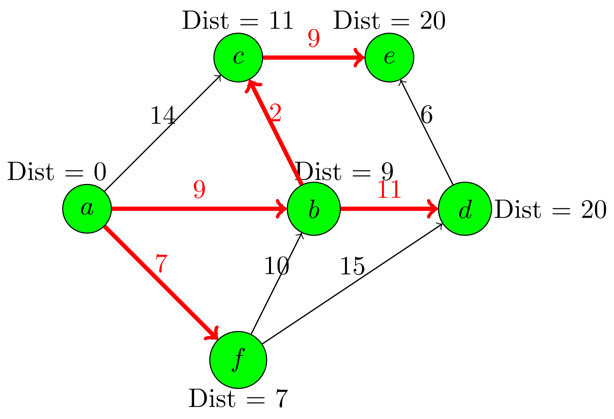
Iteration 4



Iteration 5



Iteration 6



Problem 4.

True or False. Suppose $G = (V, E, \omega)$ is a weighted graph for which all edges are positive, except for those edges of a node s which may or may not be negative. If Dijkstra's algorithm is run on G with s as the source, the correct shortest paths will be found. Assume that the graph does not have any negative loops. Justify your answer.

soln

It is **True.**
This is because, Dijkstra's algorithm doesn't work with negative edges as it has a property that it does not check all the possible edges and rules out edges based on the fact that if it already takes a longer distances to get to one node, the node in questions (node we are trying to get the shortest path too), cannot have any other possible shorter path. However, in the presences of negative nodes this is possible hence the reason that Dijkstra's algorithm requires the assumption that all path weights are positive.

However, if the negative weights are located ony around the source, Dijkstra's algorithm will check all the edges around the source, hence preventing any possibility of a path length to decrease in a further path, therefore allowing Dijkstra's algorithm to still get the correct shortest path.