Homework 7: Q3

Name: Xinkai Lin, xinkaili

*Don't forget to input your list of collaborators and sources on **AutoLab**.*
Please submit this file as a PDF.

1 Algorithm Idea

In this algorithm, since the giving input is a Directed Acyclic Graph, I will first sort it in a topological order. Then I will create an array and set the starting node to be 0, all the other nodes will be initialized to be infinity. Also, I will create another array initialized to be 0, and it will later save the parent node of the shortest path. Then, for every vertex u in topological order, I will loop through all the adjacent v node to u. If the weight at v is greater than the weight of u plus the weight of edge, update its weight, and do the same for all the vertex. Also, save u in the other array that I create to be the parent of v. When the algorithm end, all the node will set to be the shortest weight, then I will use the array I create to get the shortest path.

2 Algorithm Details

```
Input: Given a Directed Acyclic Graph (or DAG) GG with arbitrary weights on the edges
Sort G in a topological order
shortest = {}
dist = []
path = []
for all node in G do
    dist[i] = infinity
    path = infinity
end for
set dist[s] = 0
for each vertex u in G
    for all the neighbor node v that connected to u do
        if (dist[v] > dist[u] + w (u, v)) do
            dist[v] = dist[u] + weight (u, v)
            path[v] = u
        end if
    end for
end for
while path[v] is not s do
    add all path[v] to shortest
end while
```

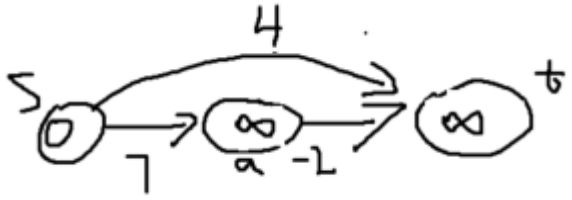
reverse shortest to s to t
return shortest

3 Proof of Correctness Idea

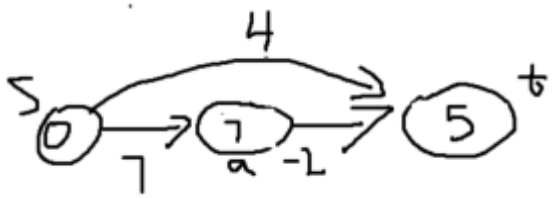
This algorithm is correct because when the algorithm end, it will return the shortest path from s to t. First of all, I will sort the graph as topological order, which mean that all nodes only going forward. Then for every adjacent node v I will compare its own weight with the weight of u and the edge weight, if weight of v is greater, then update the weight, else do nothing. Until we loop thought all the nodes. This way, when the loop end, we will have the node update with the shortest weight. Then, since I have save their parent in all the nodes, I can simply use a while loop to print out the parent which will be the path that came from. Which will be the shortest path, so my algorithm is correct.

4 Proof Details

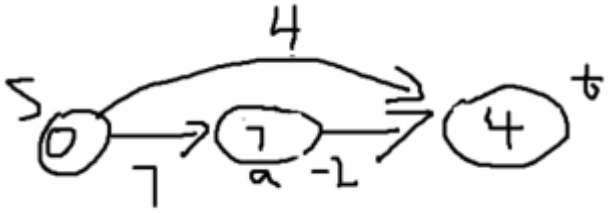
Consider the following example, s, a, t.



from s to t, where we set s to be 0, and others to be infinity.
Now, loop through all the adjacent nodes. when we go from s to a, we can see that 7 is less then infinity, so we update a to be 7, and from the a to t, we can now see 7-2 is less then infinite, so we update t to be 5 as follow.



Finally, we are going from s to t, right now we have 5 as weight at t. and we can go from s to t its 4, and 4 is less than 5, so we update t to be 4.



Then we have our shortest path which is from s to t.

5 Runtime Analysis

At the beginning of this algorithm, I will sort the graph as in topological order. We have proven that in class that this is a runtime of O(m+n). Next, I will initialize all the node as infinity with a for loop, this will take up to O(n) time. Then, I have loop through all the nodes adjacent node, since this is graph will be going forward, so it will actually be $\sum_u n_u$ which take up to O(m) time. Finally, there is a while loop, which takes O(n) to reverse the order. Thus, the final runtime for this algorithm is O(m+n) + O(n) + O(m) + O(n), which is O(m+n).