Homework 6

Problem 1:

Part 1:

Let “a” be the source vertice of the negative edge and “b” the destination vertice so the negative

Remove edge a 🡪 b from G and create a new graph G’ = {V’, E’)

Run Dijkstra’s algorithm on (G’, s)

Run Dijkstra’s algorithm on (G’, b)

If the total weight of the 2 runs of Dijsktra is less than the negative of weight of a 🡪 b

Return “YES CYCLE”

Else

Return “NO CYCLE”

Part 2:

Let “a” be the source vertice of the negative edge and “b” the destination vertice so the negative

Remove edge a 🡪 b from G and create a new graph G’ = {V’, E’)

Compute the dist(s, t) by running Dijkstra’s algorithm twice on G’

Once starting at s to compute both dist’(s, t) and dist’(s, a)

Once starting from v to compute dist’(b, t)

Return all distances from the runs of Dijkstra’s algorithm

Problem 2

BellmanFord(Graph graph, int src)

Let V be graph.V

Let E be graph.E

Set dist[] to be the size of V

Initialize the first value of dist[0] to 0

Initialize all other values of dist[] to infinity

For j = 1 to V-1

For i = 1 to E

Let u be graph.edge[j].source

Let v be graph.edge[j].destination

Let weight be graph.edge[j].weight

Let temp be dist[u] + weight

if temp < dist[v] and temp > 0

        dist[v] = temp

if temp < dist[v] and temp < 0

        dist[v] = negative infinity

return dist[v]

Problem 3

topologicalSortBFS( Graph, numVertices)

Create a new array “in-degrees” to store the in degrees for each vertice

Create a queue “q” for all vertices with no in bound edges

//Iterate over the Graph to populate the “in-degrees” array

For i < number of Vertices

If in-degrees[i] is 0 // no inbound edges

Add vertice to “q”

End if

End For

Create a new array “sorted” to store the topological sort

While q is not empty

Pull the vertice “v” from q

Add v to “sorted”

Remove v from the graph

Decrease the indegree of the other ends of the v by one

Loop over the adjacent vertices

If in-degrees is 0

Add vertice to “q”

Return “sorted”

Problem 4

Part 1

There will not be remaining edges in the graph when the algorithm terminates. The algorithm picks Vj before Vi so Vi is still in Gj-1 but then (Vi, Vj) Gj-1 so In(Vi) was not empty hence we have a contradiction.

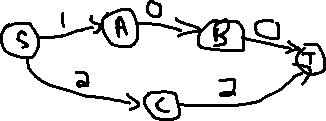
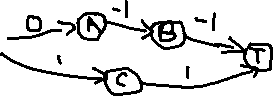
Part 2

Yes, every vertex remaining in G’ will be part of some directed cycle since each vertex will have all least one incoming edge.

Problem 5

The shortest path in Graph G is 2 from S🡪C🡪T.

In the modified Graph G’, the shortest path is 1 from S🡪A🡪B🡪T.

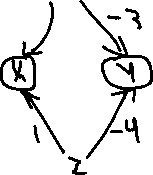
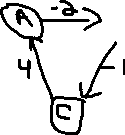


Problem 6

Part 1



Graph G



It doesn’t matter which of the vertices of the graph G you choose as your source vertex because you will not be able to get to all of the other five vertices, this is why you need a dummy source.

Only graphs that don’t have a negative cycle and that all vertices are reachable by all other vertices will work.

Part 2

I don’t know