Abel Marin

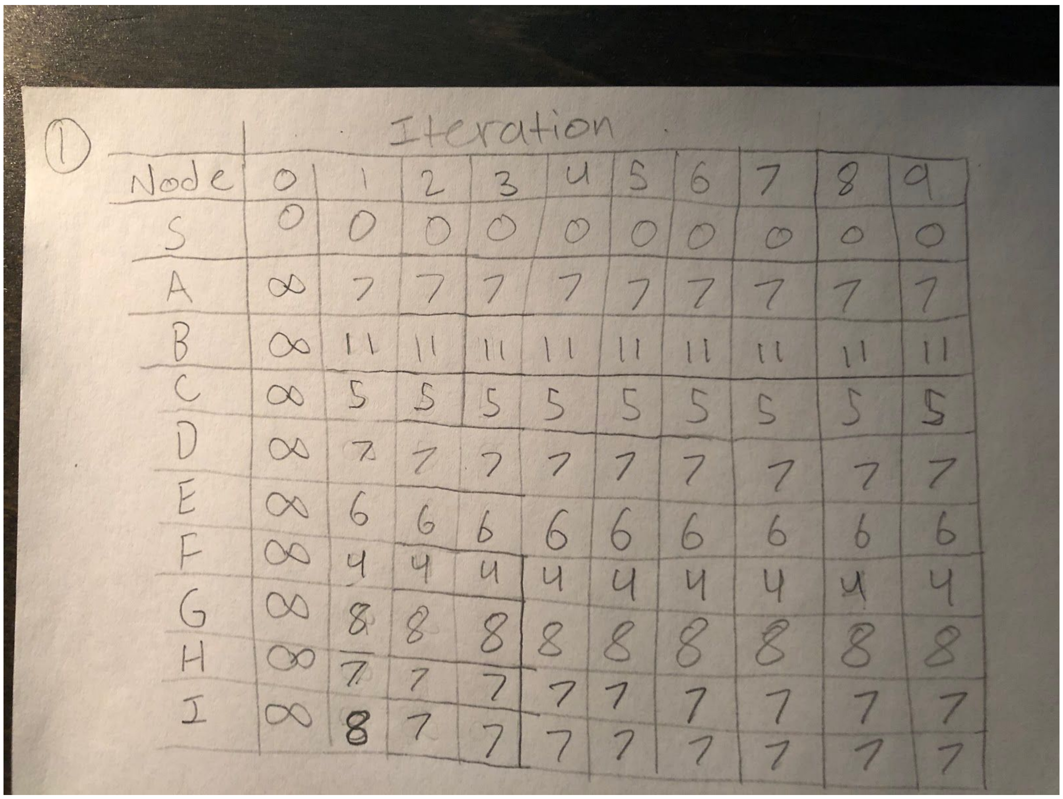
Professor Kanj

CSC 321

November 11, 2019

Homework #4

1. Problem 1



(1)

	Iteration									
Node	0	1	2	3	4	5	6	7	8	9
S	0	0	0	0	0	0	0	0	0	0
A	∞	7	7	7	7	7	7	7	7	7
B	∞	11	11	11	11	11	11	11	11	11
C	∞	5	5	5	5	5	5	5	5	5
D	∞	7	7	7	7	7	7	7	7	7
E	∞	6	6	6	6	6	6	6	6	6
F	∞	4	4	4	4	4	4	4	4	4
G	∞	8	8	8	8	8	8	8	8	8
H	∞	7	7	7	7	7	7	7	7	7
I	∞	8	7	7	7	7	7	7	7	7

2. Counterexample: suppose you have nodes A, B, and C with edges going from A->C, A->B, B->C with weights of 2, 5, and -10 respectively. We also have a large constant C = 1000. We add 1000 to each edge weight and get AC = 1002, AB = 1005, and BC = 990. We then run Dijkstra's Algorithm in order to find the shortest path from A to C. While the correct answer should be A -> B -> C, by

using the large constant method we get a shortest path of A -> C. Therefore this approach is not correct.

3. Pseudocode: Given a graph g and positive edge lengths L

<u>shortestCycle:</u>

```
shortestCycle = infinity;
for each vertex u ∈ g do
        dist[ ] = dijkstra(g, L, u);
        for each vertex v ∈ g do
                if (v, u) ∈ E then
                        shortestCycle = min(shortest, dist[v] + length(u,v));
if shortestCycle == infinity then     return ('Graph is acyclic');
else                                  return shortestCycle;
```

Explanation: We begin by initializing shortestCycle to infinity. We then have a loop which runs through all vertices of the graph. This loop runs Dijkstra's Algorithm on the current vertex u. Then a nested loop runs through all other vertices v and this checks to see whether or not there is an edge (v, u), if so then this would complete a cycle. We update shortestCycle if the new cycle is shorter. We then exit the loop. If shortestCycle is infinity then we return 'Graph is acyclic. Otherwise we return shortestCycle.

Running Time Analysis: We are running Dijkstra's Algorithm V times. If we use the array implementation of Dijkstra (which has a running time of $O(V^2)$) then the running time would be $O(V^3)$.