WA10 - Assignment Group 16

Sravani Kirla - skirla
Matthew Heil - heilx069
Terrence Hanrahan - hanra028

1.
Algorithm:

One such algorithm to find that may find a negative cycle in the graph is simply Bellman-Ford, with a simple change. The change comes in the lines of code of:

```
If v.d > u.d + w(u,v)
    Return False
```

These two lines run during a for loop that iterates over all of the edges and checks if any negative cycle exists. Instead of returning False, indicating at least one negative cycle exists, we can run a loop that iterates over all previous nodes until u is reached. It is important to note that v may or may not be part of the negative cycle, but u is. Thus, we start at u and go through the previous node of u, then the previous node of the previous node, etc. until we reach u again. And during each iteration of this loop we add those vertices to a list. Once u is reached again, that indicates the end of the negative cycle so we can return the list of vertices. The runtime for this algorithm is O(VE) because the only change is adding in a while loop that may run O(E) or less (down to O(1), depending on cycle length), inside of a for loop with runtime O(E). This creates a max runtime of O(E^2), which is still runtime of O(VE).

2.

```
DIJKSTRA(G,reliability,source)
1  INITIALIZE_SINGLE_SOURCE(G,source)
2  Q = G.V
3  while Q not Empty
4      u = EXTRACT_MAX(Q)
5      for each vertex v in u.adj
6      RELAX(u,v,reliability(u,v))

INITIALIZE-SINGLE-SOURCE(G,source)
1      for each vertex v in G.V
2          v.r = -∞ //Negative infinity
3          v.prev = NIL
4      source.r = 1

RELAX(u,v,reliability(u,v))
1      if v.r < u.r * reliability(u,v)
2          v.r = u.r  * reliability(u,v)
3          v.prev = u
```

3.

WA10 - Assignment Group 16

	A	**B**	**C**	**D**	**E**
A	0	6	3	6	5
B	4	0	7	10	9
C	1	3	0	5	4
D	-4	0	-3	0	-1
E	∞	∞	∞	∞	0