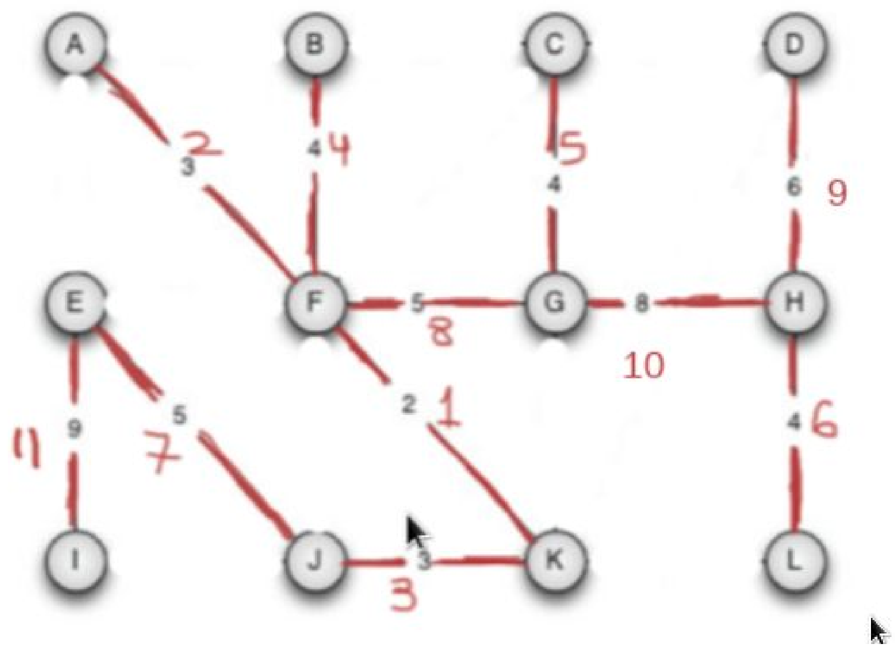
Andrew Letz

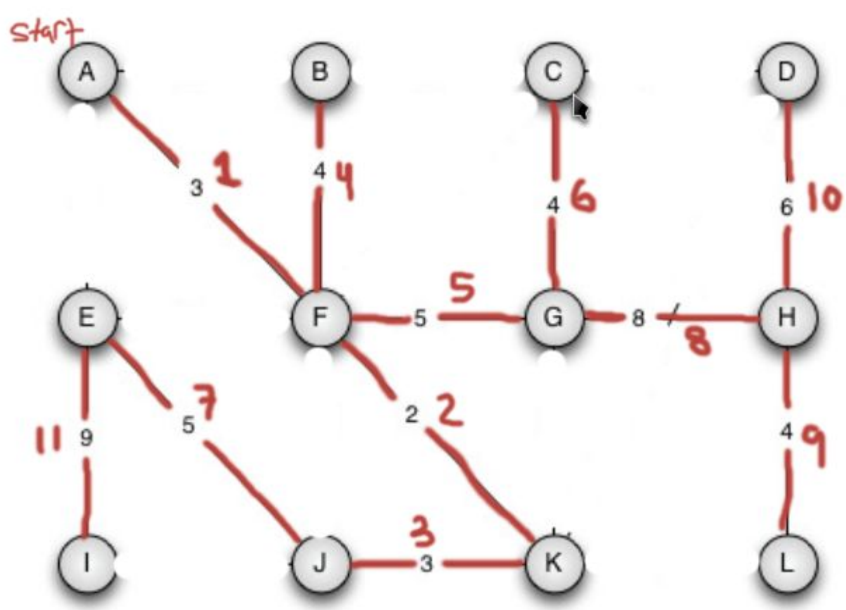
CIS 315

Professor Wilson

Assignment #3

Question 1





Question 2

You can solve this problem with a modified version of Dijkstra's algorithm.

```
def highestBandwidth(G, B, s, t):
      Q = Queue()
      for v in G:
            v.bw = 0
            Q.enqueue(v)
      s.bw = inf // bandwidth of start node is infinite
      while !Q.isEmpty():
            u = Q.removeMax() // remove max based on bandwidth
            for v in u.adj:
                  if v in Q:
                        if (min(B[u, v], u.bw) > v.bw):
                              v.bw = min(B[u, v], u.bw)
      return t.bw
```

Question 3

Similarly to question 2, Dijkstra's algorithm provides a good basis for solving this problem.

```
def mostReliable(G, r, s, t):
      Q = Queue()
      for v in G:
            v.r = 0
            Q.enqueue(v)
      s.r = inf
      while !Q.isEmpty():
            u = Q.removeMax() // remove max based on reliability
            for v in u.adj:
                  if v in Q:
                        // need to multiply to get correct value
                        v.r = max(u.r * r[u, v], v.r)
      return t.r
```

Question 4

a - The squared matrix gives information about the paths of two edges in graph G. In the case of $M^2(i, j) = 1$, this means there exists a path of 2 edges (paying no attention to weight) from vertex i to j. If it is equal to 0, there is no path of two edges between the vertices.

c - In this case, k represents the minimum weight path made up of two edges between vertices i and j. This implies that at least one such path exists.

Question 5

Due to the fact that the algorithm no longer updates shortest-path weights after m iterations, we can tell it to stop once the possibility for change is gone. We cannot do this in exactly m iterations because it is not known beforehand, but we can in m + 1.