Assignment – 2: Shortest Paths

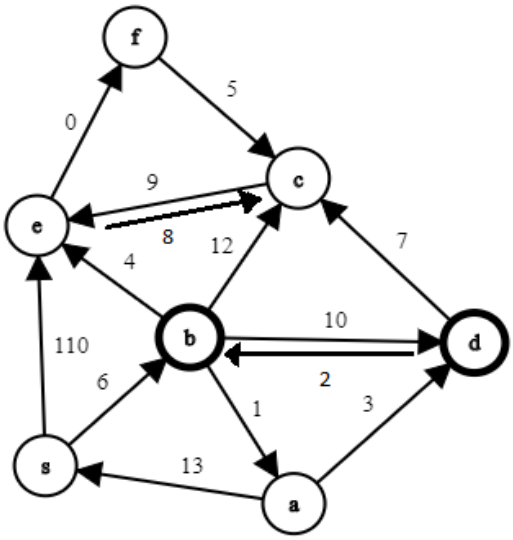
Submitted by – Farhan Ishmam, 180041120, CSE – A.

Date – 06-OCT-20

Problem – 1

None

Problem – 2(a)



Problem – 2(b)

s	a	b	c	d	e	f
0	7	6	15	10	10	10

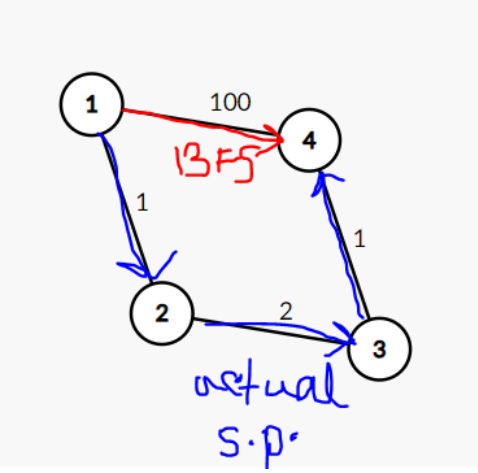
Problem – 2(c)

s -> b -> a -> d -> e -> f -> c

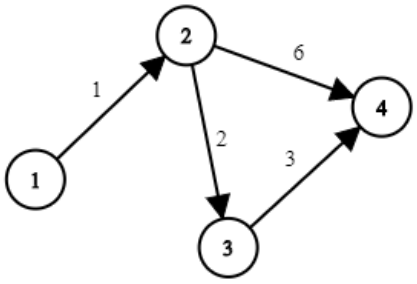
Problem – 3

No, we can't find the shortest path in a weighted graph without modifying BFS.

Using vanilla BFS, we perform level-order traversal in our graph. If it were an unweighted graph i.e. all the weights were the same then the lower level would mean the shorter path. But in case of weighted graph, a direct neighbor might not be the shortest path if the weight is too high. Let's look at the graph in the example where we find the shortest path from 1 to 4. For vanilla BFS, the path weighting 100 would be the shortest path since it has the lowest level. But the actual shortest path is 1 -> 2 -> 3 -> 4 which weighs only 4. Thus unmodified BFS won't work to find shortest path for weighted graph.



Problem – 4(a)



The given graph follows Starney's Algorithm.

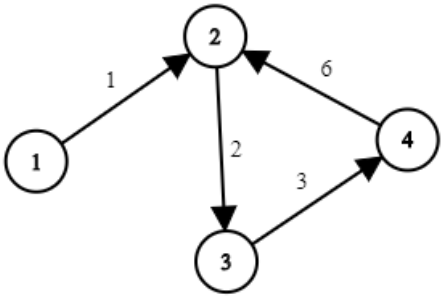
If we negate all the edges then the weights become -1, -2, -3 and -6. Let's take 1 as source node. Then by applying bellman ford we get:

1	2	3	4
0	-1	-3	-7

By negating the distances, we get 0, 1, 3 and 7. In case of the path from 1 to 4, we can clearly see that the longest path has been taken which is from 1 -> 2 -> 4 and the shortest path, 1 -> 2 -> 3 -> 4 has been avoided.

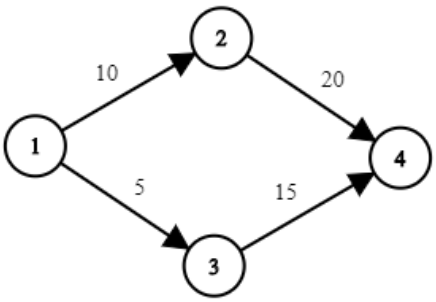
Problem – 4(b)

No, it won't work for all the graphs. Starney's algorithm is based on the negating the Bellman-Ford Algorithm. Bellman-Ford can detect negative weight cycles but not solve them. Hence, for Starney's algorithm, it can detect positive weight cycles but not solve them. If there's a positive weight cycle then the graph won't find the longest distance. Of course, it is to be noted that a positive weight cycle is like an anomaly because the distance increases every time we traverse the cycle.



The given graph has a positive weight cycle 2 -> 3 -> 4. By negating these weights we get a negative weight cycle. Then if we run Bellman-Ford it'll detect this negative weight cycle but not solve it. After (v-1) iterations the algorithm will stop. Hence, the longest distance cannot be found in this case.

Problem – 5



PERT charts are DAG or directed acyclic graphs. The given graph is an example of a PERT chart. Let's say 1 is starting point and 4 is the ending point and the weights represent the time to complete the tasks. To reach 4, we need to complete both 2 and 3 but in minimum time. In this graph if we take route 1 -> 2 -> 4 then it will take 30 units. By this time task 3 will also be completed since it takes 20 units to reach 4 using the route of 3. Hence, by finding the longest path we can solve this problem. To do so, we find the longest path on the topologically sorted vertices.

The step by step implementation is given below:

i) We pick the source node and topologically sort all the nodes
ii) The arrays are to be initialized with negative infinity instead of positive infinity.
iii) Then we can traverse the ordered graph using BFS.
iv) The distance and parent will be updated as the graph is traversed, just like Djistra's Algorithm. But in this case distance and parent will be updated only if the new distance is GREATER than the older one.
v) End when all the nodes are visited.

Both complexity of topological sort and BFS is O (V+E). So, we can easily say that the complexity will be O (V+E) for this algorithm. This is more efficient than running Djistra's Algorithm for longest path.