Quiz 6

Problem 1

1) clearly, the weight of each edge is 1 and there are m+1 edges of the path. So the shortest path is m+1

2) for all the paths, since every step we move forward to next stage, therefore, the weight of all paths in this graph is same. So they are all shortest paths

3)

a. Main idea: we use dynamic programming to solve this problem. Assuming $T_{v_{i,j}}$ represents the number of distinct paths from s to $v_{i,j}$. By using dynamic programming, the recursive part is written as: $T_{v_{i,j}} = \sum T_{v_{i-1,p}}$ $(where\ v_{i-1,p}\ are\ the\ ajacent\ node\ of\ v_{i,j})$

b. Pseudocode:

Input: graph G:

Output: T

Set the number of all node at the first stage as 1

For all nodes j in stage 1:

$T_{v_{1,j}} = 1$

For i in range(2, m):

For j in n (number of vertices in stage i):

$T_{v_{i,j}} = \sum T_{v_{i-1,p}}$ $(where\ v_{i-1,p}\ are\ the\ ajacent\ node\ of\ v_{i,j})$

For j in stage m:

$T = \sum T_{v_{m,j}}$

---

c. proof:

using dynamic programming, every step we consider the previous step, that is total path of previous node. Therefore, by recursively iterating to the stage m, we can get the number of all the distinct paths

d. time: since we are basically looping over all the edges, the time complexity if O(E)

Problem 2

a. Main idea: here we run BFS twice to remove the vertices that does not lie on the shortest path. Then we run the algorithm discussed in the Problem 1 to find the number of distinct shortest path.

More specifically, we first run BFS starting from node s, and assign distance from s to all the node v, we define as $d_1(v)$. Meanwhile, $d_1(t)$ represents the distance from s to t, here we define $P= d_1(t)$. Then we run another BFS from node t, and assign distance from t to all the node v as $d_2(v)$. then we loop over all the vertices v, if $d_1(v) + d_2(v) != P$, we remove this node. Then we will have the graph exactly same as problem 1 and run the algorithm we designed in problem 1.

b. Pseudocode: (here we ignore the code of BFS)

Run BFS starting from s, to get $d_1(v)$, $P= d_1(t)$

Run BFS starting from t, to get $d_2(v)$

For v in node V:

    If $d_1(v) + d_2(v) != P$:

        Remove v

Run algorithm in Problem 1

---

c. proof:

If we run BFS from s, we will assign the distance from s to all the nodes. If this node is on the shortest path, then the distance from this node to t must be: shortest path distance – distance to s. Based on this, we run BFS again from t and remove the nodes that does not lie on the shortest path. The reconstructed graph will be same as the graph in problem 1

d. time: O(V+E)