20301357

Ans: to the question no: 03

we know

V= vertex and E= Edges.

So, for Problem 1 ~~and Problem 2~~

Inside the dijkstra function we see that there is one "for loop" and another one is "while loop". If we look we can see that inside the "while loop" there is two for loop according to my code.

Now,

for the "while loop": time complexity is, $\rightarrow O(V)$

again, inside the "while loop", first for loop's time complexity

In → O(V)

2nd "for loops" time complexity In → O(E)

∴ Therefore the time complexity

In → O(V+E)

NOW,

As, there two "for loops" are nested with the "while loop",

∴ the finial time complexity will be ⇒ O(V) * O(V+E) + O(V)

⇒ O(V² + VE) + O(V)

⇒ O(V²) [we are using Priority queue]

⇒ O(N²)

As N = number of (V) Places

Now, for the task 2,

there is 2 "for loop" and 2 "while loop". and if we look we can see, from 1 "while loop" there is also 2 "for loop" inside the loop.

So,

for the while loop the time complexity is $\rightarrow O(V)$

again, inside the while loop,

for first "for loop" the time complexity $\rightarrow O(V)$

" 2nd " " " $\rightarrow O(E)$

$\therefore$ therefore, time complexity is $\rightarrow O(V+E)$

NOW,

as these two "for loop" are nested with "while loop". So, the final time complexity will be $\Rightarrow O(V) \cdot O(V+E) + O(V)$

$\Rightarrow O(V^2 + VE) + O(V)$

$\Rightarrow O(V^2) \Rightarrow O(N^2)$

Now,

If the number of titan's in each road is exactly 1,

BFS algorithm can solve this Problem with O(N+M) time complexity.

Here N => Places
M => roads.

The Input for this will be,

```
3
1 0
2 1
 1 2 1
 5 6
 3 5 1
 1 2 1
 2 3 1
 1 4 1
 4 3 1
 2 5 1
```

For this, the output will be,

Problem 1 output:

0
1
2

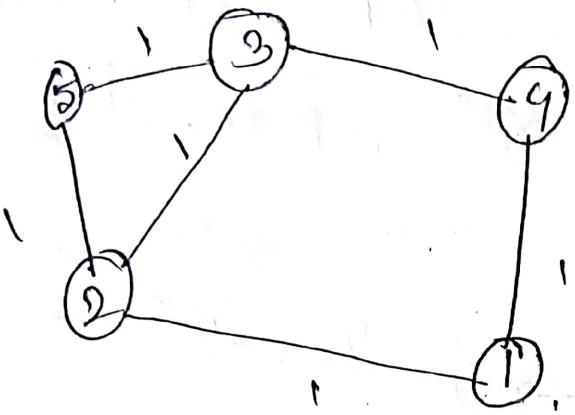
Problem 2 output:

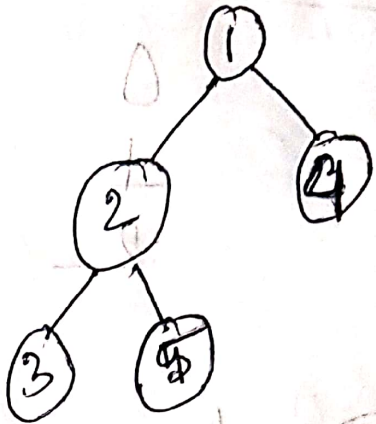
1
1 2
1 2 5

We know, Breadth first search algorithm will search for the lowest number of roads needed to reach the final goal place. For this, BFS function will need 3 arguments. The graph, starting place and final destination. Here for the given input, the

graph will be,

~~graph = [[2,4], [3,5], [5], [3], []]~~

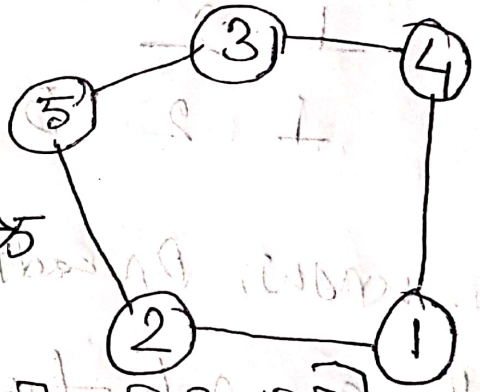




q ⇒ 1, 2, 4, 3, 5

dq → 1, 2, 4, 3, 5

∴ order → 1 → 2 → 4 → 3 → 5



∴ graph = [[2, 4], [3, 5], [5], [3], [4]]

As, there are n places and m roads, the time complexity will be O(N+M) as, there will scan for all the adjacent vertices.

We know, the time complexity depends on the number of edge and Vertices we have. The more E and V we have more time complexity will be needed to excute.

So, we are adding new edges connected to vertices each time we get one. and as, we can see on the Pseudocode that in one loop so it should be the complexity around O(n), but as we are adding new edges connected to vertices, so this n is proportional to (V+E). As it is traversing through all the vertices and edges.

∴ Time complexity = O(V+E)
= O(N+M)