Sub :

Day

Time : Date : / /

Ans: to the Que: No: 3

Given that,

there are N places = vertex

M roads = edges

According to my code,

there are 1 outer while loop which will gone through all vertex.

So the time complexity $\Rightarrow O(V)$

and there are two inner for loop. First for loop is for minimum distance (for all vertex)

time comp $\Rightarrow O(V)$

and 2nd one is for all vertex and their connected edges $\Rightarrow O(V+E)$

inner time comp $\Rightarrow O(V) + O(V+E) + O(1)$

$\Rightarrow O(V) + O(V+E)$

Time : Date : / /

∴ total $\Rightarrow O(V)\{O(V) + O(V+E)\}$

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

$\approx O(V^2)$

$= O(N^2)$ [N = all the places = V]

Part 2

the algorithm is kind of same.

in the modification the tracker is under 2nd inner for loop and it will not affect at all as its complexity $\Rightarrow O(1)$

and the last while loop which is just for the path

worst comp will be $\Rightarrow O(E)$ [gone through all the road]

∴ total complexity $\Rightarrow O(V)\{O(V) + O(V+E) + O(E)\}$

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

$\approx O(V^2)$

$\Rightarrow O(N^2)$

Sub :

Time : Date : / /

Part 3

if the number of titans in all road become 1 then we should not think about the weight at all. For those we just have to care about the shortest path as it will give me less titans).

⇒ So we can use BFS algorithm to find shortest path.

Pseudocode

⇒ ~~Algorithm~~ of BFS

~~visited = []~~

~~for~~ visited = queue = []

function (graph, visited, start, End)

Do visit (start-1)

queue.appened(start)

while queue:

pop(). from queue

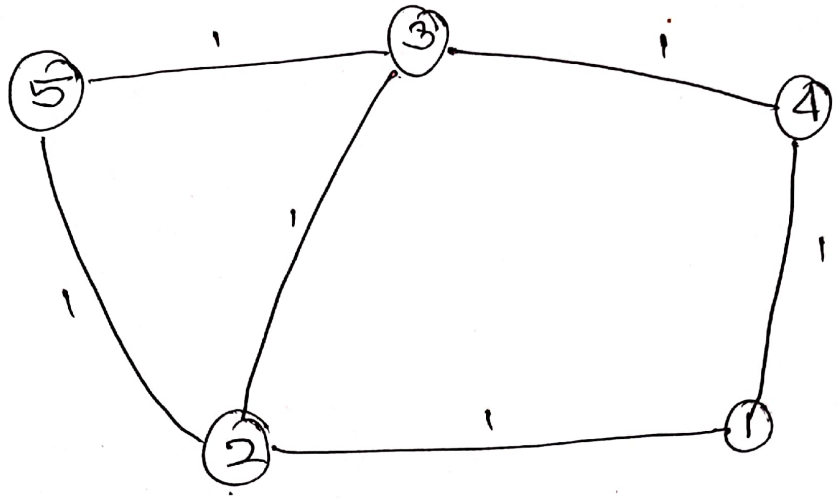
if popped item is end ! break

else visit (neighbour-1)

visited += [neighbour]

queue += [neighbour]

graph:



Start=1

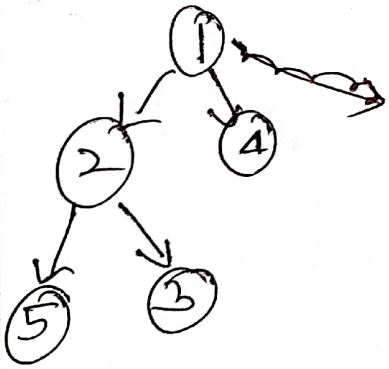
end=1

Simulation:

$1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 3$

So, we can say that from tree we getting shortest path $\Rightarrow (1,2,5)$

tree:



items $\Rightarrow 3$