Ans to the question no. 5

For bfs, we start from a node, mark it as visited, then mark all the neibours visited and add them to queue. Then fetch the next node from the queue and perform the same operation until queue is empty. So, this makes our time complexity O(E). Plus, since we add each node to the queue, it makes the time complexity O(V). So, the total time complexity becomes O(V+E). Here, V means vertecies and E means edges. and if one is larger from another the smaller can be seen as constant.

same logic can be applied to DFS. We visit each node and check each edges to dive into the depth of the graph. So, it makes dfs complexity O(V+E) too.

if we used adjancy matrix instead of list, we would have to check all the colmns connesponding to the the related row, which makes O(V). And we have to do it for all verteciey. therefore $O(V^2)$.

As the ~~gra~~ graph representing the map is unweighted, After using both DFS and BFS we get these results.

For BFS -> 1, 2, 3, 4, 5, 7, 11, 6, 12

For DFS -> 1, 2, 3, 4, 7, 11, 12

Here, we can say Gany would reach victory road first. As we can se he have to travel fewer cities. Because DFS goes further into the graph of and the victory road, it reaches first.