The time complexity of visiting V number of nodes is O(V) in both BFS and DFS.

If adjacency list is used, each node has its own list of all adjacent edges. We visit all the neighbours of each node linearly. If there are total E edges in the graph, then the time complexity is O(E). The total time complexity is O(E)+O(V)= O(V+E). This applies both to DFS and BFS alogorithms.

If the graph is represented as adjacency matrix, the time complexity is $O(V^2)$ since the matrix is VXV and we need to traverse through the entire rows of length V for V nodes to discover the edges. ~~Sim~~ This again applies to both DRS and BFS algorithms.

Gary who is using the DFS algorithm would reach the victory road first.

In a BFS algorithm, we put the node we reached in a queue and traverse through that node's neighbours and put them in the queue. If, for a certain node, there are no more unvisited adjacent nodes left, we are certain that node is completely explored and dequeue it and move on to the next node in the queue. This means that we need to visit more nodes before we reach the destination node.

However, in a DFS algorithm, once we reach a new node, instead of ~~to~~ visiting all its neighbours, we ~~explore~~ visit one of them. ~~The~~ This is my new node and the previous node is put in a stack to explore further later. We repeat this process untill we reach the destination node.