Implementing DFS using stack

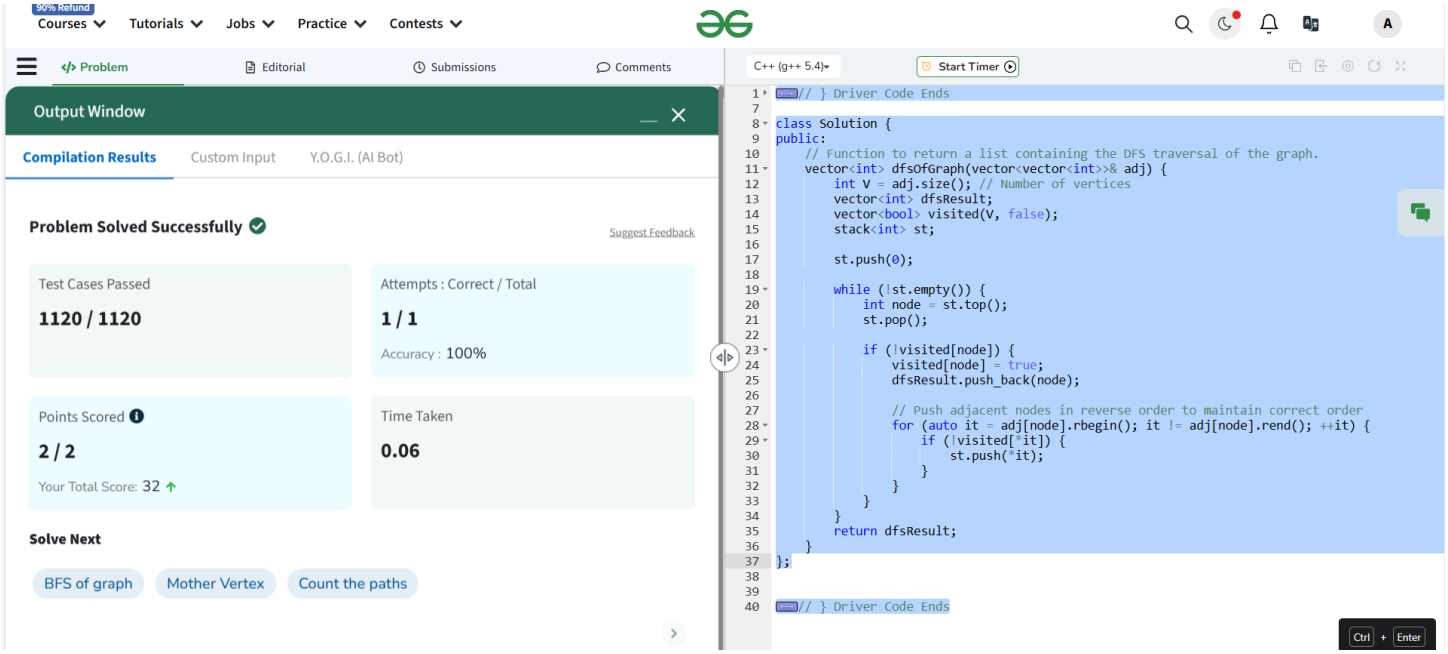
```
class Solution {
public:
    // Function to return a list containing the DFS traversal of the graph.
    vector<int> dfsOfGraph(vector<vector<int>>& adj) {
        int V = adj.size(); // Number of vertices
        vector<int> dfsResult;
        vector<bool> visited(V, false);
        stack<int> st;

        st.push(0);

        while (!st.empty()) {
            int node = st.top();
            st.pop();

            if (!visited[node]) {
                visited[node] = true;
                dfsResult.push_back(node);

                // Push adjacent nodes in reverse order to maintain correct order
                for (auto it = adj[node].rbegin(); it != adj[node].rend(); ++it) {
                    if (!visited[*it]) {
                        st.push(*it);
                    }
                }
            }
        }
        return dfsResult;
    }
};
```


90% Refund
Courses
Tutorials
Jobs
Practice
Contests
A
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1120 / 1120
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Points Scored
2 / 2
Your Total Score: 32
Time Taken
0.06
Solve Next
BFS of graph
Mother Vertex
Count the paths
C++ (g++ 5.4)
Start Timer
// } Driver Code Ends
class Solution {
public:
// Function to return a list containing the DFS traversal of the graph.
vector<int> dfsOfGraph(vector<vector<int>>& adj) {
int V = adj.size(); // Number of vertices
vector<int> dfsResult;
vector<bool> visited(V, false);
stack<int> st;
st.push(0);
while (!st.empty()) {
int node = st.top();
st.pop();
if (!visited[node]) {
visited[node] = true;
dfsResult.push_back(node);
// Push adjacent nodes in reverse order to maintain correct order
for (auto it = adj[node].rbegin(); it != adj[node].rend(); ++it) {
if (!visited[*it]) {
st.push(*it);
}
}
}
}
return dfsResult;
}
};
// } Driver Code Ends
Ctrl + Enter