|  |  |
| --- | --- |
| **Depth First Search in C++** | |
| #include <iostream>  #include <vector>  using namespace std;  class DFSDirected {  public:      static vector<int> dfs(int s, vector<bool>& vis, vector<vector<int>>& adj, vector<int>& ls) {          vis[s] = true;          ls.push\_back(s);          for (int it : adj[s]) {              if (!vis[it]) {                  dfs(it, vis, adj, ls);              }          }          return ls;      }  };  int main() {      int V = 5;      vector<bool> vis(V + 1, false);      vector<int> ls;      vector<vector<int>> adj(V + 1);      adj[1].push\_back(3);      adj[1].push\_back(2);      adj[3].push\_back(4);      adj[4].push\_back(5);      vector<vector<int>> res;      for (int i = 1; i <= V; i++) {          if (!vis[i]) {              vector<int> ls;              res.push\_back(DFSDirected::dfs(i, vis, adj, ls));          }      }      for (const auto& component : res) {          for (int node : component) {              cout << node << " ";          }          cout << endl;      }      return 0;  } | **Graph Construction:**  int V = 5;  adj[1].push\_back(3); // 1 → 3  adj[1].push\_back(2); // 1 → 2  adj[3].push\_back(4); // 3 → 4  adj[4].push\_back(5); // 4 → 5  So the graph looks like:  1 → 2  ↓  3 → 4 → 5  **🔁 DFS Traversal (starting from unvisited nodes)**  Looping over i = 1 to 5:   | **i** | **vis[i]** | **DFS Starts?** | **DFS Order (Component)** | | --- | --- | --- | --- | | 1 | false | Yes | 1 → 3 → 4 → 5, then 2 → | | 2 | true | No | Already visited from 1 | | 3 | true | No | Already visited from 1 | | 4 | true | No | Already visited from 1 | | 5 | true | No | Already visited from 1 |   **Note**: 2 is visited after 1, since it's a neighbor of 1 and called later in the loop.  So only **one DFS call** is needed, and it covers **all reachable nodes from 1**.  **📤 DFS Order (Component):**   * From node 1: 1 → 3 → 4 → 5, and then the loop in DFS continues with 2.   So final traversal list:  1 3 4 5 2  **🧾 Output:**  1 3 4 5 2 |
| **Output:- 1 3 4 5 2** | |