|  |  |
| --- | --- |
| **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:  1 → 3 → 4 → 5  ↓  2 Adjacency list:  adj[1] = {3, 2}  adj[2] = {}  adj[3] = {4}  adj[4] = {5}  adj[5] = {}  Execution Steps   1. Initialize vis = {false, false, false, false, false, false} (1-based indexing). 2. Start iterating from i = 1 to i = 5.   **DFS Starting from Node 1**:   * Call dfs(1, vis, adj, ls):   + Mark vis[1] = true, add 1 to ls.   + Visit neighbors 3 and 2 of node 1.   **Visit Node 3**:   * Call dfs(3, vis, adj, ls):   + Mark vis[3] = true, add 3 to ls.   + Visit neighbor 4.   **Visit Node 4**:   * Call dfs(4, vis, adj, ls):   + Mark vis[4] = true, add 4 to ls.   + Visit neighbor 5.   **Visit Node 5**:   * Call dfs(5, vis, adj, ls):   + Mark vis[5] = true, add 5 to ls.   + No more neighbors to visit; return.   **Backtrack**:   * Backtrack to node 4, then to 3, and finally to 1.   **Visit Node 2**:   * Call dfs(2, vis, adj, ls):   + Mark vis[2] = true, add 2 to ls.   + No more neighbors to visit; return.   **Result for DFS from Node 1**:   * First connected component: [1, 3, 4, 5, 2].   **Remaining Iterations**:   * For i = 2, 3, 4, 5, all nodes are already visited, so no new DFS is initiated. |
| **Output:- 1 3 4 5 2** | |