

1. Write a C++ program to find all the strongly connected components in the given directed graph.



strongly_component.cpp - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

strongly_component.cpp X Task_scheduling.cpp X

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  const int WHITE = 0;
6  const int GRAY = 1;
7  const int BLACK = 2;
8
9  vector<int> color;
10 vector<int> DescovaryTime;
11 vector<int> FinishTime;
12 vector<int> Ancestor;
13 stack<int> finishvectex;
14
15 int timeCount;
16
17 void DFS_VISIT(vector<vector<int>>& G, int u)
18 {
19     timeCount++;
20     DescovaryTime[u] = timeCount;
21     color[u] = GRAY;
22
23     for(int v=0; v<G[u].size(); v++)
24     {
25         if(G[u][v]==1 && color[v]==WHITE)
26         {
27             Ancestor[v] = u;
28             DFS_VISIT(G, v);
29         }
30     }
31     timeCount++;
32     FinishTime[u] = timeCount;
33     color[u] = BLACK;
34     finishvectex.push(u);
35 }
36
37 void DFS(vector<vector<int>>& G)
38 {
39     int n = G.size();
40     color.assign(n, WHITE);
41     DescovaryTime.assign(n, 0);
42     FinishTime.assign(n, 0);
43     Ancestor.assign(n, -1);
44
45     for(int u=0; u<n; u++)
46     {
47         if(color[u] == WHITE)
48             DFS_VISIT(G, u);
49     }
50
51     vector<vector<int>> Transpose(vector<vector<int>>& G)
52     {
53         int n = G.size();
54         vector<vector<int>> GT(n, vector<int>(n, 0));
55
56         for(int u=0; u<n; u++)
57         {
58             for(int v=0; v<G[u].size(); v++)
59             {
60                 if(G[u][v]==1)
61                     GT[v][u] = 1;
62             }
63         }
64         return GT;
65     }
66
67 void DFS_VISIT_snd(vector<vector<int>>& G, int u, vector<int>& compone
68 {
69     color[u] = GRAY;
70     component.push_back(u);
71
72     for(int v=0; v<G[u].size(); v++)
73     {
74         if(G[u][v]==1 && color[v]==WHITE)
75         {
76             DFS_VISIT_snd(G, v, component);
77         }
78     }
79     color[u] = BLACK;
80 }
81
82 void dfs_fscs(vector<vector<int>>& G)
83 {
84     DFS(G);
85     vector<vector<int>> GT = Transpose(G);
86     int n = G.size();
87     color.assign(n, WHITE);
88
89     while(!finishvectex.empty())
90     {
91         int u = finishvectex.top();
92         finishvectex.pop();
93         if (color[u] == WHITE)
94         {
95             vector<int> component;
96             DFS_VISIT_snd(GT, u, component);
97
98             cout<<"{";
99             for(auto i: component)
100                 cout<<i<<" ";
101             cout<<"}"<<endl;
102         }
103     }
104 }
105
106 int main()
107 {
108     vector<vector<int>> G=
109     {
110         {0,1,0,0,0,0,0,0},
111         {0,0,1,0,0,0,0,0},
112         {0,0,0,1,0,0,0,0},
113         {1,0,0,0,0,0,0,0},
114         {0,0,0,0,0,1,0,0},
115         {0,0,0,0,0,0,1,0},
116         {0,0,0,0,1,0,0,0},
117         {0,0,0,0,0,0,0,0},
118     };
119     dfs_fscs(G);
120     return 0;
121 }

```

Process returned 0 (0x0) execution time : 0.192 s
Press any key to continue.

strongly_component.cpp - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

strongly_component.cpp X Task_scheduling.cpp X

```

69 void DFS_VISIT_snd(vector<vector<int>>& G, int u, vector<int>& compone
70 {
71     color[u] = GRAY;
72     component.push_back(u);
73
74     for(int v=0; v<G[u].size(); v++)
75     {
76         if(G[u][v]==1 && color[v]==WHITE)
77         {
78             DFS_VISIT_snd(G, v, component);
79         }
80     }
81     color[u] = BLACK;
82 }
83
84 void dfs_fscs(vector<vector<int>>& G)
85 {
86     DFS(G);
87     vector<vector<int>> GT = Transpose(G);
88     int n = G.size();
89     color.assign(n, WHITE);
90
91     while(!finishvectex.empty())
92     {
93         int u = finishvectex.top();
94         finishvectex.pop();
95         if (color[u] == WHITE)
96         {
97             vector<int> component;
98             DFS_VISIT_snd(GT, u, component);
99
100             cout<<"{";
101             for(auto i: component)
102                 cout<<i<<" ";
103             cout<<"}"<<endl;
104         }
105     }
106 }
107
108 int main()
109 {
110     vector<vector<int>> G=
111     {
112         {0,1,0,0,0,0,0,0},
113         {0,0,1,0,0,0,0,0},
114         {0,0,0,1,0,0,0,0},
115         {1,0,0,0,0,0,0,0},
116         {0,0,0,0,0,1,0,0},
117         {0,0,0,0,0,0,1,0},
118         {0,0,0,0,1,0,0,0},
119         {0,0,0,0,0,0,0,0},
120     };
121     dfs_fscs(G);
122     return 0;
123 }

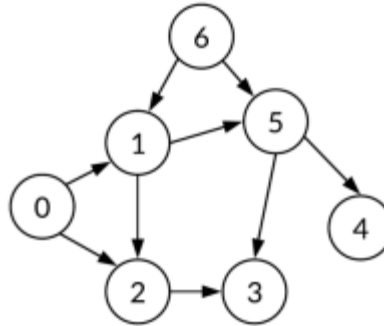
```

2. You are given a directed acyclic graph (DAG) representing N tasks, where:

- Each node corresponds to a task.
- A directed edge (u,v) indicates that task u must be completed before task v .

Write a C++ program to determine a valid order to complete all tasks. If multiple valid orders exist, any one of them is acceptable.

Unsorted graph



```
Topological_sort.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

strongly_component.cpp X Topological_sort.cpp X

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  const int WHITE = 0;
6  const int GRAY = 1;
7  const int BLACK = 2;
8
9  vector<int> color;
10 vector<int> DiscoveryTime;
11 vector<int> FinishTime;
12 vector<int> Ancestor;
13 vector<int> finishvectex;
14 int timeCount;
15
16 void DFS_VISIT(vector<vector<int>>& G, int u)
17 {
18     timeCount++;
19     DiscoveryTime[u] = timeCount;
20     color[u] = GRAY;
21
22     for(int v=0; v<G[u].size(); v++)
23     {
24         if(G[u][v]==1 && color[v]==WHITE)
25         {
26             Ancestor[v] = u;
27             DFS_VISIT(G, v);
28         }
29     }
30     timeCount++;
31     FinishTime[u] = timeCount;
32     color[u] = BLACK;
33     finishvectex.push_back(u);
34 }
35
36 void DFS(vector<vector<int>>& G)
37 {
38     int n = G.size();
39     color.assign(n, WHITE);
40     DiscoveryTime.assign(n, 0);
41     FinishTime.assign(n, 0);
42     Ancestor.assign(n, -1);
43     finishvectex.clear();
44
45     for(int u=0; u<n; u++)
46     {
47         if(color[u] == WHITE)
48             DFS_VISIT(G, u);
49     }
50
51     vector<int> Topological_sort(vector<vector<int>>& G)
52     {
53         DFS(G);
54         reverse(finishvectex.begin(), finishvectex.end());
55         return finishvectex;
56     }
57
58     int main()
59     {
60         vector<vector<int>> G=
61         {
62             {0,1,1,0,0,0,0},
63             {0,0,1,0,0,1,0},
64             {0,0,0,1,0,0,0},
65             {0,0,0,0,0,0,0},
66             {0,0,0,0,0,0,0},
67             {0,0,0,1,1,0,0},
68             {0,1,0,0,0,1,0}
69         };
70
71         vector<int> SortedOrder = Topological_sort(G);
72         for(int i: SortedOrder)
73             cout<<i<<" ";
74         cout<<endl;
75         return 0;
76     }
77 }
```

Process returned 0 (0x0) execution time : 0.185 s
Press any key to continue.

6 0 1 5 4 2 3

C:\Users\Golam Mostafa Rabby\OneDrive\B.Sc in Duet\3rd year 1st semester\Sessiona... C/C++ Windows (CR+LF) WINDOWS-1252 Line 23, Col 6, Pos 430 Insert Read/Write default 11:20 PM 12/5/2024