

```
1.  /***** Create SCC_to_DAG by Faisal *****/
2.  #include <bits/stdc++.h>
3.  using namespace std;
4.  #define F first
5.  #define S second
6.  #define pb push_back
7.
8.  #define _ ios_base::sync_with_stdio(0);
9.  #define fastt ios_base::sync_with_stdio(false); cin.tie(0);
10. #define prec cout<<fixed<<setprecision(16);
11. // #define sess cout<<endl;
12.
13. #define file freopen("000.txt","r",stdin);
14. #define balsal cout<<"dhukse"<<endl;
15.
16. #define pi acos(-1.0)
17. #define PI 3.141592653589793
18. #define INF 1e9
19.
20. #define si(n) scanf("%d",&n)
21. #define sii(a,b) scanf("%d%d",&a,&b)
22. #define sii(a,b,c) scanf("%d%d%d",&a,&b,&c)
23. #define sl(a) scanf("%lld",&a)
24. #define sll(a,b) scanf("%lld%lld",&a,&b)
25. #define slll(a,b,c) scanf("%lld%lld%lld",&a,&b,&c)
26. #define sf(n) scanf("%lf",&n)
27. #define ss(n) scanf("%s",&n)
28.
29. #define it_multiset std::multiset<int>::iterator it;
30. #define it_mii std::map<int, int>::iterator it;
31.
32. #define MOD 1000000007
33. #define mod 1000000007
34. // #define con continue;
35. // #define ret return
36. #define mx_value 1e19
37.
38.
39. long long int sett(long long int N,int pos)
40. {
41.     return N=N | (1<<pos);
42. }
43. int reset(int N,int pos)
44. {
45.     return N= N & ~(1<<pos);
46. }
47. bool check(int N,int pos)
48. {
49.     return (bool)(N & (1<<pos));
50. }
51.
52. void update(long long &x, long long val)
53. {
54.     x = x+val;
55.     if(x>mod)x-=mod;
56. }
57.
58. void fast()
```

```
59. {
60.     ios_base::sync_with_stdio(0);
61.     cin.tie(NULL), cout.tie(NULL);
62. }
63.
64. int moves[4][2] = { {1,0} , {0,1} , {-1,0}, {0,-1} };
65.
66. typedef long long int ll;
67. typedef float flt;
68. typedef double dbl;
69. typedef vector<int > vi;
70. typedef pair < int , int > pii;
71. typedef pair < ll , ll > pll;
72. typedef map<int , int> mii;
73.
74.
75. #define N 10100
76.
77. vi front_graph[N];
78. vi reverse_graph[N];
79. vi dag[N];
80.
81. int vis[N]= {};
82.
83. int bal[N];
84. int bal_indx=0;
85. int bal_cnt=0;
86. int sal[N];
87. int sal_indx=0;
88. int sal_cnt=0;
89.
90. void reset(int n)
91. {
92.     for(int i=0; i<=n; i++)
93.     {
94.         dag[i].clear();
95.         vis[i]=0;
96.         front_graph[i].clear();
97.         reverse_graph[i].clear();
98.         bal[i]=0;
99.         sal[i]=0;
100.    }
101. }
102.
103. void dfs(int node, int type)
104. {
105.     vis[node] = 1;
106.     int loop;
107.     if(type==1) loop = front_graph[node].size();
108.     else loop = reverse_graph[node].size();
109.     for(int i=0; i<loop; i++)
110.     {
111.         int next;
112.         if(type==1) next = front_graph[node][i];
113.         else next = reverse_graph[node][i];
114.         if(!vis[next])
115.         {
116.             dfs(next,type);
```

```
117.     }
118. }
119. if(type==1)
120. {
121.     bal[bal_indx++] = node;
122.     bal_cnt++;
123. }
124. else
125. {
126.     sal[sal_indx++] = node;
127.     sal_cnt++;
128. }
129. }
130.
131. int main()
132. {
133.
134. #ifndef ONLINE_JUDGE
135.     file;
136. #endif // ONLINE_JUDGE
137.     int t;
138.     si(t);
139.     int cs = 1;
140.     while(t--)
141.     {
142.         int n,m;
143.         sii(n,m);
144.         reset(n);
145.         for(int i=0; i<m; i++)
146.         {
147.             int p,q;
148.             sii(p,q);
149.             front_graph[p].pb(q);
150.             reverse_graph[q].pb(p);
151.         }
152.
153.
154.         bal_cnt=0;
155.         bal_indx=0;
156.
157.         for(int i=1; i<=n; i++)
158.         {
159.             if(!vis[i])
160.             {
161.                 dfs(i,1);
162.             }
163.         }
164.         map < int , int > map_vertex;
165.         memset(vis,0,sizeof(vis));
166.         int cnt = 0;
167.         for(int i=n-1; i>=1; i--)
168.         {
169.             if(!vis[bal[i]])
170.             {
171.                 sal_indx=0;
172.                 sal_cnt=0;
173.                 cnt++;
174.                 dfs(bal[i],2);
```

```
175.         for(int j=0; j<sal_cnt; j++)
176.         {
177.             map_vertex[sal[j]] = cnt; //asol graph r kon inedx dag er kon vertex bujhay
178.         }
179.     }
180. }
181. for(int i=1;i<=n;i++)
182. {
183.     int loop = front_graph[i].size();
184.     for(int j=0;j<loop;j++)
185.     {
186.         int next = front_graph[i][j];
187.         int a = map_vertex[i];
188.         int b = map_vertex[next];
189.         // if(cs==79)cout<<a<<" "<<b<<endl; //dag er edge gula
190.         dag[a].push_back(b);
191.     }
192. }
193. //ei code e sob vertex 1 theke hisab kora hoise
194. //ekhn amra dag paisi
195. //ekhn dag er upore ja khushi kora jabe
196.
197. }
198.
199.
200. return 0;
201. }
```