Pastebin Link: <http://paste.ubuntu.com/25076248/>

#define NODES 100105  
**using** **namespace** std;  
vector <**int**> vec, adj[NODES], adjt[NODES], adjScc[NODES];  
**bool** vis[NODES];  
**int** scc[NODES];  
**int** sccNo;  
**int** n;  
**void** dfs1(**int** node){  
 vis[node]=**true**;  
 **for**(**unsigned** **int** i=0; i<adj[node].size(); i++){  
 **int** v=adj[node][i];  
 **if**(!vis[v]){  
 dfs1(v);  
 }  
 }  
 vec.push\_back(node);  
}  
  
**void** dfs2(**int** node, **int** comp){  
 vis[node]=**true**;  
 scc[node]=comp;  
 **for**(**unsigned** **int** i=0; i<adjt[node].size(); i++){  
 **int** v=adjt[node][i];  
 **if**(!vis[v]){  
 dfs2(v, comp);  
 }  
 }  
}  
  
**void** buildSccGraph(){  
 **for**(**int** i=1; i<=n; i++){  
 **for**(**unsigned** **int** j=0; j<adj[i].size(); i++){  
 **int** v=adj[i][j];  
 **if**(scc[i]!=scc[v]){  
 adjScc[scc[i]].push\_back(scc[v]);  
 }  
 }  
 }  
}  
  
**void** dag\_sort(**int** node){  
 vis[node]=**true**;  
 **for**(**unsigned** **int** i=0; i<adjScc[i].size(); i++){  
 **int** v=adjScc[node][i];  
 **if**(!vis[v]){  
 dag\_sort(v);  
 }  
 }  
 vec.push\_back(node);  
}  
  
**void** scc\_build(){  
 memset(vis, **false**, **sizeof**(vis));  
 **for**(**int** i=1; i<=n; i++){  
 **if**(!vis[i]) dfs1(i);  
 }  
 memset(vis, **false**, **sizeof**(vis));  
 sccNo=0;  
 **for**(**int** i=vec.size()-1; i>=0; i--){  
 **int** v=vec[i];  
 **if**(!vis[v]){  
 sccNo++;  
 dfs2(v, sccNo);  
 }  
 }  
 buildSccGraph();  
 vec.clear();  
 memset(vis, **false**, **sizeof**(vis));  
 **for**(**int** i=1; i<=sccNo; i++){  
 **if**(!vis[i]){  
 dag\_sort(i);  
 }  
 }  
}