# 20221117 数据结构与算法 解题报告

## Can I Post the letter

利用邻接表储存有向图，BFS遍历即可。

时间复杂度：O(n)

#include <bits/stdc++.h>  
using namespace std;  
  
struct Edge  
{  
 int vec;  
 Edge \*next;  
 Edge(int \_vec = 0, Edge \*\_next = NULL) : vec(\_vec), next(\_next) {}  
 ~Edge()  
 {  
 if (next != NULL)  
 delete next;  
 }  
};  
  
bool judge(int n, const vector<Edge \*> &finalEdge)  
{  
 queue<int> que;  
 vector<bool> sgn(n, false);  
 que.push(0);  
 sgn[0] = true;  
  
 for (; !que.empty(); que.pop())  
 {  
 int cur = que.front();  
 for (Edge \*edge = finalEdge[cur]; edge != NULL; edge = edge->next)  
 if (!sgn[edge->vec])  
 {  
 que.push(edge->vec);  
 sgn[edge->vec] = true;  
  
 if (edge->vec == n - 1)  
 return true;  
 }  
 }  
  
 return false;  
}  
  
int main(int argc, char const \*argv[])  
{  
 // freopen("init.in", "r", stdin);  
 for (int n, m; cin >> n >> m;)  
 {  
 vector<Edge \*> finalEdge(n, NULL);  
 for (; m--;)  
 {  
 int u, v;  
 cin >> u >> v;  
 finalEdge[u] = new Edge(v, finalEdge[u]);  
 }  
  
 cout << (judge(n, finalEdge) ? "I can post the letter" : "I can't post the letter") << endl;  
  
 for (int i = 0; i < n; i++)  
 delete finalEdge[i];  
 }  
  
 return 0;  
}

## Compute Active time Interval during DFS

通过邻接表储存无向图，以公共事件戳进行DFS即可。

时间复杂度：O(n)

#include <bits/stdc++.h>  
using namespace std;  
  
struct Road  
{  
 int vec;  
 Road\* nxt;  
 Road(int \_vec, Road\* \_nxt) : vec(\_vec), nxt(\_nxt) {}  
 ~Road()  
 {  
 if (nxt != NULL)  
 delete nxt;  
 }  
};  
  
void DFS(int cur, const vector<Road\*>& final\_road,  
 vector<pair<int, int> >& IOtime, int& time)  
{  
 if (IOtime[cur].first)  
 return;  
  
 IOtime[cur].first = ++time;  
  
 for (Road\* road = final\_road[cur]; road != NULL; road = road->nxt)  
 DFS(road->vec, final\_road, IOtime, time);  
  
 IOtime[cur].second = ++time;  
}  
  
int main()  
{  
 // freopen("init.in", "r", stdin);  
  
 int T;  
 cin >> T;  
 for (; T--;)  
 {  
 int n, m;  
 cin >> n >> m;  
  
 vector<pair<int, int> > edge;  
 for (; m--;)  
 {  
 int u, v;  
 cin >> u >> v;  
 edge.push\_back(pair<int, int>(u, v));  
 edge.push\_back(pair<int, int>(v, u));  
 }  
 sort(edge.rbegin(), edge.rend());  
  
 vector<Road\*> final\_road(n + 1, NULL);  
 vector<pair<int, int> > IOtime(n + 1, pair<int, int>(0, 0));  
  
 for (auto& cur:edge)  
 final\_road[cur.first] = new Road(cur.second, final\_road[cur.first]);  
  
  
 int time = 0;  
 for (int i = 1; i <= n; i++)  
 DFS(i, final\_road, IOtime, time);  
  
 for (int i = 1; i <= n; i++)  
 cout << i << ":" << IOtime[i].first << "-" << IOtime[i].second << endl;  
 cout << "---" << endl;  
  
 for (int i = 1; i <= n; i++)  
 delete final\_road[i];  
 }  
 return 0;  
}

## Connect conponents in unddirected graph

利用并查集实现，合并时统计答案。

时间复杂度：O(n) （并查集压缩剪枝）

#include <bits/stdc++.h>  
using namespace std;  
  
int getFather(int cur, vector<int> &father)  
{  
 if (father[cur])  
 return father[cur] = getFather(father[cur], father);  
 else  
 return cur;  
}  
  
int merge(int u, int v, vector<int> &father)  
{  
 if (getFather(u, father) == getFather(v, father))  
 return 0;  
 else  
 {  
 father[getFather(u, father)] = getFather(v, father);  
 return 1;  
 }  
}  
  
int main()  
{  
 // freopen("init.in", "r", stdin);  
 int n, m;  
 cin >> n >> m;  
  
 vector<int> father(n + 1);  
  
 int ans = n;  
 for (; m--;)  
 {  
 int u, v;  
 cin >> u >> v;  
 ans -= merge(u, v, father);  
 }  
  
 cout << ans << endl;  
  
 return 0;  
}