# 20221124 数据结构与算法 解题报告

21307289 刘森元

## DAG?

对于每一个点进行DFS搜索，路径进行标记，判断有无回路。

时间复杂度：O(n^2)

p.s. 当然可以使用Tarjan算法判断联通量，但是考虑到n的范围，为了简化使用传统DFS显然更优。

#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 cur, vector<bool>& sgn, vector<Edge\*>& finalEdge)  
{  
 if (sgn[cur])  
 return false;  
 sgn[cur] = true;  
  
 for (Edge\* edge = finalEdge[cur]; edge != NULL; edge = edge->next)  
 if (!judge(edge->vec, sgn, finalEdge))  
 return false;  
  
 sgn[cur] = false;  
 return true;  
}  
  
int main(int argc, char const\* argv[])  
{  
 freopen("init.in", "r", stdin);  
 int n, m;  
 cin >> n >> m;  
 vector<Edge\*> finalEdge(n + 1, NULL);  
 for (; m--;)  
 {  
 int u, v;  
 cin >> u >> v;  
 finalEdge[u] = new Edge(v, finalEdge[u]);  
 }  
  
 int ans = 1;  
 vector<bool> sgn(n + 1, false);  
 for (int i = 1; i <= n; i++)  
 if (!judge(i, sgn, finalEdge))  
 {  
 ans = 0;  
 break;  
 }  
  
 cout << ans << endl;  
  
 for (int i = 0; i <= n; i++)  
 delete finalEdge[i];  
 return 0;  
}

## Ordering Tasks

考虑从节点入度下手，每次处理入度为0的节点，更新节点入度。由于题目要求答案字典序最小，使用堆处理入度为0的节点编号即可。

时间复杂度：O(n+ln(n)+m)

#include <bits/stdc++.h>  
using namespace std;  
  
struct Edge  
{  
 int vec;  
 Edge\* next;  
 Edge(int \_vec = -1, Edge\* \_next = NULL) : vec(\_vec), next(\_next) {}  
 ~Edge()  
 {  
 if (next != NULL)  
 delete next;  
 }  
};  
  
int main(int argc, char const\* argv[])  
{  
 freopen("init.in", "r", stdin);  
 int T;  
 cin >> T;  
 for (; T--;)  
 {  
 int n, m;  
 cin >> n >> m;  
 vector<Edge\*> finalEdge(n + 1, NULL);  
 vector<int> deg(n + 1, 0);  
 for (int i = 1;i <= n;i++)  
 finalEdge[0] = new Edge(i, finalEdge[0]), deg[i]++;  
 for (;m--;) {  
 int u, v;  
 cin >> u >> v;  
 finalEdge[u] = new Edge(v, finalEdge[u]), deg[v]++;  
 }  
  
 priority\_queue<int, vector<int>, greater<int> > que;  
 que.push(0);  
 for (; !que.empty();)  
 {  
 int cur = que.top();  
 que.pop();  
 if (cur)  
 cout << cur << " ";  
  
 for (Edge\* edge = finalEdge[cur]; edge != NULL; edge = edge->next)  
 {  
 deg[edge->vec]--;  
 if (!deg[edge->vec])  
 que.push(edge->vec);  
 }  
 }  
 cout << endl;  
  
 for (int i = 0; i <= n; i++)  
 delete finalEdge[i];  
 }  
  
 return 0;  
}

## Euler Euler

使用并查集判断图是否联通，并且统计度数为奇数的节点数量即可。

时间复杂度：O(n)

#include <bits/stdc++.h>  
using namespace std;  
  
int getFather(int cur, vector<int> &father)  
{  
 if (father[cur] == -1)  
 return cur;  
 else  
 return father[cur] = getFather(father[cur], father);  
}  
  
int merge(int u, int v, vector<int> &father)  
{  
 if (getFather(u, father) != getFather(v, father))  
 {  
 father[getFather(u, father)] = getFather(v, father);  
 return 1;  
 }  
 else  
 return 0;  
}  
  
int main(int argc, char const \*argv[])  
{  
 freopen("init.in", "r", stdin);  
 int T;  
 cin >> T;  
 for (; T--;)  
 {  
 int n, m;  
 cin >> n >> m;  
 vector<int> deg(n + 1, 0);  
 vector<int> father(n + 1, -1);  
 merge(1, 0, father);  
  
 int split = n;  
 for (; m--;)  
 {  
 int u, v;  
 cin >> u >> v;  
 deg[u]++, deg[v]++;  
 split -= merge(u, v, father);  
 }  
  
 int type = 0;  
 if (split != 1)  
 type = 5;  
  
 for (int i = 1; i <= n; i++)  
 if (!deg[i])  
 type = 5;  
 else if (deg[i] & 1)  
 type++;  
  
 if (type == 0)  
 cout << "Euler Circuit" << endl;  
 else if (type == 2)  
 cout << "Euler Path" << endl;  
 else  
 cout << "Neither" << endl;  
 }  
  
 return 0;  
}