E4E 小水獭和哈密顿路径

题目描述

给定一个 n 个点 m 条边的有向无环图，判断图中是否存在一条哈密顿路径。

输入格式

第一行一个正整数 t（$1 \leq t \leq 101 \leq t \leq 10$），表示数据组数。
对于每组数据，第一行两个正整数 n, m（$1 \leq n \leq 10^5$，$1 \leq m \leq 2 \times 10^5$），含义如题目所示。
接下来 m 行，每行两个正整数 u_i, v_i（$1 \leq u_i, v_i \leq n$），表示第 i 条边为 $u_i \to v_i$。
保证图是有向无环图，且没有重边。

解题思路

- 哈密顿路径：$G = (V, E)$ 是一个图，若 G 中一条路径通过且仅通过每一个顶点一次，称这条**路径**为**哈密顿路径。**
- 拓扑排序： 在图论中， 拓扑排序是一个 有向无环图（DAG） 的所有顶点的线性序列。 且该序列必须满足下面两个条件： **每个顶点出现且只出现一次。** 若存在一条从顶点 A 到顶点 B 的路径，那么在序列中顶点 A 出现在顶点 B 的前面。

解题步骤：

1. 构造一条拓扑排序
2. 如果该条拓扑排序前后的每个顶点之间均有一条路连接，那么它就是一条哈密顿路径。

拓扑排序求法

- 将所有点中入度为零的点入队
- 取出队首元素，将所有由这个点出发的边全部删掉。
- 检查一遍剩余点，将剩余点中入度为零的点入队
- 重复上述过程直至全部点被队列弹出
- 弹出的顺序即为拓扑排序

代码展示

```
#include<bits/stdc++.h>
using namespace std;

//#define abyss
typedef unsigned int UI;
typedef long long LL;
typedef long double LD;
#define re register;
#define ln(i)  ((i<<1))
#define rn(i)  (((i<<1|1)))
#define PLL pair<int,int>
#define MK(a,b) make_pair(a,b)
#define FOR(i,begin,end)   for(register int i = begin; i < end; ++i)
#define rFOR(i,begin,end)   for(register int i = begin; i > end; --i)
#define clr(x,y)  memset(x,y,sizeof(x))
```

```
LL gcd(LL a, LL b) {return b == 0 ? a : gcd(b,a%b);}
template <typename T> inline T read() {
    T x = 0, sgn = 1;
    char c = getchar();
    for(; c < '0' || c > '9' ; c = getchar())
        if(c == '-') sgn = -1;
    for(; c >= '0' && c <= '9'; c = getchar())
        x = (x << 1) + (x << 3) + (c ^ 48);
    return x * sgn;
}
#define maxE 200005
#define maxP 100005
typedef struct e{
    int to,next;
}e;
e edge[maxE];
int in[maxP],head[maxP],cnt;
void add(int u,int v) {
    edge[++cnt].to = v;
    edge[cnt].next = head[u];
    head[u] = cnt;
}
void init() {clr(in,0); clr(head,0); cnt = 0;}

int main()
{
#ifdef abyss
    freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
#endif

    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    int t = read<int>();
    while(t--) {
        int n = read<int>(), m = read<int>();
        init();
        FOR(i,1,m+1) {
            int u = read<int>(), v = read<int>();
            add(u,v); in[v] += 1;
            // 加边，数组元素 in[i] 表示第 i 个点的入度
        }
        vector<int> topology;  // 储存弹出元素，即拓扑排序
        queue<int> q;  // 构造一个队列
        FOR(i,1,n+1) if(!in[i]) q.push(i); //入度为零的入队
        while(!q.empty()) {
            int top = q.front();  q.pop();  //去队首元素
            topology.push_back(top);  //加入序列中
            for(int i = head[top]; i != 0; i = edge[i].next) {
                in[edge[i].to]--;  //遍历由这个点发出的全部的边并删去
                if(!in[edge[i].to]) q.push(edge[i].to);
                //如果删去后到达的那个点入度为零则入队
            }
        }
        bool got = true;
        FOR(i,0,n-1){
            int pre = topology[i], beh = topology[i+1];
            //取相邻两个顶点
            bool found = false;
```

```
            for(int i = head[pre]; i != 0; i = edge[i].next)    if(edge[i].to
== beh)
                {found = true;   break;}
            // 如果两个点之间有边则继续遍历
            if(!found) {got = false;break;}
            //若发现两个点之间无边则表示不存在哈密顿路径
        }
        if(got) cout << "yy\n"; else cout << "nn\n";
    }
    return 0;
}
```