Codeforces Round #664 (Div. 2)

比赛链接

E. Boboniu Walks on Graph

题意

给定一个 n 个点 m 条边的有向图，每个点出度 $\in [1,k]$，每条边的权为 $1\sim m$ 且各不相同。

要求输出满足下列条件的 k 元组 $(c_1,c_2\cdots c_k)$ 的数目：

1. $1\le c_i\le i$
2. 如果一个点的出度为 i，则沿他的出边中边权第 c_i 大的边走向下一个顶点。需要保证每个点能在有限步内回到起点。

题解

对某个 k 元组 $(c_1,c_2\cdots c_k)$，记 $f(u)$ 表示 u 沿上述规则走到的下一个点。

则 $(c_1,c_2\cdots c_k)$ 满足条件当且仅当 f 为 $1\sim n$ 的置换。

记出度为 i 的点集沿他的出边中边权第 j 大的边到达的点集，记为 $S_{i,j}$。

则 f 为 $1\sim n$ 的置换当且仅当 $\bigcup_{i=1}^k S_{i,c_i}=\{1,2\cdots n\}$。

考虑哈希预处理 $S_{i,j}$，则可以 $O(k)$ 时间求出 $\bigcup_{i=1}^k S_{i,c_i}$。

于是暴力枚举所有可能情况即可，时间复杂度 $O(n+m+k!)$。

```
const int
mod[3]={998244353,1004535809,1<<30},MAXN=2e5+5,MAXK=10,Base=32767;
struct Hash_num{
    int h[3];
    Hash_num(LL seed=0){_for(i,0,3)h[i]=seed%mod[i];}
    Hash_num operator + (const Hash_num &b)const{
        Hash_num c;
        c.h[0]=(h[0]+b.h[0])%mod[0];
        c.h[1]=1LL*h[1]*b.h[1]%mod[1];
        c.h[2]=h[2]^b.h[2];
        return c;
    }
    void operator += (const Hash_num &b){
        h[0]=(h[0]+b.h[0])%mod[0];
        h[1]=1LL*h[1]*b.h[1]%mod[1];
        h[2]=h[2]^b.h[2];
```

```
    }
    bool operator == (const Hash_num &b)const{
        _for(i,0,3)if(h[i]!=b.h[i])return false;
        return true;
    }
};
Hash_num a[MAXN],s[MAXK][MAXK],t;
LL get_rand(){return 1LL*rand()*Base*Base+1LL*rand()*Base+rand();}
struct Edge{
    int to,next;
}edge[MAXN<<1];
int deg[MAXN],head[MAXN],edge_cnt,k,ans;
pair<int,int> temp[MAXN];
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
void dfs(int pos,Hash_num cur){
    if(pos>k){
        if(cur==t)ans++;
        return;
    }
    _rep(i,1,pos)
    dfs(pos+1,cur+s[pos][i]);
}
int main()
{
    srand(time(NULL));
    int n=read_int(),m=read_int(),u,v,w;
    k=read_int();
    _rep(i,1,m){
        u=read_int(),v=read_int(),w=read_int();
        temp[w]=make_pair(u,v);
        deg[u]++;
    }
    for(int i=m;i;i--)
    Insert(temp[i].first,temp[i].second);
    _rep(i,1,n)a[i]=Hash_num(get_rand()),t+=a[i];
    _rep(u,1,n){
        for(int i=head[u],j=1;i;i=edge[i].next,j++){
            int v=edge[i].to;
            s[deg[u]][j]+=a[v];
        }
    }
    dfs(1,Hash_num());
    enter(ans);
    return 0;
}
```