【bzoj3693】【FJ2014集训】圆桌会议

2014年7月23日 659 0

问题描述

有n组人要一起开一个圆桌会议（编号为0~n-1），会议的圆桌上有m个位置（编号为0~m-1）。每个组有ai个人，他们需要被安排在（li，(li+1)%m，(li+2)%m，…，ri）的座位范围内。每个座位只能安排一个人就坐，并且每个人都需要被安排一个座位。现在你需要判断是否存在满足条件的座位安排。

输入格式

输入包含不超过10组数据。

第一行有一个数字T，表示数据组数。

接下来有T组数据，每组数据第一行包含两个数n，m，表示有多少组的人与圆桌的位置数。

每组数据接下来包含n行，每行包含3个数li，ri，ai。

输出格式

对于每组数据，输出”Yes”或”No”，表示是否存在符合条件的安排。

样例输入

2

2 4

0 1 2

1 2 2

2 3

2 0 2

1 1 1

样例输出

No

Yes

数据规模

对于30%的数据，n≤1000，m≤100000

对于100%的数据，T≤10，其中有不超过3组的数据范围为n≤10^5，m≤10^9，其余与30%数据范围相同

题解

二分图匹配

´假如每个人都看做一个点，构图变成二分图，问题就转换为给定一个二分图，是否存在一个选取了X部所有点的匹配

´Hall定理——对于任意的二分图G，G的两个部分为X={x1,x2,…,xn}和Y={y1,y2,…,ym}，存在一个匹配M使得|M|=|X|的充要条件为对于X的任意一个子集A，与A相邻的点集记为T(A)，一定有|T(A)|≥|A|

´所以可以利用Hall定理来解决这个问题

´首先化简问题，将圆桌当成一条链，考虑链的情况

´对于任意的区间[P,Q]，长度len=Q-P+1，将所有满足[Li,Ri]在区间[P,Q]内的组的ai求和，得到s，如果s>len，显然不存在满足条件的匹配，否则一定存在解

´显然，有意义的询问区间[P,Q]，必定满足Q=Ri（其实也满足P=Lj）

´s>len=Q-P+1，即s+P-1>Q

´对于每个组[Li,Ri]，需要询问每个区间[P,Ri](P≤Li)，是否满足s+P-1>ri

´将所有组（即询问）按照Ri升序排序，依次处理每个询问

´显然这个值key=s+P-1可以用线段树来维护，对于每个组[Li,Ri]，对于区间[1,Li]内每个值P，询问区间[P,Ri]的s都需要增加ai，所以线段树内[1,Li]的每个值key增加ai

´询问[1,Ri]中key的最大值keymax

´如果这个最大值keymax>Ri，那么就不存在合法的匹配

´对于环形的问题，只需要将环拆成链，复制两倍即可

C++

```cpp
#include<iostream>
#include<cstdio>
#include<algorithm>
#define ll long long
using namespace std;
inline int read()
{
    int x=0,f=1;char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
int T,n,cnt,m,tot;
int disc[400005];
struct que{int l,r,v;}q[200005];
struct seg{int l,r,mx,tag;}t[1200005];
inline bool operator<(que a,que b)
{
    return a.r<b.r;
}
void pushup(int k)
{
    t[k].mx=max(t[k<<1].mx,t[k<<1|1].mx);
}
void pushdown(int k)
{
```

```
    if(t[k].l==t[k].r||!t[k].tag)return;
    int tag=t[k].tag;t[k].tag=0;
    t[k<<1].tag+=tag;t[k<<1].mx+=tag;
    t[k<<1|1].tag+=tag;t[k<<1|1].mx+=tag;
}
void build(int k,int l,int r)
{
    t[k].l=l;t[k].r=r;t[k].tag=0;
    if(l==r){t[k].mx=disc[l]-1;return;}
    int mid=(l+r)>>1;
    build(k<<1,l,mid);build(k<<1|1,mid+1,r);
    pushup(k);
}
int find(int x)
{
    int l=1,r=tot;
    while(l<=r)
    {
        int mid=(l+r)>>1;
        if(disc[mid]==x)return mid;
        else if(disc[mid]<x)l=mid+1;
        else r=mid-1;
    }
}
void update(int k,int x,int y,int val)
{
    pushdown(k);
    int l=t[k].l,r=t[k].r;
    if(l==x&&y==r)
    {
        t[k].tag+=val;t[k].mx+=val;
        return;
    }
    int mid=(l+r)>>1;
    if(y<=mid)update(k<<1,x,y,val);
    else if(x>mid)update(k<<1|1,x,y,val);
    else
    {
        update(k<<1,x,mid,val);
        update(k<<1|1,mid+1,y,val);
    }
    pushup(k);
}
int query(int k,int x,int y)
{
    pushdown(k);
    int l=t[k].l,r=t[k].r;
    if(l==x&&y==r)
        return t[k].mx;
    int mid=(l+r)>>1;
    if(y<=mid)return query(k<<1,x,y);
    else if(x>mid)return query(k<<1|1,x,y);
    else return max(query(k<<1,x,mid),query(k<<1|1,mid+1,y));
}
int main()
{
    //freopen("conference.in","r",stdin);
    //freopen("conference.out","w",stdout);
    T=read();
    while(T--)
```

```
    {
        bool flag=0;tot=0;
        n=read();m=read();
        ll sum=0;
        for(int i=1;i<=n;i++)
        {
            q[i].l=read(),q[i].r=read(),q[i].v=read();
            q[i].l++;q[i].r++;sum+=q[i].v;
        }
        if(sum>m)
        {
            printf("No\n");
            continue;
        }
        cnt=n;
        for(int i=1;i<=n;i++)
            if(q[i].r<q[i].l)q[i].r+=m;
            else
            {
                q[++cnt]=q[i];
                q[cnt].l+=m;q[cnt].r+=m;
            }
        n=cnt;
        for(int i=1;i<=n;i++)
        {
            disc[++tot]=q[i].l;disc[++tot]=q[i].r;
        }
        sort(disc+1,disc+tot+1);
        sort(q+1,q+n+1);
        build(1,1,n<<1);
        for(int i=1;i<=n;i++)
        {
            q[i].l=find(q[i].l),q[i].r=find(q[i].r);
        }
        for(int i=1;i<=n;i++)
        {
            update(1,1,q[i].l,q[i].v);
            if(query(1,max(q[i].r-m+1,1),q[i].r)>disc[q[i].r])
            {
                printf("No\n");
                flag=1;
                break;
            }
        }
        if(!flag)printf("Yes\n");
    }
    return 0;
}
```