省选级别试题 第三组 题解

2732 魔环上的树

解

不妨先确定1号节点的位置，并把输入的树以1为根梳理成一棵有根树。我们会发现，在合法方案中，1的每个子树都会占用环上一段连续的位置，因而原问题可以归纳为关于1的各个子树的子问题，进而考虑到树上动态规划算法。

令f[u]表示把u及u的子树摆放在有size[u]（以u为根的子树的大小）个位置的环上的方案数，得到动态规划方程

$$f[u]=(cnt_{u\_son}+[u\neq 1])!*\prod_{v\_is\_son\_of\_u} f[v]$$

cnt_{u_son}指u的儿子个数

时间复杂度O(n)

标准代码

C++：

```
#include<algorithm>
#include<cstdlib>
#include<cstdio>
#include<cmath>
//#include<bits/stdc++.h>
#define mod 998244353
#define N 500050
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 n,head[N],pos;
struct edge{int to,next;}e[N<<1];
void add(int a,int b)
{pos++;e[pos].to=b,e[pos].next=head[a],head[a]=pos;}
void insert(int a,int b){add(a,b),add(b,a);}
int f[N],size[N],fac[N];
void dfs(int u,int fa)
{
    f[u]=1;int k=0;
    for(int i=head[u];i;i=e[i].next)
    {
        int v=e[i].to;
        if(v==fa)continue;
        dfs(v,u);
```

```
            f[u]=1ll*f[u]*f[v]%mod;
            k++;
        }
        f[u]=1ll*f[u]*fac[k+(u!=1)]%mod;
}
void init()
{
    fac[0]=1;
    for(int i=1;i<N;i++)fac[i]=1ll*fac[i-1]*i%mod;
}
int main()
{
    n=read();init();
    for(int i=2;i<=n;i++)
    {
        int x=read(),y=read();
        insert(x,y);
    }
    dfs(1,0);
    printf("%d\n",1ll*n*f[1]%mod);
}
```

2733 序列舞蹈

解

特判：如果序列中含 0 则直接输出

把序列看作无穷长

问题转化成每次区间加一个一次函数，询问全局最小值及位置

并且满足每一次操作的区间一定是后面区间的子区间

易得一点性质

最小值要么是当前序列被操作过（区间加一次函数涉及到）的第一个位置，要么是之前操作的区间 [l,r] 对应的 r+1 （需要保证r+1 被操作过）

第一个位置直接计算

对于后者，设区间 [l,r] 是加入的第 i 个区间（第 i 个区间之后的区间的右端点严格大于 r ）

设 sb 和 ss 表示已经加入的区间的 b 和 s 的前缀和，且当前已经加入了 tot 个区间

那么我们需要最小化的是

$sb_{tot}-sb_i+(ss_{tot}-ss_i)\times r_i$

化下式子

$sb_{tot}-((ss_i\times r_i+sb_i)-ss_{tot}\times r_i)$

即sb_{tot}减去一条斜率为ss_{tot}，过点$(r_i,ss_i\times r_i+sb_i)$的直线的截距

我们需要最大化这个截距

也就是我们需要维护点集$\{i|(r_i,ss_i\times r_i+sb_i)\}$的**上凸壳**

由于凸壳斜率的变化方向和ss_{tot}的变化方向相反

所以我们需要用**单调栈**维护在现在以及之后可能成为最优决策的凸壳点

每次取最优决策时，从栈顶删除不优的决策

复杂度$O(m)$

标准代码

C++11：

```
#include<algorithm>
#include<cstdlib>
#include<cstdio>
#include<random>
#include<cmath>
#define ll long long
#define x1 _x1
#define y1 _y1
#define x2 _x2
#define y2 _y2
#define fr(i,x,y) for(int i=(x);i<=(y);i++)
#define rf(i,x,y) for(int i=(x);i>=(y);i--)
#define frl(i,x,y) for(int i=(x);i<(y);i++)
using namespace std;
const int N=300003;
int n,m;
ll b[N],s[N];
struct data{
    int num,x;
}st[N];  //这是一个栈，维护凸折线上的点
int L;

inline int sign(ll &x){
    if (x>0) return 1;
    if (x<0) return -1;
    return 0;
}

inline int mul(ll x1,ll y1,ll x2,ll y2){  //这个是判断叉积是否>0的
    int w1=sign(x1)*sign(y2),w2=sign(y1)*sign(x2);
    if (w1!=w2) return w1>w2;
    if (w1==0) return 0;
    return (long double)x1*y2>(long double)y1*x2;
}

void read(int &x){ scanf("%d",&x); }
void read(ll &x){ scanf("%lld",&x); }

void chkmin(ll &x,ll y){ if (y<x) x=y; }

inline ll cal(data q,int w){ //计算A[q.x]
    return b[w]-b[q.num]+1ll*(q.x-1)*(s[w]-s[q.num]);
}

void AddPoint(int w,int x){
    if (cal(st[L],w)==0) return;
    ll y=cal((data){w,x},w);
    while(L>=2){
```

```
        if (mul(st[L].x-st[L-1].x,cal(st[L],w)-cal(st[L-1],w),x-st[L].x,y-
cal(st[L],w)))
            break;
        L--;
    }
    st[++L]=(data){w,x};
}

void PopBack(int w){
    while(L>=2&&cal(st[L],w)>=cal(st[L-1],w)) L--;
}

int main(){
    read(n);read(m);
    int tp;ll x,y;
    st[L=1]=(data){0,1};
    fr(i,1,m){
        b[i]=b[i-1];s[i]=s[i-1];
        read(tp);read(x);
        if (tp==1){
            b[i]=s[i]=0;
            st[L=1]=(data){i,1};
            n+=x;
        }else if (tp==2){
            AddPoint(i,n+1);
            n+=x;
        }else{
            read(y);
            b[i]+=x;s[i]+=y;
            PopBack(i);
        }
        printf("%d %lld\n",st[L].x,cal(st[L],i));
    }
    return 0;
}
```

2734 脱单计划

解

直接两两连边显然不行

考虑这样一个转化

$$|x1-x2|=max(x1-x2,x2-x1)$$
$$|x1-x2|+|y1-y2|=max(x1-x2+y1-y2,x2-x1+y1-y2,x1-x2+y2-y1,x2-x1+y2-y1)$$

我们额外建4个中转点表示上面的四种情况，红球和蓝球通过中转点连边，这样边数降到了$O(N)$

边权就按照上面四种情况的符号连，容量为1，跑最大费用最大流。
由于最大费用最大流的性质，保证了每个匹配都是最大的，因此恰好就是曼哈顿距离取了绝对值符号后的结果。

时间复杂度$O(maxflow(N))$

标准代码

C++11 :

```
//#include <bits/stdc++.h>
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<map>
#define fo(i,a,b) for(int i=a;i<=b;++i)
#define fod(i,a,b) for(int i=a;i>=b;--i)
const int N=2115;
const int INF=1e7;
typedef long long LL;
using namespace std;
vector <int> ap[N];
int n,n1,st,ed,f[N][N];
LL ans,pr[N][N];
void link(int x,int y,int w,LL c)
{
    ap[x].push_back(y);
    f[x][y]=w,pr[x][y]=c;
    ap[y].push_back(x);
    f[y][x]=0,pr[y][x]=-c;
}
typedef vector<int>::iterator IT;
namespace Flow
{
    LL dis[N];
    bool bz[N];
    IT cur[N];
    int d[200*N];
    bool spfa()
    {
        memset(dis,107,sizeof(dis));
        memset(bz,0,sizeof(bz));
        dis[st]=0,bz[st]=1,d[1]=st;
        fo(i,1,n1) cur[i]=ap[i].begin();
        int l=0,r=1;
        while(l<r)
        {
            int k=d[++l];
            for(IT i=ap[k].begin();i!=ap[k].end();i++)
            {
                int p=*i;
                if(f[k][p]&&dis[k]+pr[k][p]<dis[p])
                {
                    dis[p]=dis[k]+pr[k][p];
                    if(!bz[p]) bz[p]=1,d[++r]=p;
                }
            }
            bz[k]=0;
        }
        return (dis[ed]<=1e17);
    }
    int flow(int k,int s)
    {
```

```
        if(k==ed) return s;
        int sl=0,v;
        bz[k]=1;
        for(;cur[k]!=ap[k].end();cur[k]++)
        {
            int p=*cur[k];
            if(!bz[p]&&f[k][p]&&dis[p]==dis[k]+pr[k][p])
            {
                if(v=flow(p,min(s,f[k][p])))
                {
                    sl+=v,s-=v;
                    f[k][p]-=v,f[p][k]+=v;
                    ans+=(LL)v*pr[k][p];
                    if(!s) break;
                }
            }
        }
        bz[k]=0;
        return sl;
    }
}
using Flow::flow;
using Flow::spfa;
int main()
{
    cin>>n;
    n1=2*n+6,st=2*n+5,ed=n1;
    fo(i,1,n)
    {
        int x,y,z;
        scanf("%d%d%d",&x,&y,&z);
        link(st,i,z,0);
        link(i,2*n+1,z,x+y);
        link(i,2*n+2,z,x-y);
        link(i,2*n+3,z,-x+y);
        link(i,2*n+4,z,-x-y);
    }
    fo(i,1,n)
    {
        int x,y,z;
        scanf("%d%d%d",&x,&y,&z);
        link(i+n,ed,z,0);
        link(2*n+1,i+n,z,-x-y);
        link(2*n+2,i+n,z,-x+y);
        link(2*n+3,i+n,z,x-y);
        link(2*n+4,i+n,z,x+y);
    }
    ans=0;
    while(spfa())
        flow(st,INF);
    printf("%lld\n",-ans);
}
```