11.12 am

	预估	实际
T1 **desire**	0	4
T2 **dealing**	50	60
T3 **lunatic**	10	21
T4 **season**	60	40

T1 desire

看了十几分钟样例，没看明白，就看下一去了，回来再来看的时候差不多只剩一个小时了，后面才发现是只能建立在叶子节点上面，不是任意子节点上，花了差不多半个小数，乱搞了一下树，对了个样例，喜提$4pts$

分析

正解是**DP** 但是鸽

T2 dealing

嗯，很好的倍增题，考场上以为是什么树形结构加并查集，首先想到同一操作中的两区间颜色相同，考虑并查集，合并为同一集合，后面手推了一下，不同操作间交叉或重合的情况，发现是一个集合合并的过程，又发现只有暴力是我能打的，想了半天也没想到如何优化区间内修改颜色的过程，后来发现是倍增

AC code&BF

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int maxn = 1000010;
const int p = 1e9+7;

template<typename T>void read(T &x)
{
    x=0; char c=getchar(); T neg=0;
    while(!isdigit(c)) neg|=!(c^'-'),c=getchar();
    while(isdigit(c)) x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(neg) x=(~x)+1;
}
template<typename T>void wr(T x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar((x-x/10*10)^48);
    return ;
}
```

```
int n,m,col[maxn],fa[maxn][22],Ans=1;
int cnt,vis[maxn];

int find(int x,int y)
{
    return fa[x][y]==x?x:fa[x][y]=find(fa[x][y],y);
}

void paint(int x,int y,int k)
{
    int U=find(x,k);
    int V=find(y,k);
    if(U!=V) fa[V][k]=U;//
}
void init()
{
    for(int i=1;i<=n;i++)
    {
        fa[i][0]=i;
        for(int j=1;j<=21;j++)
        {
            fa[i][j]=fa[i][j-1];
        }
    }
}

void down()
{
    for(int j=21;j>=1;j--)
    {
        for(int i=1;i<=n;i++)
        {
            int U=find(i,j);
            if(U!=i)
            {
                paint(U,i,j-1);
                paint(U+(1<<(j-1)),i+(1<<(j-1)),j-1);
            }
        }
    }
}

void merge(int x,int y,int len)
{
    int num=0;
    while(len)
    {
        if(len&1)
        {
            paint(x,y,num);
            x+=(1<<num);
            y+=(1<<num);
        }
        len>>=1; num++;
```

```
    }
}
signed main()
{
    freopen("dealing.in","r",stdin);
    freopen("dealing.out","w",stdout);
    read(n);  read(m);
    init();
    for(int i=1;i<=m;i++)
    {
        int len,x,y;
        read(len), read(x), read(y);
        merge(x,y,len);
    }
    down();
    for(int i=1;i<=n;i++)
    {
        if(!vis[find(i,0)])
        {
            Ans=(Ans*26)%p;
            cnt++; vis[find(i,0)]=1;
        }
    }
    wr(Ans);
    return 0;
}
/*
6 2
2 1 2
1 4 5

10 5
1 4 6
1 3 4
1 10 10
1 9 4
2 8 8

#include<bits/stdc++.h>
#define int long long
using namespace std;
const int maxn = 1000010;
const int p = 1e9+7;

template<typename T>void read(T &x)
{
    x=0; char c=getchar(); T neg=0;
    while(!isdigit(c)) neg|=!(c^'-'),c=getchar();
    while(isdigit(c)) x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(neg) x=(~x)+1;
}
template<typename T>void wr(T x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
```

```
    putchar((x-x/10*10)^48);
    return ;
}

int n,m,col[maxn],fa[maxn],Ans=1;
int cnt,vis[maxn];

int find(int x){ return fa[x]==x?x:fa[x]=find(fa[x]); }

void merge(int x,int y,int len)
{
    for(int i=0;i<len;i++)
    {
        int U=find(x+i);
        int V=find(y+i);
        if(U==V) continue;
        fa[V]=U;
        col[x+i]=find(x+i);
        col[y+i]=find(y+i);
    }
}
signed main()
{
    freopen("dealing.in","r",stdin);
    freopen("dealing.out","w",stdout);
    read(n);  read(m);
    for(int i=1;i<=n;i++)
    {
        fa[i]=i;
    }
    for(int i=1;i<=m;i++)
    {
        int len,x,y;
        read(len), read(x), read(y);
        merge(x,y,len);
    }
    for(int i=1;i<=n;i++)
    {
        cout<<i<<" "<<col[find(i)]<<endl;
        if(!col[find(i)])
        {
            Ans=(Ans*26)%p;
            cnt++; continue;
        }
        if(!vis[col[find(i)]])
        {
            Ans=(Ans*26)%p;
            cnt++; vis[col[find(i)]]=1;
        }
    }
//  cout<<cnt<<endl;
    wr(Ans);
    return 0;
}
```

```
*/
```

T3 lunatic

同样的，样例没看懂，就去整T4了，后面返回做的时候只剩差不多半小时了，就随便整了个贪心拍排序， 过了小样例，下来听谢量讲纯暴力可以拿$51pts$ 几乎正解，有点小亏

T4 season

血亏题，看到区间修改，啪的一声，很快啊，线段树（因为太弱，不会二维查分。 既然线段可以二分，那么矩形为什么不可以四分，写完线段树，调了半小时，结果是一个判断的区间写错了，写完线段树才发现，我的最小生成树是$O(n^2)$的效率，而且线段树四倍开不下

无奈，只有$40pts$,我的评价是不如暴力

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn = 2010;
const int p = 1e9+7;

template<typename T>void read(T &x)
{
    x=0; char c=getchar(); T neg=0;
    while(!isdigit(c)) neg|=!(c^'-'),c=getchar();
    while(isdigit(c)) x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(neg) x=(~x)+1;
}
template<typename T>void wr(T x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar((x-x/10*10)^48);
    return ;
}
int n,m,fa[maxn];
ll Ans;

struct node{
    int lx,ly,rx,ry;
    int ch_1,ch_2,ch_3,ch_4;
    ll tag,val;
}re[(maxn*maxn)*4+10];

struct ed{
    int u,v;
    ll w;
}edge[maxn*maxn+10];

bool cmp(ed a,ed b){ return a.w<b.w; }
int find(int x){ return x==fa[x]?x:fa[x]=find(fa[x]); }
int tot;
int build(int lx,int ly,int rx,int ry)
{
```

```
    tot++;
    int x=tot;
    re[x].lx=lx;
    re[x].ly=ly;
    re[x].rx=rx;
    re[x].ry=ry;
    if(lx==rx&&ly==ry)
    {
        re[x].val=0;
        return x;
    }
    int midx=(lx+rx)>>1;
    int midy=(ly+ry)>>1;

    if(rx==lx&&ly+1==ry)
    {
        re[x].ch_1=build(lx,ly,midx,midy);
        re[x].ch_2=build(lx,midy+1,midx,ry);
        return x;
    }
    if(ly==ry&&lx+1==rx)
    {
        re[x].ch_1=build(lx,ly,midx,midy);
        re[x].ch_3=build(midx+1,ly,rx,midy);
        return x;
    }
    re[x].ch_1=build(lx,ly,midx,midy);
    re[x].ch_2=build(lx,midy+1,midx,ry);
    re[x].ch_3=build(midx+1,ly,rx,midy);
    re[x].ch_4=build(midx+1,midy+1,rx,ry);
//  re[x].val=re[re[x].ch].val+re[re[x].rch].val;
    return x;

}
void down(int x)
{
    if(re[x].ch_1) re[re[x].ch_1].tag+=re[x].tag;
    if(re[x].ch_2) re[re[x].ch_2].tag+=re[x].tag;
    if(re[x].ch_3) re[re[x].ch_3].tag+=re[x].tag;
    if(re[x].ch_4) re[re[x].ch_4].tag+=re[x].tag;
    re[x].tag=0;
    return ;
}
ll calc(int x)
{
    return re[x].val+(re[x].rx-re[x].lx+1)*(re[x].ry-re[x].ly+1)*re[x].tag;
}

void up(int x)
{
    ll ans=0;
    if(re[x].ch_1) ans+=calc(re[x].ch_1);
    if(re[x].ch_2) ans+=calc(re[x].ch_2);
    if(re[x].ch_3) ans+=calc(re[x].ch_3);
    if(re[x].ch_4) ans+=calc(re[x].ch_4);
```

```
    re[x].val=ans;
}

void upd(int x,int Lx,int Ly,int Rx,int Ry,ll w)
{
    if(Lx<=re[x].lx&&Ly<=re[x].ly&&re[x].rx<=Rx&&re[x].ry<=Ry)
    {
//      re[x].val+=w;
        re[x].tag+=w;
        return ;
    }

    down(x);
    int midx=(re[x].lx+re[x].rx)>>1;
    int midy=(re[x].ly+re[x].ry)>>1;

    if(Lx<=midx&&Ly<=midy&&re[x].ch_1) upd(re[x].ch_1,Lx,Ly,Rx,Ry,w);
    if(Lx<=midx&&Ry>midy&&re[x].ch_2)  upd(re[x].ch_2,Lx,Ly,Rx,Ry,w);
    if(Ly<=midy&&Rx>midx&&re[x].ch_3)  upd(re[x].ch_3,Lx,Ly,Rx,Ry,w);
    if(Rx>midx&&Ry>midy&&re[x].ch_4)   upd(re[x].ch_4,Lx,Ly,Rx,Ry,w);

    up(x);
    return ;
}


ll ask(int x,int Lx,int Ly,int Rx,int Ry)
{
    if(Lx<=re[x].lx&&Ly<=re[x].ly&&re[x].rx<=Rx&&re[x].ry<=Ry)
    {
        return calc(x);
    }
    ll ans=0;
    down(x);
    int midx=(re[x].lx+re[x].rx)>>1;
    int midy=(re[x].ly+re[x].ry)>>1;

    if(Lx<=midx&&Ly<=midy&&re[x].ch_1) ans+=ask(re[x].ch_1,Lx,Ly,Rx,Ry);
    if(Lx<=midx&&Ry>midy&&re[x].ch_2)  ans+=ask(re[x].ch_2,Lx,Ly,Rx,Ry);
    if(Ly<=midy&&Rx>midx&&re[x].ch_3)  ans+=ask(re[x].ch_3,Lx,Ly,Rx,Ry);
    if(Rx>midx&&Ry>midy&&re[x].ch_4)  ans+=ask(re[x].ch_4,Lx,Ly,Rx,Ry);
//  up(x);
    return ans;
}
int cnt;

signed main()
{
    freopen("season.in","r",stdin);
    freopen("season.out","w",stdout);
    read(n); read(m);
    for(int i=1;i<=n;i++)  fa[i]=i;

    build(1,1,n+2,n+2);
//  cout<<"strt   "<<endl;
```

```
    for(int i=1;i<=m;i++)
    {
        int a,b,c,d;
        ll w;
        read(a); read(b); read(c); read(d); read(w);
        upd(1,a,c,b,d,w);
    }
//  cout<<"buid tree "<<endl;
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
        {
            edge[++cnt].u=i;
            edge[cnt].v=j;
            edge[cnt].w=ask(1,j,i,j,i)+ask(1,i,j,i,j);
//          cout<<i<<" "<<j<<" "<<ask(1,i,j,i,j)<<endl;
//          cout<<j<<" "<<i<<" "<<ask(1,j,i,j,i)<<endl;
        }
    }
//  cout<<"sort  "<<endl;

    int j=0,i=0;
    sort(edge+1,edge+cnt+1,cmp);
    while(1)
    {
        if(j==n-1) break;
        i++;
        int U=find(edge[i].u);
        int V=find(edge[i].v);
        if(U==V) continue;
        fa[U]=V; Ans+=edge[i].w; j++;
    }
    wr(Ans);
    return 0;
}
/*
5 5
3 3 4 5 -10
1 2 3 4 20
4 4 5 5 -10
2 2 4 4 -20
1 1 2 4 0

6 8
1 3 6 6 3
4 4 6 6 10
3 3 5 6 -8
1 2 5 5 -7
1 2 6 6 -1
1 3 4 5 6
3 5 6 6 7
2 3 6 6 3
*/
```

总结

怎么说呢，该打的暴力基本都打了，不该打的也打了，对于一些之前的优化思路，长时间不碰还是会有遗忘，就像T2的倍增操作，对于T1,T3 搓不明白样例，基本都是题意的理解不够清晰，搓不出来样例就换个思路，万一题解说的就是另一个意思呢。对于T4，这种分了两个阶段的问题，一定先考虑好全局的复杂度，不要写了一半才发现自己后半段时间复杂度不对。