2022.11.13

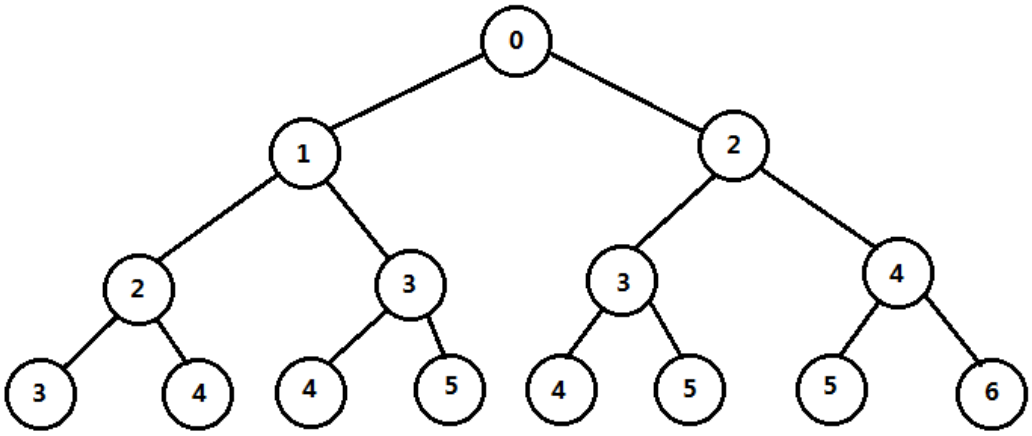
题目	预估分数	实际分数	差值
desire	0	8	+8
dealing	60	60	0
lunatic	0	14	+14
season	30	40	+10

T1 desire

题解中提到的性质“**第一小的 dep 对应第一大的 val ，第二小的 dep 对应第二大的 val ，以此类推**”，是不难发现的，然后我就卡在了如何确定深度序列。

考场上主要面对的问题是，叶子节点的位置不一定在二叉数的同一层，并且先建左子树还是先建右子树都会对最终的 dep 产生影响。

所以考虑拿部分分，赌部分数据点的叶子节点都在最底层。因为同一层的 dep 有鲜明的大小性质



观察每一层的 dep ，可以看到它们整体是递增的。所以把最底层的所有数累计起来，从小到大排序，再与 val 大小配对，喜提8pts.

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#pragma GCC optimize(2)
using namespace std;
typedef long long ll;
const ll N=1e5+5;
ll n,dep,cnt,ans;
ll a[N],len[N];
inline void build(ll p,ll val,ll d)
{
    if(d==dep)
    {
```

```
        len[++cnt]=val;
        return ;
    }
    build(p<<1,val+1,d+1);
    build(p<<1|1,val+2,d+1);
}
inline bool cmp(ll a,ll b)
{
    return a>b;
}
signed main()
{
    freopen("desire.in","r",stdin);
    freopen("desire.out","w",stdout);
    scanf("%lld",&n);
    for(register ll i=1;i<=n;i++) scanf("%lld",&a[i]);
    ll tmp=n;
    for(register ll i=1;;i++)
    {
        if(1<<(i-1)>=n)
        {
            dep=i;
            break;
        }
    }
    build(1,0,1);
    sort(len+1,len+cnt+1);
    sort(a+1,a+n+1,cmp);
    for(register ll i=1;i<=n;i++)
    {
        ans+=a[i]*len[i];
    }
    printf("%lld\n",ans);
    return 0;
}
```

T2 dealing

对于一个长度为 n 的串，如果不加任何限制，每个位置能有 26 种可能（即小写字母个数），那么串的种类就有 26^n 种。

现在加上限制，打个比方，要求 2 和 3 这两个位置上的字符是相同的。那么，如果 2 号位上的是 a ，3 号位上的也要是 a ，这两个位置就可以当作一个位置处理了。

把所有要求字符相同的位置都当作一个位置，若这时的位置数为 cnt ,那么串的种类数就是 26^{cnt} .

现在就变成了求 cnt，本蒟蒻选择用**并查集**这种暴力的方法来维护，喜提60pts.

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
typedef long long ll;
```

```
const ll mod=1e9+7;
const ll N=1e6+5;
ll n,m,cnt,ans;
ll fa[N];
inline ll ksm(ll a,ll b)
{
    ll res=1;
    for(;b;b>>=1)
    {
        if(b&1) res=1ll*res*a%mod;
        a=1ll*a*a%mod;
    }
    return res;
}
inline ll got(ll x)
{
    if(x==fa[x]) return x;
    return fa[x]=got(fa[x]);
}
signed main()
{
    freopen("dealing.in","r",stdin);
    freopen("dealing.out","w",stdout);
    scanf("%lld%lld",&n,&m);
    cnt=n;
    for(register ll i=1;i<=n;i++) fa[i]=i;
    while(m--)
    {
        ll len,x,y;
        scanf("%lld%lld%lld",&len,&x,&y);
        for(register ll i=1;i<=len;i++)
        {
            ll p1=x+i-1;
            ll p2=y+i-1;
            ll r1=got(p1);
            ll r2=got(p2);
            if(r1==r2) continue;
            cnt--;
            fa[r2]=r1;
        }
    }
    ans=ksm(26,cnt);
    ans%=mod;
    printf("%lld\n",ans);
    return 0;
}
```

T3 lunatic

考虑一种贪心策略，将所有的线段按照长度从大到小排列。现有 k 个组，将长度排在前 $k-1$ 的线段每个单独分一组，而剩下的所有线段全部分进一组。喜提14pts.

```
#include<iostream>
```

```cpp
#include<cstdio>
#include<cstring>
#include<algorithm>
#pragma GCC optimize(2)
using namespace std;
typedef long long ll;
const ll N=1e5+5;
ll n,k,g,ans;
ll minr=1e18,maxr=-1;
ll tong[N*10];
struct Segment
{
    ll l,r,id,len;
}s[N];
inline bool cmp(Segment a,Segment b)
{
    return a.len>b.len;
}
inline ll minn(ll a,ll b)
{
    return a<b?a:b;
}
inline ll maxx(ll a,ll b)
{
    return a>b?a:b;
}
signed main()
{
    freopen("lunatic.in","r",stdin);
    freopen("lunatic.out","w",stdout);
    scanf("%lld%lld",&n,&k);
    g=n-k+1;
    for(register ll i=1;i<=n;i++)
    {
        scanf("%lld%lld",&s[i].l,&s[i].r);
        s[i].id=i;
        s[i].len=s[i].r-s[i].l;
    }
    for(register ll i=1;i<k;i++)
    {
        ans+=s[i].len;
    }
    for(register ll i=k;i<=n;i++)
    {
        if(s[i].l==s[i].r) continue;
        minr=minn(minr,s[i].l);
        maxr=maxx(maxr,s[i].r-1);
        for(ll j=s[i].l;j<s[i].r;j++) tong[j]++;
    }
    for(register ll i=minr;i<=maxr;i++)
    {
        if(tong[i]==g) ans++;
    }
    printf("%lld\n",ans);
    return 0;
```

```
}
```

T4 season

直接按照题意模拟。

开一个 $n \times n$ 的二维数组 A（显然 n 开不满，不过本蒟蒻不指望能拿满分。况且可能空间没用完时间就先超限了，所以开 5000×5000 就很好），然后根据题意建边，跑 $krus$。喜提40pts.

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#pragma GCC optimize(2)
using namespace std;
typedef long long ll;
const ll N=5000;
ll n,m,cnt,ans;
ll A[N][N];
struct Edge
{
    ll u,v,w;
}e[N*N];
ll fa[N*N];
inline bool cmp(Edge a,Edge b)
{
    return a.w<b.w;
}
inline ll got(ll x)
{
    if(x==fa[x]) return x;
    return fa[x]=got(fa[x]);
}
signed main()
{
    freopen("season.in","r",stdin);
    freopen("season.out","w",stdout);
    scanf("%lld%lld",&n,&m);
    while(m--)
    {
        ll a,b,c,d,W;
        scanf("%lld%lld%lld%lld%lld",&a,&b,&c,&d,&W);
        for(register ll i=a;i<=b;i++)
            for(register ll j=c;j<=d;j++)
                A[i][j]+=W;
    }
    for(register ll i=1;i<n;i++)
    {
        for(register ll j=i;j<=n;j++)
        {
            cnt++;
            e[cnt].u=i;
            e[cnt].v=j;
```

```
            e[cnt].w=A[i][j]+A[j][i];
        }
    }
    sort(e+1,e+cnt+1,cmp);
    for(register ll i=1;i<=n;i++) fa[i]=i;
    for(register ll i=1;i<=cnt;i++)
    {
        ll u=got(e[i].u);
        ll v=got(e[i].v);
        if(u==v) continue;
        fa[u]=v;
        ans+=e[i].w;
    }
    printf("%lld\n",ans);
    return 0;
}
```

总结

一、骗分很重要

纵观整场考试，没有一道题打了正解。T1赌数据点，T2暴力，T3毫无正确性的贪心，T4部分分暴力。但总分居然还有点好看（？）

二、那就跳

为了找到T1中那个统计 dep 的方法，我用来大概一个半小时（然后还没想出来）。选择跳T2,秒杀了那个极其好打的并查集暴力，状态一下就回来了。

三、自我主宰

明知道自己拿不到满分，就不要企图用满分的眼光来审题。

像T4，二维数组要是开出来肯定不能开满。但我深知自己拿不到满分，二维数组就嗯开个部分分，稳拿40美滋滋。

四、万一呢

说不定有些数据点就有某种独特的性质能让不具有正确性的算法过呢。