一眼线段树，虽然能过，不过太傻X了233

每次我们摧毁了一个区间，下一次如果还要摧毁这个区间或者它的子区间的话，我们就不用处理了。这样我们可以把每一个区间抽象成一个点，用并查集来维护。合并时将[L,R]区间全部合并，然后n--。这样每个点只会被合并一次，复杂度O(nα(n))

这道题据说可以用线段树做。然而一看，如果我们把一个颜色的点用并查集归到一块儿，用一个while循环，如果左右两端点颜色不一样，就依次把从右端点向左合并father数组。注意要加上路径压缩。

父节点都标记成x-1的父节点

因为如果标记成[2,8]当中任何一个点的话，最终这些点的父节点还是一个独立，不同于其他元素的结点

比如[3,3]染色，染和不染没有区别

#include <cstdio>

#include <cstring>

#include<iostream>

#include<algorithm>

using namespace std;

int fa[200010];

int Find(int x)

{

if(fa[x]==x)return x;

return fa[x]=Find(fa[x]);//是否需要带路径压缩还要考虑一下,有时带了会超时

//return Find(f[x]); //不路径压缩版

}

int main()

{

int n,m,l,r;

while(cin>>n>>m)

{

memset(fa,0,sizeof(fa));

for(int i=0;i<=n;i++)

fa[i]=i;

int ans=n;

while(m--)

{

cin>>l>>r;

while(Find(r)!=Find(l-1))

{

fa[Find(r)]=fa[Find(r)-1];

ans--;

}

cout<<ans<<endl;

}

}

return 0;

}

第二种写法：

#include <cstdio>

#include <cstring>

#include<iostream>

#include<algorithm>

using namespace std;

int fa[200010];

bool b[200010];

int Find(int x)

{

if(fa[x]==x)return x;

return fa[x]=Find(fa[x]);//是否需要带路径压缩还要考虑一下,有时带了会超时

//return Find(f[x]); //不路径压缩版

}

int main()

{

int n,m,l,r;

while(cin>>n>>m)

{

memset(fa,0,sizeof(fa));

for(int i=0;i<=n;i++)

{

fa[i]=i;

b[i]=1;

}

int ans=n;

while(m--)

{

cin>>l>>r;

for(int j=r;j>=l;j--)

if(b[j])

{

ans--;

b[j]=0;

fa[j]=l;

}

else j=fa[j];

cout<<ans<<endl;

}

}

return 0;

}