注意不能i=x;i<=y;i++这样去染

1-2染了

3-4染了

其实还有2-3没染

但如果这么操作的话就会认为是染了

i=x;i<y;i++

判断有多少个染色段的时候b[i]!=b[i-1]这么判断

不能b[i]！=b[now]（now是上一个段被染过色的下标）

因为如果像1-2倍染红色，2-4没染，4-5再染红色，这样红色就是两段，但那么判断的话就成一段了

要注意它给你的染色的线段的左右端点并不是<=n-1的，题中并没有这么说，只是样例给的凑巧了而已

#include <cstdio>

#include <cstring>

#include <iostream>

#include<set>

#include<map>

using namespace std;

int b[8010];

int main()

{

//freopen("input.txt","r",stdin);

int n;

while(~scanf("%d",&n))

{

int x,y,z;

memset(b,-1,sizeof(b));

for(int i=1;i<=n;++i)

{

scanf("%d%d%d",&x,&y,&z);

for(int j=x;j<y;++j)

b[j]=z;

}

map<int,int>\_map;

int i=0;

while(b[i]==-1)i++;

\_map[b[i]]++;

++i;

for(;i<=8000;++i)

if(b[i]!=-1 && b[i]!=b[i-1])

++\_map[b[i]];

map<int,int>::iterator it;

for(it=\_map.begin();it!=\_map.end();++it)

cout<<it->first<<' '<<it->second<<endl;

cout<<endl;

}

return 0;

}

线段树：

#include <cstdio>

#include <cstring>

#include <iostream>

#include<set>

#include<map>

using namespace std;

int a[8010\*4];

int color[8010\*2];

inline void push\_down(int root)

{

if(a[root]!=-1)

{

a[root<<1]=a[root<<1|1]=a[root];

a[root]=-1;

}

}

inline void update(int root,int l,int r,int x,int y,int z)

//在(l,r)中找(x,y)区间染色

{

if(x<=l && r<=y)

{

a[root]=z;

return;

}

if(a[root]==z)return ;

if(a[root]!=-1)push\_down(root);

int mid=(l+r)>>1;

if(x<=mid)

update(root<<1,l,mid,x,y,z);

if(y>mid)

update(root<<1|1,mid+1,r,x,y,z);

}

inline void query(int root,int l,int r)

{

if(a[root]>=0)

{

for(int i=l;i<=r;++i)

color[i]=a[root];

return;

}

if(l!=r && a[root]==-1)

{

int mid=(l+r)>>1;

query(root<<1,l,mid);

query(root<<1|1,mid+1,r);

}

}

int main()

{

//freopen("input.txt","r",stdin);

int n;

while(~scanf("%d",&n))

{

int x,y,z;

memset(a,-1,sizeof(a));

for(int i=1;i<=n;++i)

{

scanf("%d%d%d",&x,&y,&z);

if(x>=y)continue;

update(1,1,8000,x+1,y,z);

}

memset(color,-1,sizeof(color));

query(1,1,8000);

map<int,int>\_map;

map<int,int>::iterator it;

int i=1;

while(color[i]==-1)++i;

++\_map[color[i]];

++i;

for(;i<=8000;++i)

if(color[i]!=-1 && color[i]!=color[i-1])

{

++\_map[color[i]];

}

for(it=\_map.begin();it!=\_map.end();++it)

cout<<it->first<<' '<<it->second<<endl;

cout<<endl;

}

return 0;

}