题意

区间值覆盖，求整个区间不同数字个数

思路

1.离散化

将数字放进数组

排序

用lower_bound()找到x对应的位置，即为x离散化后的值

2.update()

某区间先被x覆盖，然后被y覆盖，等效于直接被y覆盖

所以更新标记直接覆盖

当前操作区间被完全覆盖，打上标记直接返回

否则将该区间标记下推，向下继续

3.query()

某一区间，如果该区间只有一种颜色d，则这个点被打了标记d

如果当前区间有标记，判断该颜色是否出现过，没有答案就+1有就忽略，然后返回上一层，不必继续往下

如果当前区间标记为0，继续往下，直到叶子结点返回

4.注意点

离散化的时候，如果相邻两个值，相差大于1，就在这两个值间多插一个值

这个题，输入的(x,y)为10000，所以最终的结点数最多为40000，

代码

```
#include<iostream>
#include<map>
#include<cstdio>
#include<set>
#include<algorithm>
#include<cstring>
#include<vector>
#define For(i,a,b) for(int i=(a); i<=(b) ; i++)
#define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
#define Memset(a,b); memset((a),(b),sizeof((a)));
#define Cin(a); scanf("%d",&(a));
#define Cinc(a); scanf(" %c",&(a));
```

```cpp
#define Cins(a); scanf("%s",(a));
#define Cout(a,b);  printf("%d",(a));printf(b);
#define Coutc(a,b);  printf("%c",(a));printf(b);
#define Couts(a,b);  printf("%s",(a));printf(b);
using namespace std;
typedef  long long LL;
typedef  unsigned long long ULL;
typedef  long double LDB;
inline int readint() {int x;cin>>x;return x;}
int lisan[200000];
int n,x[20005],y[20005];
bool book[200000];
int lazy[200000];
void pushdown(int o)
{
    lazy[o<<1] = lazy[o];
    lazy[o<<1|1] = lazy[o];
    lazy[o] = 0;
}
void update(int o,int l,int r,int L,int R,int d)
{
    if(l>=L&&r<=R)
    {
        lazy[o] = d;
        return ;
    }
    if(lazy[o]>0) pushdown(o);
    int M = (l+r)/2;
    if(M>=L)
        update(o<<1,l,M,L,R,d);
    if(M+1<=R)
        update(o<<1|1,M+1,r,L,R,d);
}
int ans = 0;
void query(int o,int l,int r)
{
    if(lazy[o]!=0)
    {
        if(!book[lazy[o]])
        {
            ans++;
            book[lazy[o]] = true;
        }
        return ;
    }
    if(l==r) return;
```

```
    int M = (l+r)>>1;
    query(o<<1,l,M);
    query(o<<1|1,M+1,r);
}
int main()
{
    int _;
    Cin(_);
    while(_--)
    {
        Cin(n);
        memset(book,false,sizeof(book));
        memset(lazy,0,sizeof(lazy));
        int tot = 0;
        For(i,1,n)
        {
            Cin(x[i]);Cin(y[i]);
            lisan[++tot] = x[i];
            lisan[++tot] = y[i];
        }
        sort(lisan+1,lisan+tot+1);
        int m = unique(lisan+1,lisan+tot+1)-lisan;
        int t=m;
        for(int i=2;i<=t;i++)
        {
            if(lisan[i]>lisan[i-1]+1)
                lisan[++m]=lisan[i-1]+1;
        }
        sort(lisan+1,lisan+m+1);
        For(i,1,n)
        {
            int xx = lower_bound(lisan,lisan+m,x[i]) - lisan;
            int yy = lower_bound(lisan,lisan+m,y[i]) - lisan;
            update(1,1,m,xx,yy,i);
        }
        ans = 0;
        query(1,1,m);
        printf("%d\n",ans);
    }
}
```