差分数组

数列游戏 NKOJ3754

给定一个长度为N的序列：

首先进行X次操作，每次操作在Li和Ri这个区间加上一个数Ci。

然后进行Y次询问，每次询问Li到Ri的区间和。

初始序列都为0。

1<=N<=1000000,1<=X<=N, X<=Y<=N

1<=Li<=N,Li<=Ri<=N,|Ci|<=100000000000000

线段树裸题！
有没有其它简便的解法？

差分数组！

差分数组(差分数列)

对于一个数组A[ ]，其差分数组D[i]=A[i]-A[i-1] (i>0)且D[0]=A[0]

令SumD[i]=D[0]+D[1]+D[2]+…+D[i] （SumD[ ]是差分数组D[ ]的前缀和）

则**SumD[i]**=A[0]+A[1]-**A[0]**+A[2]-**A[1]**+A[3]-**A[2]**+…+A[i]-**A[i-1]**=**A[i]**

即A[i]的差分数组是D[i]，而D[i]的前缀和是A[i]

对于“数列游戏”这题:

如果每次修改都修改从L到R的值的话，一定会TLE。

注意特殊处：这道题是先进行整体区间修改，最后才统一查询。

所以，我们只要维护一个差分数组就行了。

维护差分数组，对于将区间[L,R]加C，我们只需要将**D[L]+C**和**D[R+1]-C**

当修改完毕后，我们先求一遍差分前缀和就得到了修改后的数组A[ ]，

然后再对A[ ]求一遍前缀和，这样每次查询的时候只要计算一次就可以得到结果了

//参考代码

```
cin>>N>>X>>Y;
for(i=1;i<=X;i++)
{
	cin>>L>>R>>C;
	D[L]=D[L]+C;
	D[R+1]=(D[R+1]-C);
}
for(i=1;i<=N;i++)A[i]=(D[i]+A[i-1]);                //D[i]=A[i]-A[i-1];
for(i=1;i<=N;i++)SumA[i]=SumA[i-1]+A[i];
for(i=1;i<=N;i++)
{
	cin>>L>>R;
	cout<<SumA[R]-SumA[L-1]<<endl;
}
```

数列操作 NKOJ3786

给定一个长度为n的数列{ A1,A2...An }，每次可以选择一个区间[L,R]，使这个区间内的数都+1或者都-1。

问至少需要多少次操作才能使数列中的所有数都一样,并求出在保证最少次数的前提下,最终得到的数列有多少种。

n=100,000，0<=Ai<2147483648

数列操作 解题分析

由于操作都是区间操作，那么区间里的数相对差值是不变的，那么我们可以考虑构建 **差分数组:D[i]=A[i]-A[i-1]**

显然，当**D数组除了第1个元素外所有其他元素都为0**时，A数组才会满足题目要求。
即最后所有A数组中所有元素的值都是A[1]

讨论一次操作对D数组的影响：

假设对A数组的区间**[L,R]整体+1**，那么只会使D数组的元素**D[L]+1**，元素**D[R+1]-1**，其它保持不变；
假设对A数组的区间**[L,R]整体-1**，那么只会使D数组的元素**D[L]-1**，元素**D[R+1]+1**，其它保持不变；

我们想要将D数组的[2,n]都调整为0
对于D数组，一次操作可以使“**D[i]+1同时D[j]-1**”,即i,j两个元素对消
也可以使“**D[1]+1同时D[i]-1**”或者 “**D[1]-1同时D[i]+1**”，即i与1对消。
也可以使“**D[n+1]+1同时D[i]-1**”或者 “**D[n+1]-1同时D[i]+1**”，即i与n+1对消。
也就是对于D[2..n]**一次操作可以同时改变两个元素的值，也可以改变一个元素的值**

所以对于第1问，只需计算D数组中所有**正数的总和Sum1**与所有**负数的总和Sum2**
答案所求的**最少操作次数=max(Sum1,|Sum2|)**

数列操作 解题分析

讨论第2问，求数列的方案数：

在第1问，已计算出了D数组中所有**正数的总和Sum1**与所有**负数的总和的绝对值Sum2**

将D[2...n]中的i,j配对消除所需次数为min{Sum1,Sum2}

此后D数组中还剩|Sum1-Sum2|个数没有变为0，现在可以与D[1]或D[n+1]配对消除操作次数为|Sum1-Sum2|，所以最终D[1]的取值可能有|Sum1-Sum2|+1种。

所以，最终第2问的答案为|Sum1-Sum2|+1

```
int main()
{
    ......
    for(i=n;i>1;i--)
        if(a[i]-a[i-1]>0)  Sum1+=a[i]-a[i-1];
        else   Sum2+=a[i-1]-a[i];
    cout<<max(Sum1,Sum2)<<endl
    cout<<abs(Sum1-Sum2)+1<<endl;
}
```

借教室 NKOJ1887

　　我们要处理接下来n天的借教室信息，其中第i天学校有ri个教室可供租借。共有m份订单，每份订单用三个正整数描述，分别为dj, sj, tj，表示某租借者需要从第sj天到第tj天租借教室（包括第sj天和第tj天），每天需要租借dj个教室。

　　对于每份订单，我们只需要每天提供dj个教室，而它们具体是哪些教室，每天是否是相同的教室则不用考虑。

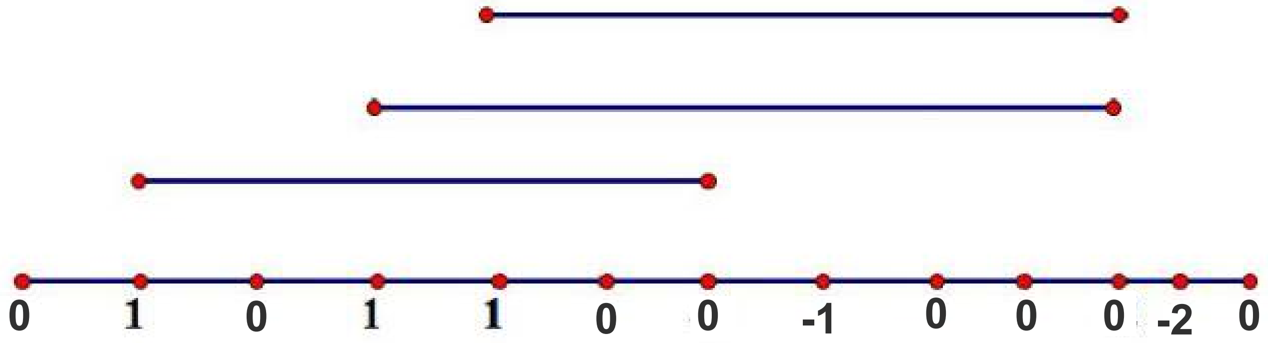
　　借教室的原则是先到先得，也就是说我们要按照订单的先后顺序依次为每份订单分配教室。如果在分配的过程中**一旦遇到一份订单无法完全满足，则需要停止所有教室的分配**，通知当前申请人退单。

　　“无法满足”指从第sj天到第tj天中有至少一天剩余的教室数量不足dj个。现在我们需要知道，是否会有订单无法完全满足。如果有，需要通知哪一个申请人。

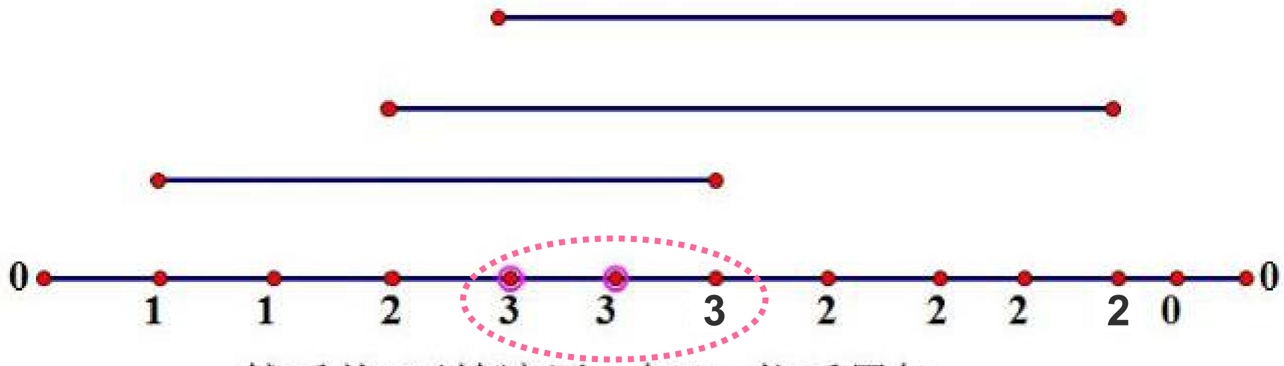
$1 \le n, m \le 10^6, 0 \le ri, dj \le 10^9, 1 \le sj \le tj \le n$

我们用**差分数组**来解决它！

技巧：求被所有线段都覆盖过的点？



可以考虑区间求交的方法：新建一个flag数组，区间开头+1，结尾-1



然后从1开始遍历，把flag往后累加，如果当前点的flag等于区间的个数，它就是所有区间交集中的一个点

借教室 解题分析

显然可以把每个要求订单看作是一条线段，然后每条线段都有厚度（需要的教室数量），叠加后判断是否超过可以承受的即可。

于是设置差分数组S[i]
对于一个订单[Li,Ri]区间每天需要Di间教室: S[Li]+Di，S[Ri+1]-Di
最后从左往右扫描一遍，第i天需要的教室数Need[i]=Need[i-1]+S[i]
若第i天可用的教室A[i]<Need[i],则无法满足！

具体实现时采用二分答案的方式，二分[1...mid]号订单是否满足，用差分数组去验证。
时间复杂度O(nlogn)

//参考代码

```
main()
{
  //二分答案
    int L=1,R=m;
    while (L<=R)
    {
        mid=(L+R)>>1;
        if (!judge(mid))ans=mid,R=mid-1;
        else L=mid+1;
    }
    if (!ans) printf("0\n");
    else printf("-1\n%d\n",ans);
}
```

```
bool judge(int mid)
{
    memset(b,0,sizeof(b));
    int Need=0;
    for (i=1; i<=mid; i++) //构造差分数组
    { S[L[i]]+=D[i]; S[R[i]+1]-=D[i];}
    for(i=1; i<=n; i++)
    {
        Need+=S[i];
        if (Need>A[i]) return false;
    }
    return true;
}
```