经过转化后，问题变成，给出若干条线段，求这些线段和起点为l，终点为r的线段最大相交长度。

~~铭神说没见过这道题，我就愉快的出了= =~~

因为强制在线，所以卡掉了一波离线。。。。

***但是我大可持久还是卡不掉的，所以第一种做法就是可持久话线段树：***

我们把每条线段拆成两个点，用pair型保存（r, l）,r表示某条线段的右端点，l表示左端点，然后对它sort, 我们对每个位置都建一棵线段树。

这颗线段树维护的东西是(max\_len, maxr):

max\_len: 对于i这个位置，如果对于线段j, 它的r<=i, 那么我们就去把max\_len更新为r-l+1, 不然就是 - inf.(对于rt[i]这棵线段树)

maxr: 表示这个区间的最大的r

然后对于询问的区间[pl, pr]，我们分成三次去询问：

1. 询问rt[pr]这棵线段树在[1, pl]上的maxr
2. 询问rt[pr]这棵线段树二分寻找在[pl, pr]最小的一个位置x,它的maxr>pr
3. 询问rt[pr]这棵线段树在[pl,pr]上的max\_len

然后3次去最大即可，因为可能碰到询问恰巧没有一个从者能打败区间任意一只怪物的情况，所以还要和0取最大。

***最近欧气爆发的仓老师，写暴力对拍时写着写着想出的做法，也是一种在线的做法，很漂亮，~~但是我大可持久就是跑得快~~：***

首先开两个st表，分别维护两个数组：

L[i]: 以i为右端点的最小的左端点，

R[i]: 以i为左端点的最大的右端点，

分别用两个st表去维护。

然后对于询问区间[pl, pr]时，我们可以分别去查两个st表[1, pl], [pr, n]，去更新ans

然后我们只要好好的去处理在[pl, pr]这个区间内的情况：

苍老师的用线段树维护了两个值(max\_len, maxr):

max\_len: 表示在区间中[l, r]所有以l为起点的线段的最大的长度

maxr: 表示max\_len对应的最大的r

然后在线段树区间中寻找时，有两个剪枝规则，设ans为：

1. If (ans>T[o].maxlen) return;

If (T[o].maxr>=pr) ans = max(ans, T[o].maxlen-(T[o].maxr-pr))

else ans = max(ans, T[o].maxlen)

然后我们可以发现，因为先搜左节点

1. 如果T[o].maxr>=pr，那么当前左节点的父亲的整棵右子树就被删掉了
2. 反之，T[o].maxlen的必定完整的存在于[pl, pr]那么，就不用递归下去了，然后就跑去父亲的右儿子了，然后。。。

你会发现利用这个方法，每次规模都能减少一办，那么就是一个完美的log()的做法了。