[BZOJ 4653 [Noi2016]区间（Two pointers+线段树）](https://www.cnblogs.com/forever97/p/bzoj4653.html)

【题目链接】 <http://www.lydsy.com/JudgeOnline/problem.php?id=4653>

【题目大意】

　　在数轴上有n个闭区间 [l1,r1],[l2,r2],...,[ln,rn]。  
　　现在要从中选出m个区间，使得这m个区间共同包含至少一个位置。  
　　对于一个合法的选取方案，它的花费为被选中的最长区间长度减去被选中的最短区间长度。  
　　区间[li,ri]的长度定义为它的右端点的值减去左端点的值。  
　　求所有合法方案中最小的花费。如果不存在合法的方案，输出-1。

【题解】

　　我们将所有的区间按照长度从小到大排序，那么我们枚举左端点和右端点，  
　　将区间内的所有区间在线段树上更新1，  
　　那么当有个点被覆盖m次的时候这个区间就可以用来更新答案，该操作只要求最大值即可。  
　　区间的最大值对于左右端点的枚举具有单调性，  
　　可以用Twopointers来实现，前后指针各移动n次，  
　　复杂度O(nlogn)。

【代码】

1. #include <cstdio>
2. #include <algorithm>
3. #include <cstring>
4. using namespace std;
5. const int N=2000000;
6. struct node{int l,r,a,b,tag,max;}T[N];
7. const int INF=0x3f3f3f3f;
8. int tot,n,m,l,r,c;
9. void addtag(int x,int tag){
10. T[x].tag+=tag;
11. T[x].max+=tag;
12. }
13. void pb(int x){
14. if(T[x].l){addtag(T[x].l,T[x].tag);addtag(T[x].r,T[x].tag);}
15. T[x].tag=0;
16. }
17. void up(int x){T[x].max=max(T[T[x].l].max,T[T[x].r].max);}
18. void build(int l,int r){
19. int x=++tot;
20. T[x].a=l;T[x].b=r;T[x].tag=T[x].l=T[x].r=T[x].max=0;
21. if(l==r)return;
22. int mid=(l+r)>>1;
23. T[x].l=tot+1;build(l,mid);
24. T[x].r=tot+1;build(mid+1,r);
25. up(x);
26. }
27. void change(int x,int a,int b,int p){
28. if(T[x].a>=a&&T[x].b<=b){addtag(x,p);return;}
29. if(T[x].tag)pb(x); int mid=(T[x].a+T[x].b)>>1;
30. if(mid>=a&&T[x].l)change(T[x].l,a,b,p);
31. if(mid<b&&T[x].r)change(T[x].r,a,b,p);up(x);
32. }
33. int cnt,disc[N<<1];
34. struct data{int l,r,len;}p[N];
35. bool cmp(data a,data b){return a.len<b.len;}
36. int main(){
37. scanf("%d%d",&n,&m);
38. for(int i=1;i<=n;i++){
39. scanf("%d%d",&p[i].l,&p[i].r); p[i].len=p[i].r-p[i].l;
40. disc[++cnt]=p[i].l; disc[++cnt]=p[i].r;
41. }sort(disc+1,disc+cnt+1);
42. cnt=unique(disc+1,disc+cnt+1)-disc-1;
43. build(1,cnt);
44. for(int i=1;i<=n;i++){
45. p[i].l=lower\_bound(disc+1,disc+cnt+1,p[i].l)-disc;
46. p[i].r=lower\_bound(disc+1,disc+cnt+1,p[i].r)-disc;
47. }sort(p+1,p+n+1,cmp); int pt=0,ans=INF;
48. for(int i=1;i<=n;i++){
49. while(T[1].max<m&&pt<n){
50. pt++; change(1,p[pt].l,p[pt].r,1);
51. }if(T[1].max==m)ans=min(ans,p[pt].len-p[i].len);
52. change(1,p[i].l,p[i].r,-1);
53. }if(ans==INF)puts("-1");
54. else printf("%d\n",ans);
55. return 0;
56. }