```
1  Cover(L[1,n],R[1,n]):
2      priority queue P,Q,A
3      for i=1 to n.............................初始化O(n)
4          l[L[i]]=i,r[R[i]]=i//便于根据L，R的值索引index
5          P.enque(L[i]),Q.enque(R[i])
6      while(!P.empty())
7          x=P.ExtractMin(),y=Q.ExtractMax()
8          if R[l[x]]<=L[r[y]]//确保选出的两个不冲突
9              A.add(l[x]),A.add(r[y])//A里记录被选择的区间的序号
10             for i=1 to n//去除所有与已选冲突的.....................O(n)
11                 if L[i]<R[r[x]] and i!=r[x]
12                     P.dequeue(L[r[x]]), Q.dequeue(R[r[x]])
13                 if R[i]>L[l[y]] and i!=l[y]
14                     P.dequeue(L[l[y]]), Q.dequeue(R[l[y]])
```

时间

正确性

- 由于2n个endpoints都不同，所以可以建立l[L[i]]=i,r[R[i]]=i这种从数组值来反过来索引index的数组
- 每次选择的最先开始的和最后结束的这两个，都是不选不行的，而且没有选择其他的，只选了这两个
- 每次选择后，都在剩余的中把和这两个冲突的删去，下个循环选出的两个不会出现冲突
- 最后一次选择，为了防止被选出的两个有冲突，需要判断（第8行）$E=mc2$