

将每个区间按照右端点从小到大进行排序

从前往后枚举区间，end值初始化为无穷小

如果本次区间不能覆盖掉上次区间的右端点， ed < range[i].l

说明需要选择一个新的点， res ++ ; ed = range[i].r;

如果本次区间可以覆盖掉上次区间的右端点，则进行下一轮循环

算法1

按照右端点排序

C++ 代码

#include<iostream>

#include<algorithm>

#include<cstring>

using namespace std;

const int N=1e5+10;

struct Range

{

int l,r;

bool operator< (const Range &W)const

{

return r<W.r;

}

}ranges[N];

int main()

{

int n;

cin>>n;

for(int i=0;i<n;i++)

{

int l,r;

scanf("%d%d",&l,&r);

ranges[i]={l,r};

}

sort(ranges,ranges+n);

int res=0,st=-2e9;

for(int i=0;i<n;i++)

{

if(st<ranges[i].l)

{

res++;

st=ranges[i].r;

}

}

cout<< res <<endl;

return 0;

}

算法2

按照左端点排序

C++ 代码

#include<iostream>

#include<algorithm>

using namespace std;

const int N=1e5+10;

struct Range

{

int l,r;

bool operator <(const Range &W)const

{

return l<W.l;

}

}range[N];

int main()

{

int n;

cin>>n;

for(int i=0;i<n;i++)

{

int l,r;

scanf("%d%d",&l,&r);

range[i]={l,r};

}

sort(range,range+n);

int res=0,st=-2e9;

for(int i=0;i<n;i++)

{

if(st<range[i].l)

{

res++;

st=range[i].r;

}

else

{

st=min(st,range[i].r);

}

}

printf("%d\n",res);

return 0;

}

作者：Accepting

链接：https://www.acwing.com/solution/content/10615/

来源：AcWing

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

Pair写法见附件