**算法训练 校门外的树**

时间限制：1.0s   内存限制：256.0MB

问题描述

　　某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数 轴上的每个整数点，即0，1，2，……，L，都种有一棵树。  
　　由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已 知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树 都移走后，马路上还有多少棵树。

输入格式

　　输入文件的第一行有两个整数L（1 <= L <= 10000）和 M（1 <= M <= 100），L代表马路的长度，M代表区域的数目，L和M之间用一个空格隔开。接下来的M行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点 和终止点的坐标。

输出格式

　　输出文件包括一行，这一行只包含一个整数，表示马路上剩余的树的数目。

样例输入

500 3  
150 300  
100 200  
470 471

样例输出

298

数据规模和约定

　　对于20%的数据，区域之间没有重合的部分；  
　　对于其它的数据，区域之间有重合的情况。

锦囊1

排序，区间运算。

锦囊2

先将所有的区间排序，然后从左到右处理所有的区间，可以依次得到合并后的区间。用总长度减去合并后的区间长度即可。

本题的C++参考代码如下：

#include<stdio.h>

#include<algorithm>

using namespace std;

struct Area

{

int l, r;

};

Area a[101];

bool cmp(Area a, Area b)

{

return a.l < b.l;

}

int main()

{

int L, m, i;

scanf("%d%d", &L, &m);

for(i = 0; i < m; i++)

scanf("%d%d", &a[i].l, &a[i].r);

sort(a, a+m, cmp);

int tot = 0, crt = -1;

for(i = 0; i < m; i++)

{

if(a[i].l > crt)

{

crt = a[i].l;

tot++;

}

if(crt < a[i].r)

{

tot += a[i].r - crt;

crt = a[i].r;

}

}

printf("%d\n", L-tot+1);

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

typedef struct

{

int start;

int end;

int flag;

}extent;

int main()

{

int L,M,i,j;

extent e[101];

scanf("%d%d",&L,&M);

for(i=1;i<=M;i++)

{

scanf("%d%d",&(e[i].start),&(e[i].end));

e[i].flag=1;

for(j=1;j<i;j++)

{

if(!(e[i].end<e[j].start||e[i].start>e[j].end)&&e[j].flag)

{

e[j].flag=0;

if(e[i].start>e[j].start)

e[i].start=e[j].start;

if(e[i].end<e[j].end)

e[i].end=e[j].end;

}

}//调整区间

}

for(i=1;i<=M;i++)

if(e[i].flag)

L=L-(e[i].end-e[i].start+1);

printf("%d",L+1);

return 0;

}

本题的Java参考代码如下：

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int m = input.nextInt();

int n = input.nextInt();

int[] a = new int[2 \* n];

int d = 0, e, max;

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

a[i] = input.nextInt();

a[i + n] = input.nextInt();

}

max = a[0];

for (int j = 0; j < 2 \* n; j++) {

if (max < a[j]) {

max = a[j];

}

}

int[] c = new int[max\*10];

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

for (int j = a[i]; j <=a[i + n]; j++) {

c[j] = 1;

}

}

for (int i = 0; i < max; i++) {

if (c[i] == 1) {

d = d + 1;

}

}

e = m - d;

System.out.println(e);

}

}