题目描述：

选择那些大晴天的日子,行走在孤单的海岸线,静静地种花给自己看~

我们假设把海岸线分为n块,每块的分别标记为1...n,每块都可以种花,每次种花可以选择某个[left,right]的闭区间,每块种上一朵花.经过m次种花操作后,根据输入的区间,求该区间内花的总数.

输入描述：

多组输入

对每组输入，第一行有两个整数n m,分别代表总块数和种花的次数.(1 <= n, m <= 100)

接下来的m行, 每行两个整数 L,R 代表[L,R]区间内每块种上一朵花.(1 <= L <= R <= n)

最后一行,输入两个整数 a,b 代表最后要查询的花的总数的区间.(1 <= a <= b <= n)

输出描述：

对每组测试数据，输出区间[a,b]内花的总数

样例输入：

对每组输入，第一行有两个整数n m,分别代表总块数和种花的次数.(1 <= n, m <= 100000)

接下来的m行, 每行两个整数 L,R 代表[L,R]区间内每块种上一朵花.(1 <= L <= R <= n)

最后一行,输入两个整数 a,b 代表最后要查询的花的总数的区间.(1 <= a <= b <= n)

输出描述：

对每组输入，输出区间[a,b]内花的总数

样例输入：

5 2

1 5

1 2

2 3

样例输出：

3

刚开始我们很容易想到种一朵花，就在每块土地上加一；

每组种花把土地标记 （循环n组）

每一组怎么种呢；（循环当下这一组开头到结尾）

然后最后把你要的土地上的话都加起来就好了；

美滋滋通过1350；

#include <iostream>

#include <cstring>

int a[1000005];

using namespace std;

int main()

{

int m, n, num1, num2, testnum1, testnum2;

while (scanf("%d%d",&m,&n) != EOF) {

int sum = 0;

memset(a,0,sizeof(a));

for (int i = 1; i <= n; i++) {

cin >> num1 >> num2;

for (int j = num1; j <= num2; j++) {

a[j]++;

}

}

cin >> testnum1 >> testnum2;

for (int i = testnum1; i <= testnum2; i++)

sum += a[i];

cout << sum << endl;

}

return 0;

}

but，提交到1351（数据加强版）就超时；

回顾一下

for (int i = 1; i <= n; i++) {

cin >> num1 >> num2;

for (int j = num1; j <= num2; j++) {

a[j]++;

}

}

1. 此时算法复杂度是n2；

优化必须降低复杂度；只有把嵌套的for修改；

2. 这里采用每种一组话计数一次（取代之前的每朵++一次，复杂度得到优化） ，

这就需要提前知道最后要查询的区域，

这里采用，把数据先读入存储，得到查询区域后重新遍历 （多储存一次复杂度n+n->n不影响） ；

并且提前知道查询区域，亦可略过不在范围内的数据，也是优化；

3. 这就要根据查询区间和种花区间一步操作得到种花数目（即当组种的数目，以便累加）

即去两个闭区间交集的长度 if((a[i][0] <= b && a[i][1] >= c)

count += min(c,a[i][1]) - max(a[i][0],b) +1;

总结: 每种一组话计数一次，取代之前的每朵++一次；（n2->n);

#include <cstring>

#include <iostream>

using namespace std;

int a[100005][2];

int main() {

int n, m;

while(scanf("%d%d",&n,&m) != EOF ) {

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

scanf("%d%d",&a[i][0],&a[i][1]);

}

int b, c, count = 0;

scanf("%d%d",&b,&c);

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

if((a[i][0] <= c && a[i][1] >= b) )

count += min(c,a[i][1]) - max(a[i][0],b) +1;

}

cout << count << endl;

}

}

第三题

第三次选择那些大晴天的日子,第三次行走在孤单的海岸线,第三次静静地种更多的花给自己看~

我们假设把海岸线分为n块,每块的分别标记为1...n,每块都可以种花,

每次种花可以选择某个[left,right]的闭区间,每块种上一朵花.经过m次种花操作后,

输入t次区间, 根据输入的区间,求该区间内花的总数.

注意这一次,我们要看更多次的花儿，所以在第一行要输入看花的次数t

输入描述：

多组输入

对每组输入，第一行有三个整数n m t,

分别代表总块数和种花的次数以及我们希望查询区间的次数.(1 <= n, m, t<= 100000)

接下来的m行, 每行两个整数 L,R 代表[L,R]区间内每块种上一朵花.(1 <= L <= R <= n)

接下来的t行, 每行输入两个整数 a,b 代表最后要查询的花的总数的区间.(1 <= a <= b <= n)

输出描述：

每组输入中, 对每次查询, 输出区间[a,b]内花的总数

样例输入：

5 2 2

1 5

1 2

2 3

3 4

样例输出：

3

2

样例说明

第一行的三个数5 2 2 分别代表一共有5块可以种花的地方

, 种花2次, 种完花后要查询2次

下面的两行 1 5 以及 1 2 表示在区间[1,5],[1,2]分别种一次花,不难算出,

种完花后每个位置花的总数分别为2 2 1 1 1,

最后两行2 3 以及3 4 表示我们要分别求出[2,3],[3,4]区间内花的总数,

所以输出的结果分别为3 2

把题目2 多输入t后代码用到第三题上来；很不幸又超时了；

还是要优化算法时间复杂度；

//本题思路接受徐浩闻指导，使用差分思想

本次优化最大亮点在于种花的存储方式：

in >> n1 >> n2;

map[n1]++;

map[n2+1]--;

12345 6 78

00000 0 00

10000 -1 00

12 3 45 6 78

00 0 00 0 00

11 -1 00 -1 00

意思是从n1（包括n1）开始到土地结束，都被我种了一朵

但是，明明不是种完接下来的所有土地；所以从n+1（包括n+1）开始到土地结束我种上-1朵，这样才平衡了；

那没操作的其他每一块map[i]，如 4,5意思是i（包括i）开始到土地结束，都被我种了0朵

此时怎样求一块土地上a[t]被种了多少花呢， 就是 t（包括t）前面所有种花次数相加

用map2[]存每块土地上的花数量

比如map2[1] = 1;map2[2] = 1 +1;map2[3] = 1 + 1 + -1;map2[4] = 1 + 1 + -1 + 0;

如果想求某些区间的话；求和就行，但是一个个累加时间复杂度又高；

这里我们采用mapsum3[i]数组存[1,i]上的总花数，;

所以区间[a,b]不用一个个叠加，只需要mapsum[b] - mapsum[a-1];

#include <iostream>

#include <cstring>

int map[100005];

int map2[100005];

int summap3[100005];

using namespace std;

int main()

{

int n, m, t, n1, n2;

while (scanf("%d%d%d",&n,&m,&t) != EOF) {

memset(map,0,sizeof(map));

memset(map2,0,sizeof(map2));

memset(summap3,0,sizeof(summap3));

for (int i = 1; i <= m; i++) {

cin >> n1 >> n2;

map[n1]++;

map[n2+1]--;

}

int countmap = 0, countmap2 = 0;

for(int j = 1; j <= 1000003; j++) {

countmap += map[j];

map2[j] = countmap;

countmap2 += map2[j];

summap3[j] = countmap2;

}

for(int i = 1; i <= t; i++) {

cin >> n1 >> n2;

cout << summap3[n2] - summap3[n1-1] << endl;

}

}

return 0;

}