**绍兴一中 NOIP 水题赛小题解**

**——by CYM**

|  |  |  |  |
| --- | --- | --- | --- |
| **中文题目名称** | **间谍** | **距离** | **染色** |
| **子目录名** | **spy** | **dist** | **paint** |
| **可执行文件名** | **spy** | **dist** | **paint** |
| **输入文件名** | **spy.in** | **dist.in** | **paint.in** |
| **输出文件名** | **spy.out** | **dist.out** | **paint.out** |
| **时间限制** | **2s** | **2s** | **2s** |
| **内存限制** | **256M** | **256M** | **256M** |
| **测试点数目** | **14** | **10** | **10** |
| **每个测试点分值** | **7~8** | **10** | **10** |
| **结果比较方式** | **全文比较（忽略行末空格及文末回车）** | | |
| **题目类型** | **传统** | **传统** | **SJ** |

T1(间谍，spy)：

第一道题无论从题面上还是从数据范围上看都说明了这是道水题。（题面里有出题人的直面题意也是醉了）。

本题n≤500，n3算法也能轻易水过，有floyd。太看得起这道题的算法有tarjan（…）。由于以上算法要么没有必要，要么过于简单，我就不描述了。重点讲解正常算法：DFS。

在输入的时候键图，标记一下入度为0的点，也就是所谓的上级。

一遍for以循环到的上级节点开始dfs，每个点开一个book数组记录可以从几个上级到达，然后在dfs里给每个可以遍历到的点book++，因为上级不会被举荐，所以不用考虑会不会重复。但是，数据里会有出现环的情况，在样例里就体现了8由7举荐，7由8举荐的情况（LaJi公司吃枣药丸），所以每次dfs，每个点最多只能被搜到一次。一个无关紧要的剪枝（没有好像也能过），就是搜索到一个点的时候它已经可以由两个上层达到的话，就不用由这个点再dfs下去了。

T2(距离，dist)：

（一开始想到了一个自认为完美的算法，结果是暴力……）

这道题是考数据结构的，不过因为数据的原因可以分块……（~~反正我不讲~~），你可以用三种方式来写。

第一种：树状数组。

这里要维护两个树状数组，具体维护什么之后会讲到。读入存到a数组里，本来是需要排序的，不过数据范围比较小可以直接用权值当下标来存在树状数组中（相当于桶排的思想）。一个树状数组是存权值的为b数组，另一个就是在对应位置++的（初始为0）c数组，一遍for循环中参数为i，存答案的变量为ans初始为0，求c数组[1,a[i]]区间和为num与b数组[1,a[i]]区间和为numLeft，ans+=num\*a[i]-numLeft。这是在a[i]节点左边节点到a[i]节点的距离和。右边也同理，ans+=numRight-（i-num+1）\*a[i]。输出ans后重置ans为0，再将a[i]加入两数组。

第二种：线段树。

这里只用维护一个线段树，但线段树中的参数（除去l与r）有三个，num，numLeft，numRight。num存这段区间内节点的数量，numLeft存这段区间内所有节点到区间左端的距离和，numRight同numLeft，不过是到区间右端点的距离和。之后就可以计算了，不需要Lazy，每次直接更新就行。我贴一下query函数

Void query(int p,int x){//p为当前线段树节点，x为a[i]

int l=tree[p].l,r=tree[p].r;

int mid=(l+r)>>1;

if(l==r)

return 0;

if(mid>=x)

return query(p\*2,x)+tree[p\*2+1].numLeft+tree[p\*2].num\*(mid-x);

else

return query(p\*2+1,x)+tree[p\*2].numRight+tree[p\*2+1].num\*(x-mid+1);

}

其实看一下代码也就看懂了。更新的地方就自己写吧。

我就不给了。

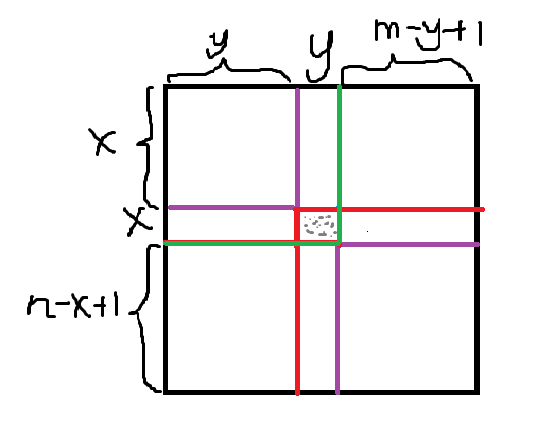
第三种：线段树

其实第三种也是线段树，不过实现方式与上面不一样。

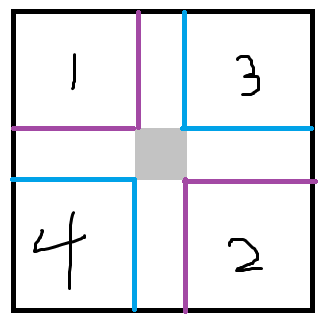
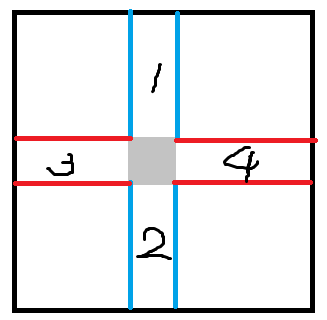
其实也就是把第一种的树状数组换成了线段树。。。（讲了跟没讲一样）。

T3（染色，paint）

最后一道题其实是数学题，主要思路是将每个点的被染色概率求出加在一起就是答案。概率的计算方式比较玄学，就是这个点被染色的情况数除以总情况数，由于直接求被染色的情况数比较难求（或者不可求），我们可以转换成1-（总的没有被染色的概率）。根据乘法原理，由于有k次操作，而每次操作时可以重复的，于是乎，（总的没有被染色的概率）=（一次没有被染色的概率）**k** 。然后（一次没有被染色的概率）=1-（一次被染色的概率）。于是就可求了，最后会变成一个简单的计算问题。

因为要求每一个点的概率，于是直接枚举每个点的坐标（x,y）。接下来我借图分析。我这里画了张丑图，中间有灰色小点的是当前枚举到的点（x,y）。其中由不同的线条画出了不同的空间，中间的量我也表得很清楚了。由于每次选取两个点为对角来选取矩形染色，我们要让灰块落在选取出来的矩形内。

首先得分类讨论，每种情况我都会配张图解释。

1. 这里我分成了四个小区块（原来的图纯属好看>\_<）我将它们编号为1,2,3,4，我们引入对块这个观念，其中1与2是对块，3与4是对块，要想将灰块包含进去，两个点的位置必须是分别来自一对对块的两块。我们先分析1与2这对对块根据上面的结论，我们可以得到，其中一个点要在1内而另一个点要在2内。在块1内有（x\*y）（也就是块的大小）个点可以选择，在块2内有（（n-x+1）\*（m-y+1））个点可以选择。根据乘法原理总的方案数为（x\*y）\*（（n-x+1）\*（m-y+1））；因为两个点并不重合，所以位置交换也没有关系，最后这对对块中的总方案数为（x\*y）\*（（n-x+1）\*（m-y+1））\*2。同理3与4中就为（x\*（m-y+1））\*（y\*（n-x+1））\*2。
2. 第二种情况中我们也分了四个小区块。同样我们再次引入对块这个观念。1与2,3与4；分析方法与上面几乎一样。不过这里块大小不同罢了。分析1与2.其中块1的大小为（x\*1）。块2为（（n-x+1）\*1）；再交换乘2；总方案数为（（x\*1）\*（（n-x+1）\*1））\*2；同理3与4中总方案数为（（y\*1）\*（（m-y+1）\*1））\*2；
3. 第三种情况其实非常简单，因为选取的两个点可以重复，显然我们剩下缺的就是两个点都是（x,y）的情况。也就是1 。

然后用加法原理得到一次可以选取到这个点的方案数=（x\*y）\*（（n-x+1）\*（m-y+1））\*2+（x\*（m-y+1））\*（y\*（n-x+1））\*2+（（x\*1）\*（（n-x+1）\*1））\*2+（（y\*1）\*（（m-y+1）\*1））\*2+1

这有点长，但我们可以因式分解简化它：设a=（x\*（n-x+1））\*2；b=（y\*（m-y+1））\*2；

可以得到原式=a\*b+a+b+1=(a+1)\*（b+1）;

最后根据我之前的给的算法算就好了。K次幂的时候用快速幂会比较好，然后就是一定要用double存答案