20231107题解

1. 烽火报信

原题P3047

40pts：

直接以每个结点为起点进行DFS，统计距离为k以内的结点的兵力的和就可以了。时间复杂度O(n2)。

50pts：

对k=1的情况，直接枚举边(a,b)，将a处的和加上b处的兵力，同时b处的和加上a处的兵力即可，时间复杂度O(n)。

100pts：

树形DP。先指定任意一个结点为根。

令dp[i][k]为以i为根的子树中距离为k以内兵力结点的和。

在第一次dfs的时候，显然有dp[i][k]=sum{dp[j][k-1]}，j是i的儿子。

在第二次dfs的时候，dp[i][k]就变成了到i距离为k以内结点兵力的和。

第二次dfs有两个思路：

1. 容斥。

第二次的时候直接用父亲去更新儿子，则可以有f[i][k]=f[j][k-1]，j是i的父亲。

但是这样一来就有重复计算了，因为i所在子树的那些更新到j的值被重新计算了，因此还需要减去f[i][k-2]，因为这些是更新上去f[j][k-1]的部分，而这一部分到i的距离不是k。

时间复杂度O(n)。

1. 换根。

将i换根成j的时候，需要先将i处j造成的贡献给减掉，再把j处i造成的贡献加上，这样就换根成功了，回溯的时候直接做逆处理就可以。

时间复杂度O(n)。

1. 出行计划

原题P2349，数据加强

40pts：

枚举最坏边，根据最坏边判断能不能走，然后Dij跑最短路，时间复杂度O(m2logn)。

60pts：

注意到最大边权很小，我们改为枚举边权，复杂度O(mWlogn)。原题正解有这个做法但是被我的数据卡掉了。和40pts方法兼容的话可以判断最大边权是否满足条件就好。

100pts：

需要对最短路进行一个较好的改进。

可以开一个dist数组记录最短路，开一个pre数组记录该最短路的最长边。在更新的时候比较的值是dist+pre的值。当然如果当前边的权值比pre值还大的话，就要用当前的边来代替pre，所以比较的时候应该写成：dist[i]+pre[i]>dist[j]+max(pre[j],val)+val，以此来用j点更新到i点的最短路。

注意这里的Dij如果只用dist的值来维护优先队列，就不要用vis数组来标记一个点是否访问过，因为可能在加上最长边之后的值有变化，需要重新更新。当然如果你在更新优先队列的时候将dist和pre两个值都算进去的话也可以进行标记，可能会更好。

复杂度O(mlogn)。

忘记卡SPFA了，回头加个subtask。

1. 数组累加

原题P4828

30pts：

直接按照题目要求模拟即可，时间复杂度O(nxQ)。

50pts：

可以将询问离线下来按照顺序排好再模拟，时间复杂度O(nx)。

100pts：

显然不可能直接模拟的。考虑每次单独求出对应元素而不管其他元素。

我们先写几项找规律，举例，对于第1项，前几次操作的结果如下：

a1 a2 a3 a4 ...

a1+a2 a2+a3 a3+a4 a4+a5 ...

a1+2a2+a3 a2+2a3+a4 a3+2a4+a5 a4+2a5+a6 ...

a1+3a2+3a3+a4 a2+3a3+3a+a5 a3+3a4+3a5+a6 a4+3a5+3a6+a7 ...

可以发现每一项的ai的系数都有杨辉三角的规律。而我们本质上可以把这个n项的序列看成一个无限长度的序列，以n个数为一循环的那种。例如上面的例子，如果n=5，那么a6=a1,a7=a2，以此类推。那么真正的做法就出来了。

先预处理处杨辉三角的前maxx行（建议多预处理几行），然后对每个询问，找到第x行，从第y个元素开始，乘上对应的系数之后加到答案中。由于每一行最多x个元素，故总时间复杂度O(xq)，可以通过本题。

1. 数子矩形

原题P7306

20pts：

直接枚举每个矩形，判断内部是否有雷，如果没有的话给矩形内每个格子的值加上1，最后求和即可。枚举左上右下顶点O(n4)，判断有无雷O(n2)，复杂度O(n6)。

50pts：

思路一：可以改进一下枚举次序并执行一些剪枝。例如说，枚举的时候先固定左上，然后枚举右下。如果右下是雷，那么这个点的右下就不再需要枚举了。对每个左上角，可以发现，需要判断的右下的点的序列必定满足x坐标递增时y坐标递减，所以可以考虑利用二分来判断一个点能否作为右下角。时间复杂度O(n4logn)。雷数少的时候大概O(n4)，雷数多的时候常数比较小的话也可以通过。

思路二：注意到每个矩形都会给它覆盖的点增加1的贡献，所以本题实际上是求矩形的面积和。考虑每一个点作为左上角能产生多少面积的矩形，每次枚举一个点，枚举向下延伸多长，可以得到以这条边最远能向右平移多远，最后等差数列直接计算，时间复杂度O(n4)。对原题数据大概可以拿80分。

100pts：

单调栈。

考虑每条横向边作为下边能够产生多少个矩形。

先开一个数组u[i][j]表示(i,j)最多能向上走多少（悬线法有印象吗），用单调栈维护边计算一个点(i,j)能提供多少贡献。如果新加入的这一点小于栈顶数据则改变栈顶状态。

考虑一个宽为w的横边，它带来的贡献是sum{i\*(w-i+1)}，1≤i≤w，因为宽为i的矩形有w-i+1种选择方法。这个式子求值是w(w+1)(w+2)/6。再考虑这个矩形的高，设我们的单调栈中的高度区间为[x,y]，则高度的贡献为sum{i}，x≤i≤y，这个式子求值是(x+y)\*(y-x+1)/2。最后计算栈中剩余数据即可，当当前的点比栈中第二大的还小时，则只去掉上面的部分，这些贡献值将会被分层计算到。

时间复杂度O(n2)。