problems

WerKeyTom_FTD

June 15, 2018

前言

分享一点题目，可能很原很简单。

一个nm的棋盘，n和m均为奇数。每个格子上都写了字母。
现在用了$\frac{nm-1}{2}$个1×2的多米诺骨牌覆盖棋盘。只有左上角没有被覆盖。
你每次可以选择移动一个骨牌，希望求出最少移动次数，使得每个元音格子都曾露出过。

bzoj2990 Keyboard

考虑每个位置向其相邻能走到的位置连边。
每次移动有一维会变化2，如果我们的图有环，长度一定是偶数。
考虑实际意义，一个环一定由一个空位和若干个骨牌拼接而成，那么环长度一定是奇数。
因此这个图是无环的，也就是树。
那么建出树，答案是关键点形成虚树边长和的两倍减掉最深关键点到根距离。

XIII Open Cup named after E.V. Pankratiev. GP of Saratov A Box Game

一个石子游戏，每个石子堆有石子上限，第i个石子堆石子上限为c_i。
现在有一个人初始局面，接下来两人轮流操作。
每次可以将一个石子堆里的石子取走或在一个石子堆里放入一个石子。
不能到达已经到达过的重复局面。
你可以选择先手或后手来加入这个游戏，你希望你能赢。
$\prod_{i=1}^{n}(c_i+1) \le 50000$。

XIII Open Cup named after E.V. Pankratiev. GP of Saratov A Box Game

每一次操作都会改变石子总数的奇偶性，不如根据这个将状态间建成二分图。
然后我们考虑求个最大匹配。
如果总状态数是偶数，显然存在完美匹配，即假设第一堆石子上限是奇数，那么将第一堆石子有i个的局面和第一堆石子有i^1个的匹配即可。
此时先手必胜，每次沿匹配边走，后手无法返回，走到另一个状态，由于存在完美匹配，后手最后陷入绝望。
如果总状态数为奇数，说明每堆石子上限均为偶数，我们类似的考虑可以发现石子总数为奇数的那边是匹配满了的。
此时如果初始石子总数为奇数先手也是必胜的。
而如果是偶数后手是可以必胜的，因为二分图是联通的，而且其中一部是匹配满的，我们删除另一部的一个状态，显然存在增广路调整，使得新的图仍然能匹配满。

有一个卖肥宅快乐水的机子，一罐肥宅快乐水需要r元。机子里初始预备了d块硬币。
大佬吴董超现在手头上有a张票子和b块硬币，一张票子是10^6元，一块硬币就是1元。
每次超哥可以把若干张票子和若干块硬币放入机子里，假如价值等于z元，要求$z \geq r$。
然后机子会找给超哥$z - r$个硬币，如果硬币数量不够，机子既不会找钱，也不会出肥宅快乐水。
如果硬币数量够，机子就会找钱，并且会出一罐肥宅快乐水。你投进去的硬币也会被机子用来找钱。
问超哥最多能喝到多少灌肥宅快乐水？
所有数在10^9内。

策略是没有用的，我们都能买到同等数量的肥宅快乐水。
关键在于加速模拟的过程。
我们先考虑这样一个策略，每次都拿出$\lfloor \frac{r}{10^6} \rfloor$张票子，然后剩余的钱用硬币补，如果硬币不够，多加一张票子让它找钱。
假设票子是够用的，我们可以发现这个过程类似于$b- > b\%p- > b+q- > b\%p$。由于一张票子只有10^6元，这个过程循环节就是10^6。
当票子不够用时，显然有$a \leq 1000$，这时我们的策略变成每次拿点票子加上硬币去凑之类的，但不管如何这个程度已经可以暴力模拟了。

Yandex.Algorithm 2018, final round E Guess Me If You Can

这是一道交互题。
有一个长度为n的排列p，你并不知道它是什么。
你每次可以将某个位置的数加一，然后交互库会告诉你现在排列里数的种类是多少。
你要在50n次操作以内，找出原排列中n的位置。

我们来考虑做50次下列过程：
随机一个排列q，然后依次让p_{q_i}加一，如果种类数减少，可以排除这个位置是n。
对于一个x，只有其在q中位置在$x-1$和$x+1$前面时其才会被排除。
考虑正确率，$(1-(\frac{2}{3})^{50})^{1000}$，很有保证。

定义对正整数x的依次变换为：
有$\frac{1}{2}$的概率变成$x + lowbit(x)$,$\frac{1}{2}$的概率变成$x - lowbit(x)$。
定义$f(x)$表示正整数x变换为0的期望步数。
给定L, R,求$\sum_{x=L}^{R} f(x)$
$0 \leq L \leq R < 2^{31}$

考虑计算f的前缀和。
乍一看加加减减的，似乎一个数操作若干次会变回来，其实不然。
仔细分析性质，我们可以发现：
1，不管是+lowbit还是-lowbit，过程虽然是曲折的，数值可能会变大变小，但是后缀连续零是前进的，每次操作后严格递增的。
2，一个位置最多只会被进位一次。

因此我们可以考虑一个从低位往高位的数位dp，令$f_{i,j,k}$表示0到$i-1$位上都已经变成了0，当前这一位是（否）有来自$i-1$的进位，以及0到$i-1$组成的二进制数是（否）小于上界这些位置对应的二进制数情况下的期望，为了转移，我们还要类似地记录这种情况的概率。
转移很简单，直接考虑这一位填什么，简单讨论就好了。

现在还有一个问题就是我们可能在填完$log(X)$位之后，依然有进位。
注意到这个时候只有这一个最高位的1，简单的期望知识就可以得到它能在期望2步内被删除，我们统计这种情况出现的概率乘上2就能处理了。

经典问题

一个字符集为大小写字母长度为n的字符串。
在线的每次询问一个区间，其本质不同子序列有多少个?
$n,q \leq 10^6$。

区间本质不同子序列个数，强制在线。

$n, q \le 10^6, \sum \le 52$。

Solution

首先考虑如何求本质不同的子序列个数，有一种DP是记 $f(i,j)$ 表示当前考虑到 i 以 j 结尾的子序列个数，那么转移的时候每个数可以选或者不选，这样有两种转移，但 $j = s[i]$ 时就必须选，这样就完成了去重，那么把转移写成矩阵的话，大概长这样(最后一维用来记答案)：

$$A_0 = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}, \quad A_0^{-1} = \begin{pmatrix} 1 & -1 & -1 & \cdots & -1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}。$$

那么我们就是要求 $\begin{pmatrix} 1 \\ 1 \\ 1 \\ \cdots \\ 1 \end{pmatrix}^T A_{s_r} A_{s_{r-1}} \cdots A_{s_l} \begin{pmatrix} 0 \\ 0 \\ 0 \\ \cdots \\ 1 \end{pmatrix}$ 的值。

转化成求后缀积和逆矩阵的前缀积的形式，那么单次询问可以做到 $O(\sum^3)$ 的复杂度。

如果发现了只需要求矩阵的一行，单次询问可以优化到 $O(\sum^2)$。

注意到这个矩阵的性质非常优美，记 $J_i = A_{s_i} A_{s_{i-1}} \cdots A_1$，将 J_i 拆分成 $\sum +1$ 个行向量 $u_{i,j}$ 的形式，记 $S_i = \sum u_{i,j}$，考虑

$S_i = \sum u_{i,j}$ ，考虑

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{i,0} \\ u_{i,1} \\ u_{i,2} \\ \cdots \\ u_{i,\sum+1} \end{pmatrix} = \begin{pmatrix} S_i \\ u_{i,1} \\ u_{i,2} \\ \cdots \\ u_{i,\sum+1} \end{pmatrix} 。$$

注意到每个 u_i 都是之前的某个 S_i ，并且只改变了一行，那么就可以快速维护 J_i 了。

再考虑 $I_i = A_1^{-1}A_2^{-1}\cdots A_i^{-1}$ ，类似地，将 I_i 拆分成一些列向量 $v_{i,j}$ 的形式，这里发现

$$\begin{pmatrix} v_{i,0} \\ v_{i,1} \\ v_{i,2} \\ \cdots \\ v_{i,\sum+1} \end{pmatrix}^T \begin{pmatrix} 1 & -1 & -1 & \cdots & -1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} = \begin{pmatrix} v_{i,0} \\ v_{i,1}-v_{i,0} \\ v_{i,2}-v_{i,0} \\ \cdots \\ v_{i,\sum+1}-v_{i,0} \end{pmatrix}^T 。$$

如果类似 S_i 构造一个减法就可以降低复杂度了，具体地，记一个列向量 D_i 表示所有列向量需要减掉的值，即

$$\begin{pmatrix} v_{i,0}-D_i \\ v_{i,1}-D_i \\ v_{i,2}-D_i \\ \cdots \\ v_{i,\sum+1}-D_i \end{pmatrix}^T \begin{pmatrix} 1 & -1 & -1 & \cdots & -1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} = \begin{pmatrix} v_{i,0}-D_i \\ v_{i,1}-v_{i,0} \\ v_{i,2}-v_{i,0} \\ \cdots \\ v_{i,\sum+1}-v_{i,0} \end{pmatrix}^T 。$$

而新的 $D_{i+1} = v_{i,0}$ 就可以回归原来的形式了，这样只有 D_i 和 $v_{i,0}$ 改变，也可以类似快速维护。

经典问题

而新的 $D_{i+1}=v_{i,0}$ 就可以回归原来的形式了，这样只有 D_i 和 $v_{i,0}$ 改变，也可以类似快速维护。

考虑计算 $[i+1,j]$ 这个区间的答案，我们是要求

$$\begin{pmatrix}1\\1\\1\\\cdots\\1\end{pmatrix}^T\begin{pmatrix}u_{j,0}\\u_{j,1}\\u_{j,2}\\\cdots\\u_{j,\sum+1}\end{pmatrix}\begin{pmatrix}v_{i,0}-D_i\\v_{i,1}-D_i\\v_{i,2}-D_i\\\cdots\\v_{i,\sum+1}-D_i\end{pmatrix}^T\begin{pmatrix}0\\0\\0\\\cdots\\1\end{pmatrix}$$

利用结合律，前两个矩阵的积就是 S_j，而后两个矩阵的积是 $v_{i,\sum+1}-D_i$，不难发现 $v_{i,\sum+1}=\begin{pmatrix}0\\0\\0\\\cdots\\1\end{pmatrix}$

最后要求的就是 $S_j(\begin{pmatrix}0\\0\\0\\\cdots\\1\end{pmatrix}-D_i)$，这样就可以单次询问 $O(\sum)$ 了。

对一个包含若干互不相同的1到9之间的整数序列，我们可以用以下方法生成若干条提示：

1，随机选择序列中的一个位置作为起点。

2，随机选择一个方向（左/右）。

3，从起点开始沿着选定的方向走，遍历完这个方向的每个数字，将每个数字第一次出现的顺序记录下来。

现在给定n条提示，请找到长度最短的满足条件的整数序列。

$n \leq 10$。

CERC2016 I, Invisible Integers

不妨先考虑假如所有的提示都是从左向右的怎么做。
首先不难发现，提示y能比提示x晚出现，当且仅当y中所有数都在x中出现。
考虑从左往右匹配的过程，可以观察到提示之间可能存在类似“转化”的关系。
定义提示x在第i位之后可以转化成提示y，当且仅当：
1，到目前为止，提示x前i位数字都已经被匹配了。
2，提示y能比提示x晚出现。
3，提示x第i位后面的数字，在提示y中出现的相对顺序不变。
可以观察到，如果提示x在第i位之后可以转化成提示y，那么后面我们继续填数时就可以忽略掉提示x是否被满足，因为在这时提示x被满足的条件已经严格弱于提示y被满足的条件。

CERC2016 I, Invisible Integers

有了这个思路，我们可以得到一个顺序的dp算法，令$f_{s,x,i}$表示当前所有被转化过的提示集合为s,当前最后转化到提示编号为x，这个提示匹配到的位置为i的状态下，序列长度的最小值。每次我们枚举最后一维新加入的数字，显然这个数字要么是匹配的下一位，要么是当前提示前i位数字中的一个。

接下来考虑从右向左的提示，可以发现我们只需要在原来基础上加多两维，分别记录当前从右向左的提示最后转移到了提示y，这个提示匹配到的位置j(从右向左)。
转移依然是和只考虑一种方向时候的类似的。