NOIp2018 提高组
解题报告

NaVi_Awson

2019 年 8 月 17 日

目录

1 前言

~~终于赶在 NOIp2019 前把这六道题写完了...~~

听闻 NOIp 没了???

从此 NOIp2018 成绝唱了???

只好以此题解来缅怀我逝去的金钱多年的 OI 生涯...

考虑到 NOIp2018 网上很难找到靠谱的题解。比如很多神仙 day2t2 直接丢公式，day2t3 直接说动态 dp 裸题~~（确实挺裸的）~~，然后直接丢动态 dp 教学...

因此，本题解旨在服务于真 · NOIp 选手，不会出现花里胡哨的算法，尽可能做到通俗易懂。

代码见我的博客：https://www.cnblogs.com/NaVi-Awson/p/11366652.html。

2 Day1 T1 铺设道路

由贪心的思想，我们需要每次最大化下陷的效果。即每一次下陷，都要尽可能地带动边上的位置下陷。

那么我们可以从左至右扫一遍高度数组 d，若 $d_i > d_{i-1}$，那么我们需要的天数增加 $d_i - d_{i-1}$ 天。这是由于 i 这个位置的前 d_{i-1} 的高度可以跟随第 $i-1$ 个位置一起下陷，而剩下 $d_i - d_{i-1}$ 的高度需要独自下陷，当然也可能带动后边的位置下陷。

- 时间复杂度：$O(n)$；
- 空间复杂度：$O(1)$ 或 $O(n)$，取决于是否开数组存高度。

3 Day1 T2 货币系统

一个显然的结论就是最简的货币系统中的某一种面值一定不能被该系统中其他的面值表示。因为有这种面值存在时，你可以除去之，使系统更简。

进一步地，我们可以知道，最简的货币系统中一定只包含原系统中的面值。

可以用反证法证明，我们感性地理解一下（注意，感性 $\neq$ 不严谨）。其一，若出现的面值可以被原系统的面值表示，由前面第一条结论，并且该面值不可能代替原有的比它小的面值，显然不合理；其二，若出现的面值不能被原系统的面值表示，这就不符合题目含义了，更加不可能。

因此我们需要做的就是找到所有“冗余”（即能被其他面值表示）的面值，把它删掉即可。于是做一次完全背包，若一种面值能被除自己以外的方法表示，就把它剔除。

- 时间复杂度：$O(n \times \max\{a_i\})$；
- 空间复杂度：$O(\max\{a_i\})$。

4 Day1 T3 赛道修建

题目要求最大化最小值，显然直接二分答案就好了。

记二分的长度为 l，现在问题就转化成了，是否能在这棵树上找到 m 条以上长度 $\geq l$ 的路径。

具体可以树形 dp 来实现。树形 dp 基于一个贪心：只要某条路径长度 $\geq l$ 就把他截断，因为剩余部分加在这条路径中并不会使得答案更优。考虑怎么处理节点 u：

首先我们需要将子树信息合并。对于节点 u 的某一个儿子（设为 v）所在的子树，由于边 (u,v) 不能重复使用，那么我们设在**最优状态下**（最优状态的定义之后再说）连上来的最长链的长度为 f_v（f_v 中包含了 (u,v) 的长度）。遍历完所有子树后，我们可以从每一个子树中都得到一条链，接下来需要做的就是将能够两两合并使长度和 $\geq l$ 的链合并。对于某一条长度为 f_v 的链，由贪心的思想，需要在未合并的链中找到长度 $\geq l - f_v$ 且最小的一条出来。那么我们需要一个可支持删除，查找后继的数据结构出来。可以选择手写平衡树或直接用 STL 的 multiset。当然了事先需要将链集合中长度 $\geq l$ 的链统计答案并舍去。

最后匹配完全后，就是我们说的最优状态。若此时集合中还剩元素，我们就把最长的那条作为 f 值传给 u 的父亲，记得加上边 (u, fa_u) 的长度。若集合为空，那就直接把边 (u, fa_u) 长度作为 f 值传上去。

- 时间复杂度：$O\left(n \log n \log \frac{\sum l_i}{m}\right)$，并不满，极端情况是菊花图；

- 空间复杂度：$O(n)$。

5 Day2 T1 旅行

先考虑对于一棵树的情况怎么做。根节点一定是 1，这是肯定的。由于题目说了走到叶子节点才能往回走。那么显然对于当前节点 u，选一个未走过的编号最小的儿子走才会最优。找编号最小儿子，可以事先将儿子丢进一个 vector，排序即可。

而对于基环树，直接暴力考虑删掉环上的某条边，将其变成树，再以 1 为根贪心遍历即可。

- 时间复杂度：$O(n^2)$；
- 空间复杂度：$O(n)$。

6 Day2 T2 填数游戏

这题题解的画风稍微转变一下，请先将题目读懂。

引理 6.1 交换 n, m 的值，答案不会发生变化。

证明 6.1 交换后，直接将每个格子的 *01* 状态取反即可。

引理 6.2 对于 $x + y$ 为定值的所有点 (x, y)，构成一条斜线，自左向右是连续的一段 *1* 加上连续的一段 *0*，可以全部相同。

证明 6.2 由于“*DR* 字符串”向下走字典序更小。所有一条到 (x, y) 的路径向两个方向走的时候必须要有 $(x+1, y)$ 上的数字不小于 $(x, y+1)$ 即任意 (x, y) 不小于 $(x-1, y+1)$。

引理 6.3 若 $(x-1, y)$ 和 $(x, y-1)$ 上的数字相同，那么以 (x, y) 为左上角的极大的子矩阵中在每个斜行上的数字都相同。即对于 $a \geq x \cap b \geq y$ 的 (a, b) 有约束。

证明 6.3 这是因为当 $(x-1,y)$ 和 $(x,y-1)$ 上的数字相同，从 $(x-1,y-1)$ 经过这两个位置再到 (x,y) 的路径的 *01* 串相同。也就是说到 (x,y) 有至少两条 *01* 串相同的路径。那么接下来的每一步必须相同，否则会存在不满足限制的路径。

同时**引理 6.2** 与**引理 6.3** 是题目的充要条件，故之后的讨论只要满足这两点即可。由**引理 6.1** 我们默认 $n \le m$，接下来对题目分析。

6.1 $n = 1$ 的情况

这个时候填的每个数都不受约束，故答案为 2^m。

6.2 $n = 2$ 的情况

注意到，此时起点和终点可以随便填，方案数为 4，考虑剩下 $m-1$ 条斜线怎么填。

可知由于**引理 6.2**，同一对角线从左至右只能填 00,10,11 这三种情况。同时注意到：

若两个蓝色部分填数一样，**引理 6.3** 影响到的只会是红色的部分。而使红色部分中斜线填数一样，就等于没限制。

因此答案为 $4 \times 3^{m-1}$。

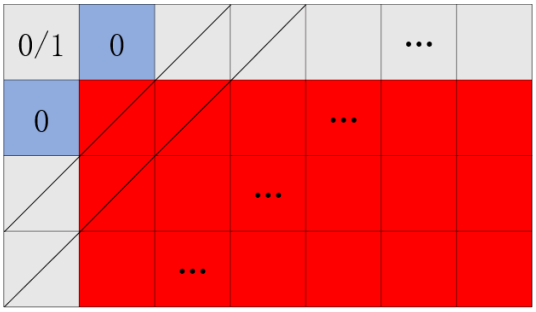
6.3 $n \ge 3$ 的情况

同样，起点和终点随便选。

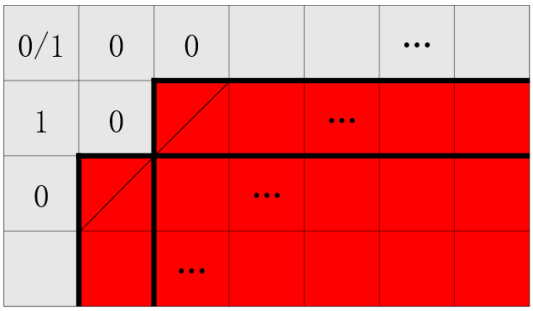
若 $(0,1)$ 和 $(1,0)$ 相同

那么如图，会发现画了黑斜线的同一斜线上的部分等价于只有 3 个格子填数，其余部分的斜线等价于只有 2 个格子或者 1 个格子，三个格子填数有 000,100,110,111 共 4 种情

况，可以直接计算。



若 $(0,2),(1,1)$ 和 $(2,0)$ 相同



如图，此时这三个格子填相同的数时影响的分别是两个矩阵。可以发现，尽管画斜线的两个格子均为红色且在一条斜线上，但由于它们分别属于两个不同的矩形，故不会相互约束，因此还要额外判断这一条斜线填什么，之后的就与上一种情况相同了。

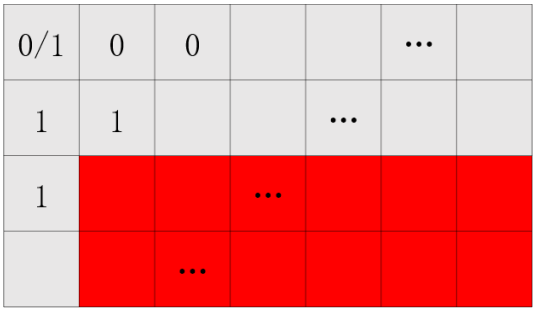
这里的话，应该要发现一条规律：长度为 x 的斜线，满足要求的填数方案为 $x+1$ 种。证明很显然。

若 $(1,1)$ 与 $(2,0)$ 或 $(0,2)$ 相同

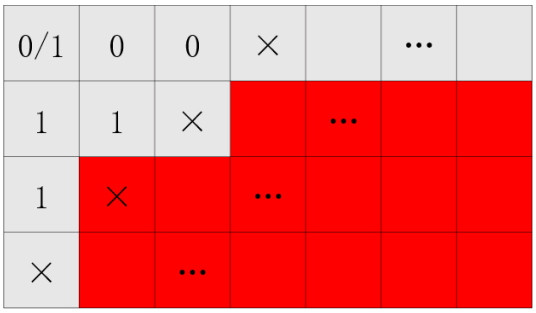
以 $(1,1)$ 与 $(2,0)$ 为例，另一种情况类似。

如图，此时影响的只有红色的区域。并且，这样填了之后，以后的每条斜线中本质上不同的格子都降到了 4 个及以下。

接着考虑后面一条斜线怎么填。



如果填 $0000, 1100, 1000, 1111$，可以发现得到的局面都是这样的：



此时后面一行就和第二种情况相似了，填完后面这一行后，就能够变成第一种情况。

而如果填 1110，此时之后的状态会维持现状。但这个时候这一行是唯一确定的，故我们只需要去考虑再下一行怎么填就行了。

因此对于 $(1,1)$ 与 $(2,0)$ 或 $(0,2)$ 相同这种情况，我们只需要去考虑当前这一行怎么填。若是 $0000, 1100, 1000, 1111$ 这四个中的一种，只需额外考虑下一行怎么填，之后的归为第一种大情况；若是 1110，则去枚举下一行怎么填，如此往复。而且到最后斜线一定会都位于红色区域内，是有终点可循的。

因此最后只需要考虑如何快速求出第一种大情况的答案。其实只需按斜线的顺序（总共 $n+m-1$ 条斜线），用每条线能填的方案数做一遍后缀积即可。

- 时间复杂度：$O(n+m)$；
- 空间复杂度：$O(n+m)$。

这个算法还可以继续优化，但既然是纯 NOIp 题解，就不继续讲了。

7 Day2 T3 保卫王国

若不存在钦定的点，那么这道题就是很经典的有关独立集的问题。

我们记 $dp_{u,1/0}$ 表示以 1 为根的树中，节点 u 选/不选时，以 u 为根的子树中满足条件的最小花费。

那么可以得到转移方程：$\begin{cases} dp_{u,0} = \sum\limits_{\text{v is a child of u's}} dp_{v,1}, \\ dp_{u,1} = \sum\limits_{\text{v is a child of u's}} \min\{dp_{v,0}, dp_{v,1}\}. \end{cases}$

而对于钦定为选或不选的点，我们只需将它的 dp 值改为 ∞ 即可。而若每组询问都重新来一次树 dp 显然复杂度承受不了。不过注意到，每次修改 a, b 两点状态时，发生影响的点只存在 (a, b) 这条链上，以及 $\text{lca}(a, b)$ 至根节点 1 的路径。

我们可以开一个倍增数组 $f_{u,i,1/0,1/0}$ 表示点 u 选/不选，u 的 2^i 祖先选/不选，这条链上所有点（不包含 u）以及从属它们的子树中选点符合条件时的最优解。

这个 f 数组其实很好预处理，倍增时只需考虑中转的点的状态即可。

考虑怎么统计答案，先考虑一般情况（a, b 从属 $\text{lca}(a, b)$ 的不同子树）。

首先先要将深度大的点（设为 a）跳到与 b 同一高度，边跳边更新 dp 数组，实际上不需要直接在原数组上修改，直接开两个变量，表示跳到当前点时，当前点选/不选得到的子树中最优值。之后再两者一起跳到 $\text{lca}(a, b)$。

不过处理 $\text{lca}(a, b)$ 时是需要合并两个子树的信息的，采取的策略是，利用原 $\text{lca}(a, b)$ 处的 dp 值减去原来 a, b 所在子树的 dp 值，再加上这一次跳的值来更新。

最后再让 $\text{lca}(a, b)$ 跳到根节点即可。

对于特殊情况：当 a, b 在一条链上时，倍增 a 之后已经在 $\text{lca}(a, b)$ 上了，此时直接倍增 $\text{lca}(a, b)$ 即可；当 $\text{lca}(a, b)$ 为根节点时，显然不需要倍增 $\text{lca}(a, b)$ ，此时直接算出总答案。

- 时间复杂度：$O(n \log n)$；
- 空间复杂度：$O(n \log n)$。