DP 选讲

larryzhong

前言

- 今天我们讨论一些 DP 相关的趣题

- 题目不难，每道题目思考 5-10 min
- 欢迎大家积极发言

题意

Count Sequences

You are given 2 integers N and M.

For each x such that $0 \le x < M$, count the number of non-negative integer sequences $A_1, A_2, \ldots, A_M$ such that:

- $\sum_{i=1}^{M} A_i = N$
- $\prod_{i=1}^{M} i^{A_i} = x \pmod{M}$

Since the answer can be large, output the answer modulo $10^9 + 7$.

- $1 \le T \le 40$
- $1 \le N \le 10^9$
- $2 \le M \le 40$

CodeChef COUNTSEQ2

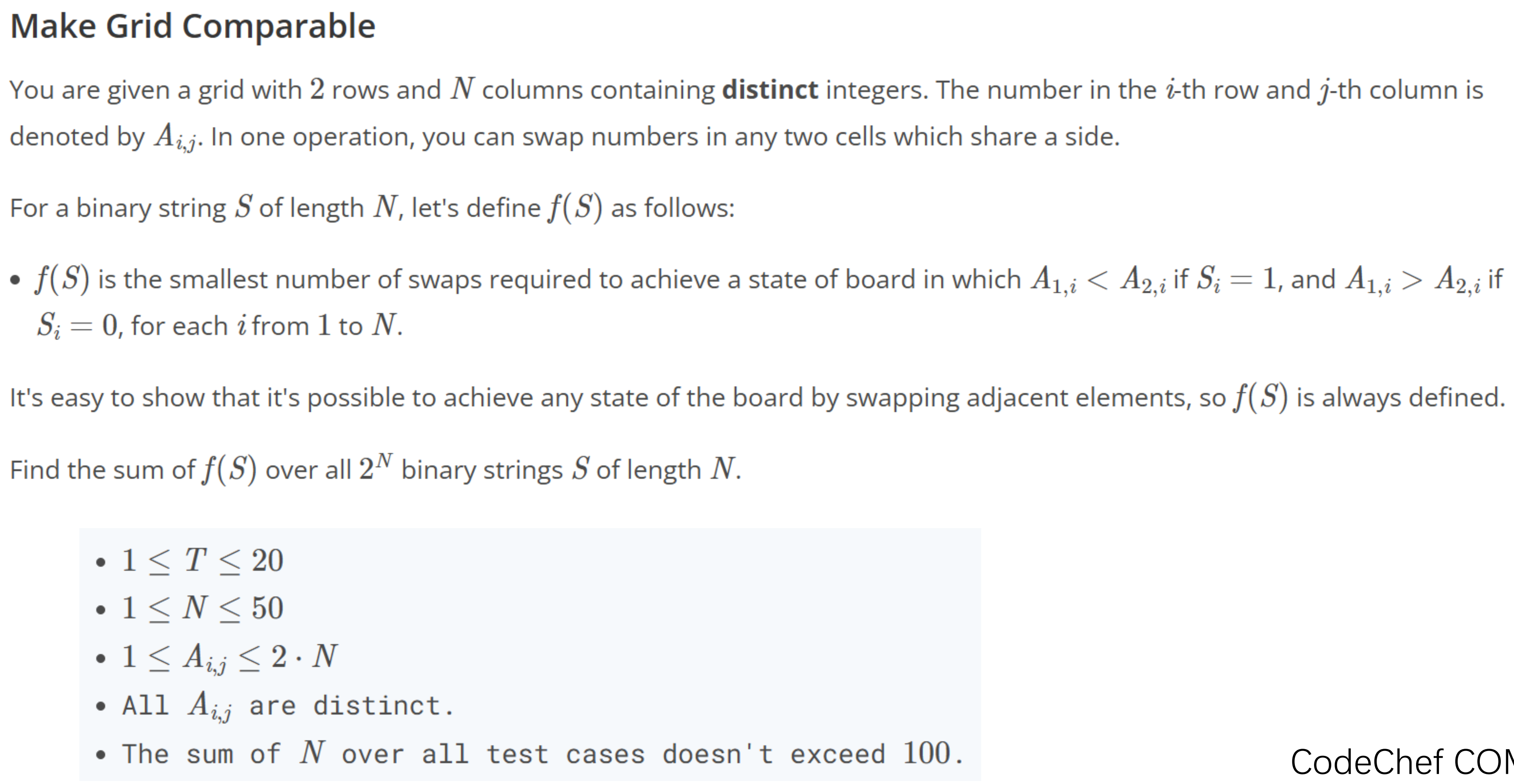
题解

注意到 $x^0 \bmod m, x^1 \bmod m, \cdots$ 构成 ρ 型，且环长是 $\varphi(m)$ 的约数。也就是说 a_i 总是可以约束在 $O(\varphi(m))$ 级别，最后再乘以组合数计算方案数。为了计算方案数，我们除了总和与乘积还关心有多少个 a_i 在环上。

DP 状态是 $f(\text{sum}, \text{product}, \text{number of elements in cycles})$，对每个 x 依次背包即可。

时间复杂度为 $O(m^5)$ 或 $O(m^6)$，取决于是否使用前缀和优化。

题意



Make Grid Comparable

You are given a grid with 2 rows and N columns containing **distinct** integers. The number in the i-th row and j-th column is denoted by $A_{i,j}$. In one operation, you can swap numbers in any two cells which share a side.

For a binary string S of length N, let's define $f(S)$ as follows:

- $f(S)$ is the smallest number of swaps required to achieve a state of board in which $A_{1,i} < A_{2,i}$ if $S_i = 1$, and $A_{1,i} > A_{2,i}$ if $S_i = 0$, for each i from 1 to N.

It's easy to show that it's possible to achieve any state of the board by swapping adjacent elements, so $f(S)$ is always defined.

Find the sum of $f(S)$ over all 2^N binary strings S of length N.

- $1 \le T \le 20$
- $1 \le N \le 50$
- $1 \le A_{i,j} \le 2 \cdot N$
- All $A_{i,j}$ are distinct.
- The sum of N over all test cases doesn't exceed 100.

CodeChef COMPRBLEGRID

题解

- 假如第 i 列和第 $i+1$ 列交换过，连边 $(i,i+1)$。考虑这张 n 个点的图的连通块 $[l,r]$。因为直接暴力就是 $r-l+1$ 次，所以需要判定能否用 $r-l$ 次 $(i,i+1)$ 类型的交换达成目标。
- 这可以用 DP 来判定。状态：
  - l,r
  - 当前第 l 列的状态
  - 当前第 r 列的状态
- 转移很简单，枚举第一次交换的 $(i,i+1)$ 并递归子问题即可。
- 注意到 DP 状态是 $O(n^4)$ 的，所以总复杂度为 $O\left(n^5\right)$。

题意

有 n 个硬币。已知恰好一个硬币是假币，其余都是真币。所有真币的重量都相同，且与假币不同。

有一个天平秤来称量硬币。称量时，将一些硬币放在左边的秤上，另一些硬币放在右边的秤上。称量有三种可能的结果：

- “<”：左侧较轻。
- “=”：两侧重量相同。
- “>”：左侧较重。

你的任务是找出哪枚硬币是假币，以及**它比真币更轻还是更重**。现在已经用天平秤做了 m 次称量。给你这些结果，求最坏情况下，还需要最少额外多少次称量。

保证 $3 \le n \le 62, 0 \le m \le 50$ 且输入合法。

TopCoder 某个题，题号忘了；
当时我放在联考里，如果做过的同学太多就不讲

题解

某次称量如果返回“=”，那么涉及到的硬币都是真币；否则，不妨假设返回“<”，这给出了如下信息：

- 未涉及到的硬币都是真币。
- 要么左侧的某个硬币是比真币轻的假币，
- 要么右侧的某个硬币是比真币重的假币。

如果 m 次称量全部返回“=”，我们只用关心**有多少个硬币可能是假币**；否则，如果有任意一次返回不等关系，我们只用关心**有多少个硬币可能是比真币轻的假币**和**有多少个硬币可能是比真币重的假币**。两种情况的状态数分别是 $O(n)$ 和 $O(n^2)$。

一旦我们处于第二种情况，就会一直处于第二种情况。对于第一种情况，如果进行的称量返回不等关系，就会变成第二种情况。

考虑 DP 求解。如何描述一次称量？以第二种情况为例，显然两侧不可能都放真币（否则各移除一个），因此我们只用关心左右两侧的硬币数量、左侧的“轻”币数量、右侧的“轻”币数量、右侧的真币数量。枚举量为 $O(n^4)$。

转移时枚举一次称量，并选择最坏情况下答案最小的称量方式即可。

时间复杂度为 $O(n^6)$，常数很小。

题意

- 给定 n 个点的树，带点权
- 给定区间 $[L,R]$。$\forall 0 \le i \le k$：判断能否断掉 i 条边使得，所有 $i+1$ 个连通块内部的点权和都在 $[L,R]$ 中
- 保证 $n \le 10^3, k \le 50$，权值 10^{18}

GYM 103627K

题解

- 考虑设 $f(u,i,x)$ 表示 u 子树内选取了 i 个连通块，目前还未封闭的连通块权值能否为 x。当一个点合并完所有儿子后，再尝试封闭当前的连通块（即 $x \in [L,R]$），转移到 $f(u,i+1,0)$。
- 这看上去不太能做。

题解

- 不妨将 $f(u,i)$ 看做一个集合，下面的引理指出有用状态数并非指数级。

- 引理 1： $\max\{f(u,i)\} - \min\{f(u,i)\} \le k(R-L)$
- 证明：已经确定的连通块的权值和在 $[kL,kR]$ 间。

- 引理 2：对于集合 S，如果 $a,b,c \in S$ 且 $a < b < c, c-a \le R-L$，那么舍弃 b 不影响最终的判定结果。
- 证明：我们只关心 S 内是否有长度为 $R-L$ 的特定区间中的点。

- 单次合并是 $O(k^2)$ 的，共 $O(nk)$ 次合并，所以总复杂度是 $O(nk^3)$。

题意

You are given $2^k - 1$ numbers $c_1, c_2, \ldots, c_{2^k-1}$ and k numbers $a_0, a_1, \ldots a_{k-1}$.

You want to find nonnegative integers $x_1, x_2, \ldots, x_{2^k-1}$ such that for all j $(0 \le j < k)$

$$\sum_{i=1}^{2^k-1} \left(\lfloor i/2^j \rfloor \bmod 2\right) x_i = a_j$$

holds and $\sum_{i=1}^{2^k-1} x_i c_i$ is maximized.

In the first line, T $(1 \le T \le 100)$ — the number of test cases.

For each test case:

- In the first line, k $(2 \le k \le 4)$.
- In the second line, $c_1, c_2, \ldots, c_{2^k-1}$ $(0 \le c_i \le 10^8)$.
- In the third line, $a_0, \ldots, a_{k-1}$ $(1 \le a_i \le 10^9)$.

QOJ 6194;
类似的题：CF 1290F

题解

This problem is solvable by solving the linear programming, then trying to enumerate how to round the solution, or just copying an ILP solver, or some heuristics. But they are not intended.

The intended solution is digital dp. A similar problem here.

Basically, the problem is a four-dimension knapsack problem. But if we solve this four-dimension knapsack problem directly, the time complexity will be at least $O(W^4)$, which is not acceptable.

So we can consider fixing the value for each x_i from the highest bit to the lowest bit in binary, and record the remaining volume for these four dimensions. We can see if the volume for one dimension is greater than 15 (the actual volume is $15 \times 2^b + a_i \bmod 2^b$) before we consider the b-th bit, it can not be filled even if we set $x_i = 2^b - 1$ for all remaining bits. So we can find a solution that runs in $O(\log W \times 16^4)$. But it may be hard to fit in the time limit for its large time constant.

We can find the upper bound 15 is not tight. For example, if the numbers of subsets $1, 1234, 12, 134$ are greater than 0, it's always to decrease and increase the pairs $(1, 1234), (12, 134)$ and make the answer not smaller. So there are at most 5 sets used in the optimal solution when we consider one bit. The upper bound of remaining volumes can be reduced to 9, which is enough to fit in the time limit.

Actually, we can show the upper bound can be improved to 7 with ~~a subtle analysis~~ stress test.

题意

- 给定 $w_1, w_2, \cdots, w_n$，求

$$\sum_{x\in[n]^n} \prod_{i=1}^{n-2} w_{\max\{x_i, x_{i+1}, x_{i+2}\}} \bmod (10^9+7)$$

- 保证 $n \leq 2000$。

题解

- 直接暴力是 $O(n^3)$ 的
- 记录 x_{i-2} 和 x_{i-1} 的最大值以及位置即可
- 具体转移留作习题

题意

对于一棵 n 个点的树 T，定义 $f(T)$ 为:

- 若 $n = 1$ 则 $f(T) = 1$。
- 否则对于所有 $e \in T$，设 $T_{e,1}, T_{e,2}$ 为切掉这条边形成的两棵子树，那么 $f(T) = \frac{1}{n} \sum_e f(T_{e,1}) \times f(T_{e,2})$。

给定 T，求 $f(T) \bmod P$ 的值。

数据范围：$n \le 5000$。

AGC 058F

题解

- 我们希望给 $f(T)$ 一个组合意义。
- 比较人类智慧的是注意到 $x^{-1} = (x+P)^{-1}$。基于此，将每条边中间拆出一个点，然后挂 $P-1$ 个叶子。
- 那么 $f(T)$ 就是每个点赋随机实数权值，最终每条边中间的点权都大于所有邻点的概率。原题中的转移方程对应于枚举最大值，除以的总点数实际上是 $(n-1)P+n \equiv n \pmod P$。
- 这是一个经典问题，容斥后就是若干个内向树。DP 细节留作习题。

题意

给定两个长度为 n，且仅包含 `X` 或 `Y` 的字符串 S, T。对于每个整数 $i \in [1, n]$，你都可以选择交换 S_i, T_i 或者不交换。

请你最大化交换完后的最长公共子序列长度，并输出一组合法的最长公共子序列。如果有多种合法答案，请输出 **字典序最小的**。$n \le 50$。

ARC 157F

题解

- Key observation: 答案至少是 $\left\lfloor\frac{2n}{3}\right\rfloor$。
- Hint：考虑长为 3 的 01 串。

- 所以最终答案任意两个匹配的字符距离不超过 $\left\lceil\frac{n}{3}\right\rceil$。
- 状压 DP 复杂度为 $O\left(n2^{n/3}\right)$。

题意

两人玩游戏。游戏中有 n 个递增的序列 $a(1\sim n)$，还有另外 n 个序列 $b(1\sim n)$，b 初始为空。

两人轮流做以下操作：选出一个 $1\le i\le n$，从第 i 个序列中选出一个还没选出过的数，放在序列 $b(i)$ 的末尾，要求

1. $b(i)$ 保持递增
2. $b(i)$ 保持凸（即，任意 j 有 $b(i)_{j+1}-b(i)_j > b(i)_j - b(i)_{j-1}$）。

不能做操作的人输，问先手是否必胜。

$n\le 10^3$，序列长度和 $\le 10^5$，序列中元素 $\le 10^5$

CF 1434E

题解

- 只需要对每个序列分别求出 SG 值，显然 SG 值总是 $O(\sqrt{V})$ 级别的（考虑最长转移链）。
- 考虑 DP，我们需要记录最后选取了哪个元素以及最后两个元素的差。一种常见的想法是交换值域，即记录 $f(i,d)$ 表示以 i 结尾，SG 值为 d 时最后两个元素的差的最大值。
- 倒着做。转移时枚举 $k > i$，用 $f(k,j)$ 更新 $f(i,j+1)$。我们需要一个数据结构支持区间对一个数取 max 以及单点查询，可以根据单调性用 $O(\sqrt{V})$ 个并查集完成区间覆盖。
- 时间复杂度为 $O\left(nV + L\sqrt{V} \cdot \mathrm{alpha}(L)\right)$。

题意

- 有 n 个箱子（第 i 个箱子的价值为 a_i）和 m 种类型的锁（第 i 种类型的锁对应钥匙的价值为 b_i）。
- 首先 Alice 会给每个箱子加上某些类型的锁，给第 i 个箱子加上第 j 种类型的锁需要花费 $c_{i,j}$。
- 然后 Bob 会购买某些类型的钥匙，然后用这些钥匙来打开他能打开的箱子。
- Alice 需要让 Bob 无法获利，即打开的箱子的价值之和总是不超过钥匙的价值之和，求 Alice 配置锁需要花费的最小代价。
- 保证 $n, m \leq 6, a_i, b_j \leq 4, c_{i,j} \leq 10^7$。

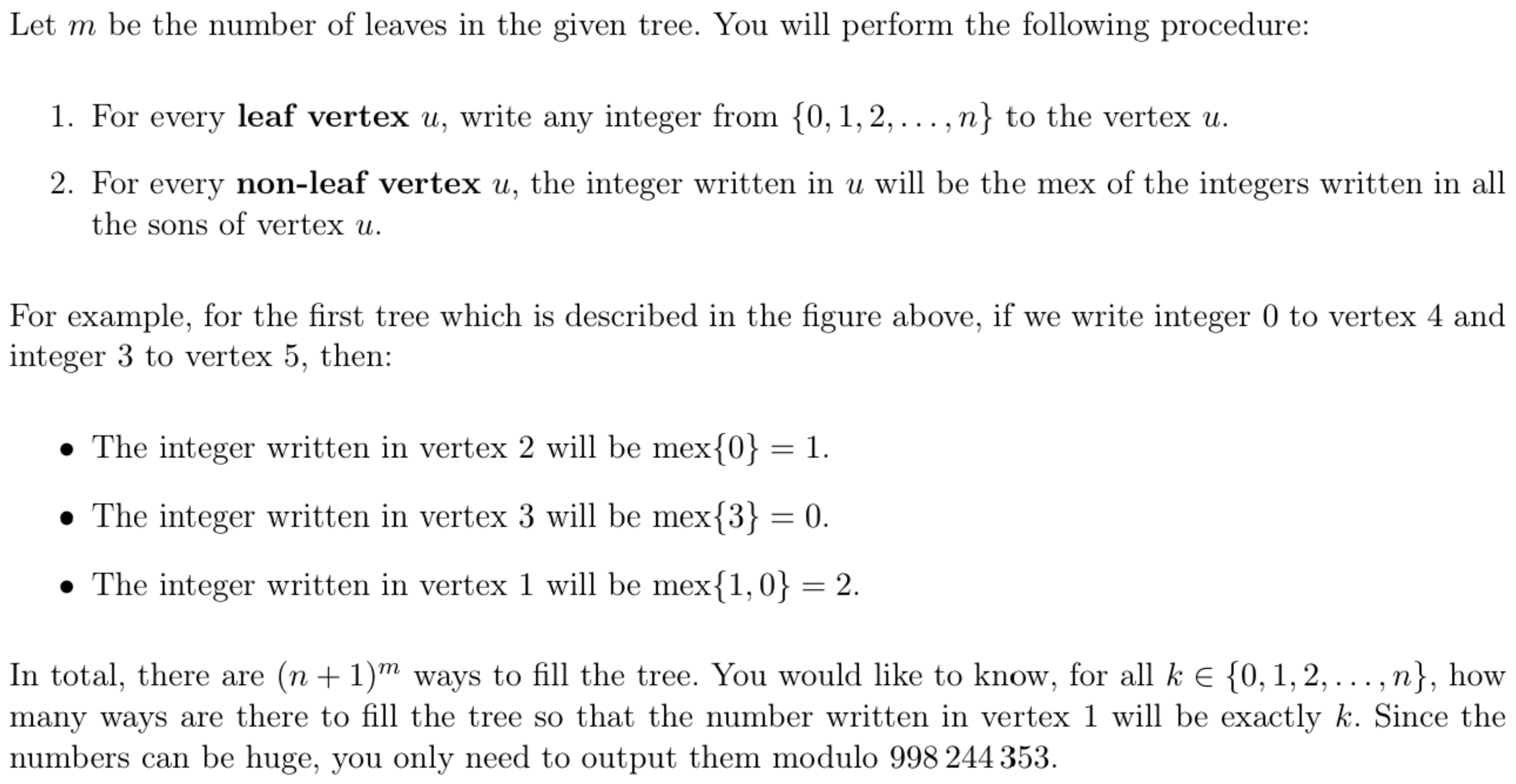
CF 1519F

题解

- 考虑固定锁的配置，判断 Bob 能否获利。可以用最小割：
  - 建立左部 n 个点、右部 m 个点的二分图。从源点向左部第 i 个点连接容量为 a_i 的边，从右部第 j 个点向汇点连接容量为 b_j 的边。第 i 个箱子加上了第 j 种类型的锁，则从左部第 i 个点向右部第 j 个点连接容量为 $+\infty$ 的边。
- Bob 的获利为 $\sum a_i$ 减去最小割，必须为 0。用最大流的视角，即从源点出发的边必须满流。
- 因为 a_i, b_j 非常小，记录 $(i, j, r;\ f_1, f_2, \cdots, f_m)$ 进行 DP，表示当前考虑到了左部的第 i 个点，已经向右部的前 j 个点流了 r 的流量，且右部的点到汇点的当前流量为 $f_1, f_2, \cdots, f_m$ 的最小代价。
- 时间复杂度为 $O(nm \cdot a^2 b^m)$。

题意

- 假如还有时间的话就讲



Let m be the number of leaves in the given tree. You will perform the following procedure:

1. For every **leaf vertex** u, write any integer from $\{0, 1, 2, \dots, n\}$ to the vertex u.
2. For every **non-leaf vertex** u, the integer written in u will be the mex of the integers written in all the sons of vertex u.

For example, for the first tree which is described in the figure above, if we write integer 0 to vertex 4 and integer 3 to vertex 5, then:

- The integer written in vertex 2 will be $\operatorname{mex}\{0\} = 1$.
- The integer written in vertex 3 will be $\operatorname{mex}\{3\} = 0$.
- The integer written in vertex 1 will be $\operatorname{mex}\{1, 0\} = 2$.

In total, there are $(n + 1)^m$ ways to fill the tree. You would like to know, for all $k \in \{0, 1, 2, \dots, n\}$, how many ways are there to fill the tree so that the number written in vertex 1 will be exactly k. Since the numbers can be huge, you only need to output them modulo $998\,244\,353$.

QOJ 4811

题解

Let's denote the set of the sons of vertex u as S_u, and $d_u = |S_u|, \delta_u = \max_{v \in S_u} d_v, t_u = \sum_{v \in S_u} [d_v > 0]$. Note that for all non-leaf vertex u, the number written on vertex u will be between 0 and d_u.

Our first insight is using dynamic programming to solve the problem. Let $f_{u,v}$ denotes the number of ways to fill the subtree of the vertex u, so that the number writing on vertex u will be exactly v.

Part 1

The first way to calculate f is using the inclusion-exclusion principle. Let dp_S be the number of ways if all the numbers written on its children don't belong to S. First, it seems we need $O(2^{d_u})$ states, but note that we don't need to add the contribution of the leaf children to the state. So there are only $O(2^{\delta_u})$ states.

Part 2

The second way to calculate f is using bitmasks dp. Let $f_{i,S}$ be the number of ways that:

- Let's fill the numbers from small to large,
- We have considered the numbers $0, 1, \cdots, i$,
- The set of unfilled sons is S

There will be $O(d_u \cdot 2^{d_u})$ states. But we can also treat all leaves independently. Let's add a dimension $[k]$ in our dp, which means an additional constraint that there are k leaves that have been filled. By using SOS dp, the time complexity will be $O(2^{t_u} \cdot \text{poly}(t_u))$

题解

Part 3

If we use these two ways at the same time, we will have an $O(\min(2^{t_u} \cdot \text{poly}(t_u), 2^{\delta_u} \cdot \text{poly}(\delta_u)))$ solution, but it's still not enough to pass it, because t_u and δ_u can be large at the same time.

So let's try to split the set of the sons of u and use an sqrt-decomposition-like process. For a threshold B, let's split all the sons with $d_v \leq B$ to the set A_u, and the others to the set B_u.

Let's run the DP described in Part 1 for A_u, and run the DP described in Part 2 for B_u simultaneously. When solving A_u, we need to know the states in B_u. Let $l = |B_u|$, then the time complexity will be $O(2^{B+l})$.

Note that if all $B + l$ is at least M, then we have $\sum_v d_v \geq \frac{M(M+1)}{2}$, but $\sum_i d_i = n - 1$, so $M = O(\sqrt{n})$ (or more precisely, $M \sim \sqrt{2n}$).

By selecting an optimal M while doing the dp process, the time complexity will be $O(2^{\sqrt{2n}} \cdot \text{poly}(n))$