序列

$n, m, q \leq 20$

对于初始状态和每次修改后，dfs 整棵树的同时暴力枚举该点选不选进序列里即可。

时间复杂度：$O(q2^n)$。

$n, m, q \leq 400$

对于初始状态和每次修改后，设 f_i 为以 i 结尾的子序列 $[1, 2, 3, \ldots, a_i]$ 数量，那么当 $a_i = 1$ 的时候 $f_i = 1$，否则枚举每个祖先 j 如果满足 $a_j = a_i - 1$ 那么就将 f_j 加进 f_i 里。

时间复杂度：$O(qn^2)$

$n, m, q \leq 5000$ **或** $q \leq 50$

对于初始状态和每次修改后，直接设 sum_v 代表 i 的祖先 j 中满足 $a_j = v$ 的 f_j 之和，特别地 $sum_0 = 1$，那么此时 $f_i = sum_{a_i - 1}$，省去了暴力枚举祖先的过程，且 sum_v 的修改，撤销都是 $O(1)$ 的。

时间复杂度：$O(qn)$

$m \leq 50$

考虑将修改每 n 个分一块修改，当修改到第 i 个点的时候，由于前 $i - 1$ 已被修改过，只需要查询当前的 $sum_{a_i - 1}$ 并修改到新的 f_i 中。

但因为我们还要快速查询答案，注意到原来所有经过点 i 的子序列都会被删除，新增的全是经过点 i 的子序列，而子序列的前 $i - 1$ 个点来自祖先，后 $n - i$ 个点来自后代，所以我们还需要查询 i 子树内子序列 $[a_i + 1, a_i + 2, \ldots, m]$ 数量。

不妨设 $g_{u,i}$ 为 u 子树中子序列 $[i, i + 1, \ldots, m]$ 的数量，转移是好转移的，但状态数为 $O(nm)$。

时间复杂度：$O((n + q)m)$

正解

能不能进一步优化，注意到我们其实只需要知道 $g_{u, a_u + 1}$ 即可，不妨直接记 g_i 为当前枚举到的所有节点中，子序列 $[i, i + 1, \ldots, m]$ 的数量，那么进入 u 子树之前，我们记录下当前的 $g_{a_u + 1}$，当遍历完 u 子树后，我们可以得到目前的 $g_{a_u + 1}$，二者一相减就是子树内的子序列 $[a_u + 1, a_u + 2, \ldots, m]$ 数量。

时间复杂度：$O(n + q)$

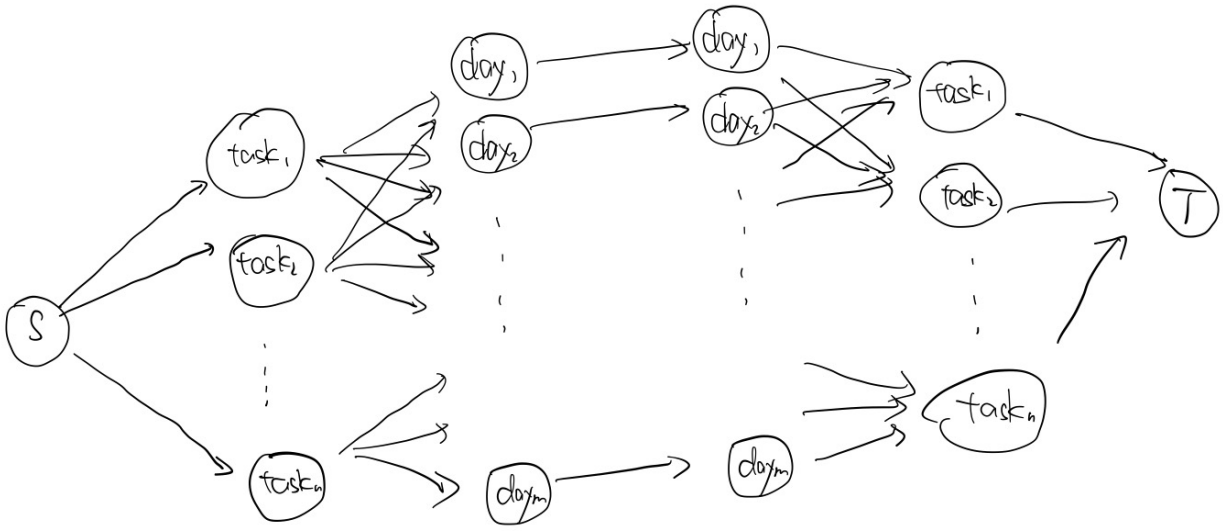
此外，一些使用数据结构维护的带 log 做法也可以通过本题。

Designant.

首先注意到，在确定好每天选择的任务类型后，蚂蚁🐜做出的决策实际上能够最大化它们完成的任务数量；也就是说我们可以把问题变成，蚂蚁🐜会选择当天任务类型正确的任务中的任意一个，我们需要最小化蚂蚁🐜完成的任务数量的最大值。

考虑建两张二分图 G_0, G_1，G_i 的左部点是类型为 i 的所有任务，右部点是 $1, 2, \cdots, m$ 代表每一天，如果第 x 个任务可以在第 y 天完成就把左边 x 连向右边 y。那么选择 w 序列就相当于选择一个集合 $S \subseteq \{1, 2, \cdots, m\}$，然后在 G_0 的右部点中只保留 S 中的元素，G_1 的左部点中只保留 $\{1, 2, \cdots, m\} \setminus S$ 中的元素，接下来求一个最大匹配；我们需要最小化这两边最大匹配的和。

我们建一张图，如图所示：



其中左边的 task 是类型为 0 的所有任务，右边的是类型为 1 的；$\mathrm{task}_i \to \mathrm{day}_j$ 有连边当且仅当第 j 天可以完成第 i 个任务，即 $s_i \le j \le t_i$。这里，所有边的容量均为 1。

那么，这张图中 $S \to T$ 的最大流就是答案。

证明：我们考虑，首先如果有一个最大流，它流过的中间边（即左边 day_i 到右边 day_i 的边）的集合为 A，那么不管 A 集合中的点放在哪边，都一定可以在匹配内，所以答案不会比最大流的大小更小。

另一方面，在求出最大流后，考虑残量网络中的每个点，由于残量网络中不存在 $S \to T$ 的路径，因此每个点要么 S 能到它，要么它能到 T。那么我们就把 S 能到达的所有 day_i 放到 T 那边，剩下的放到 S 那边。

于是可以直接暴力建图跑最大流，有 $O(n+m)$ 个点，$O(nm)$ 条边。

由于图是单位网络，时间复杂度可以看做 $O(nm\sqrt{nm})$，可以通过 $n, m \le 400$。

注意到连边是区间连边的形式，因此可以线段树优化建图。

这样边数就只有 $O(n \log m)$ 条，时间复杂度 $O(n \log m \sqrt{n \log m})$，可以获得满分。

长野原龙势流星群 II

我们考虑树形 DP，用 $dp_{x,i}$ 表示在以 x 为根的子树中，大小为 i 的包含 x 的连通块的点权和的最大值。

然而，$dp_{x,i}$ 本身并不是凸的，不过注意到以下结论：

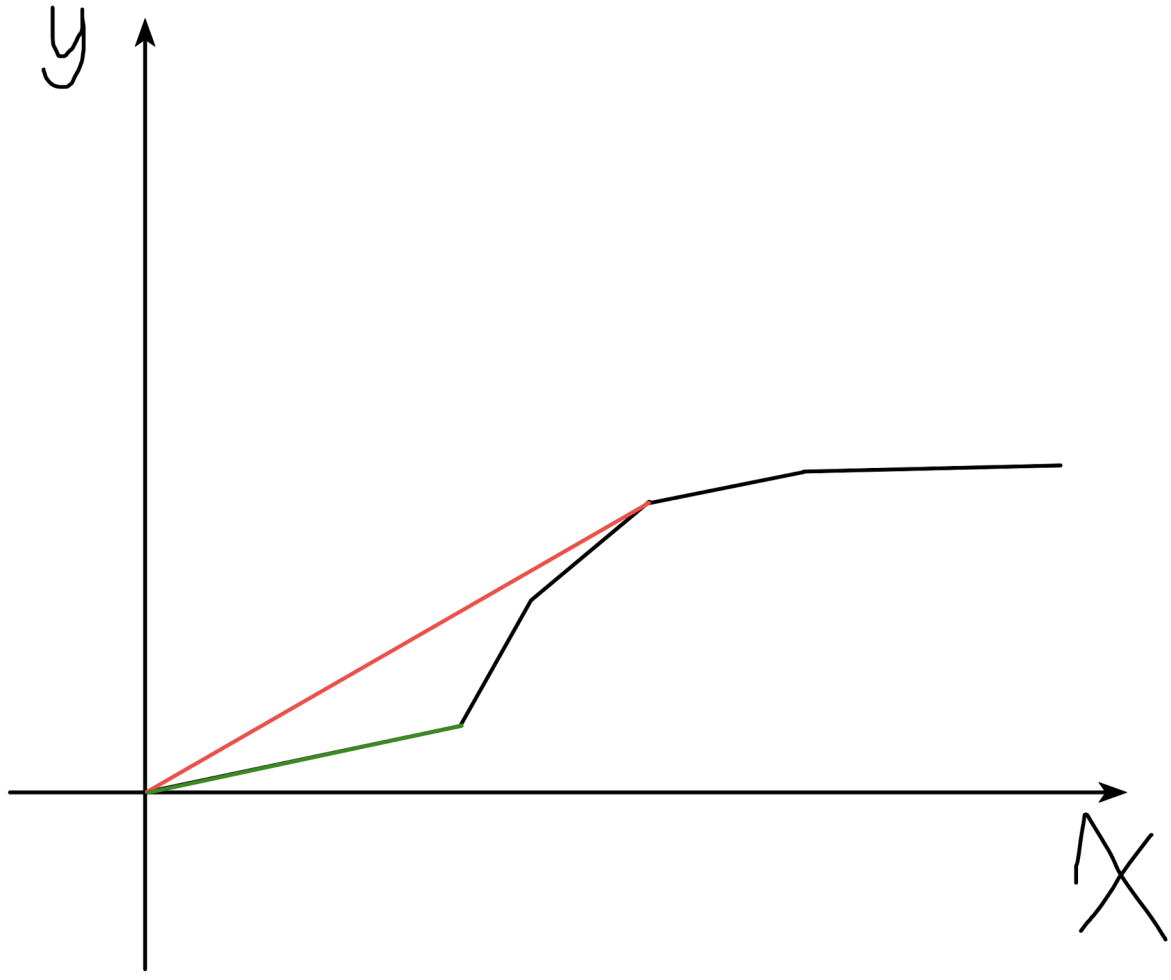
- 对于平面上的若干个点 (x_i, y_i)，这些点和原点连线的斜率的最大值只会在它们的上凸壳中取到。

证明只需要考虑三个点，如果形成下凸，那么至少有一个点比中间的那个点更优。如此去除无用的点，最后只会剩下上凸壳中的点。

我们记 f_x 表示 x 子树内 DP 值构成的凸包。树上背包合并的过程可以看作是：

1. 先将所有子节点的凸包做闵可夫斯基和；
2. 加入节点 x，并合并一段前缀的边来保证凸性。

以下为过程示意图，其中绿色边是 $(1, a_x)$，后面黑色边是原凸包上的边：



我们记 F_x 表示 x 子树外 DP 值构成的凸包。

在换根的过程中可以利用重链剖分进行维护。记结点 x 的重儿子为 $wson_x$。

在过程中维护以下数据：

- 到根路径经过的重儿子编号集合 S_x；
- 全局的凸包 G_i，表示到 x 的距离 $\geq i$ 的点到 x 的距离和。

假设在当前节点 x，步骤如下：

1. 将所有轻儿子加入 F；
2. 对于重儿子：不需要做额外处理；
3. 对于轻儿子：删除结点 y 在 F 内的贡献，然后将重儿子的结点编号 $wson_x$ 加入 S_y。

换根过程中，维护 $O(\log n)$ 个数据结构中加入当前向量，保持凸包的过程。

算法 1：与值域相关的做法

我们考虑使用权值线段树来维护。

虽然维护的是分数，但是实际上，对于任意两个不同的分数 $\frac{b}{a}$ 和 $\frac{d}{c}$，满足 $1 \leq a, c \leq n$，一定有

$$\left|\frac{b}{a}-\frac{d}{c}\right| = \frac{|bc-ad|}{ac} \geq \frac{1}{n(n-1)}$$

所以我们可以把所有数都乘上 n^2 并下取整，然后用权值线段树维护。

由于合并一定是将斜率 $\leq x$ 的全部合并，所以每次查询时，利用权值线段树的结构相同，可以在过程中同时二分。

时间复杂度为 $O(n \log n(\log n + \log a))$，空间复杂度为 $O(n(\log n + \log a))$

算法 2：不基于值域的做法

首先，假设当前向量和所有平衡树的左子树以及树根合并后，斜率比树根斜率最小的大。

那么，说明斜率最小的那棵平衡树的树根和右子树都不需要被合并，于是可以递归到左子树。

定理

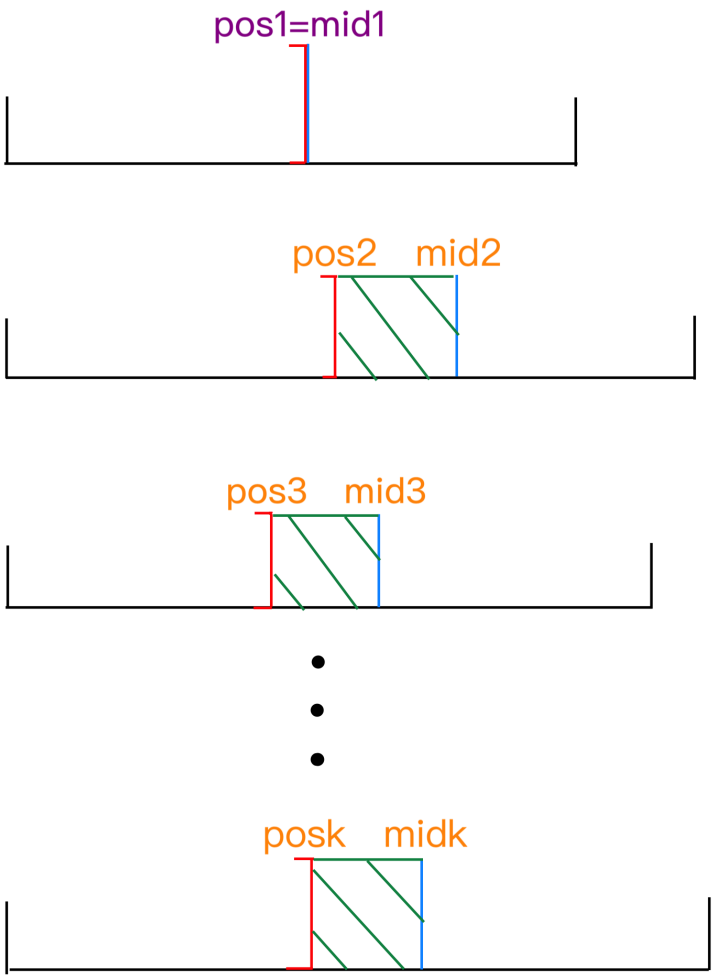
斜率最大的那棵平衡树的左子树和树根都必须要被合并。

证明

假设其不应被合并。

令每个数组斜率 $\geq$ 最大的位置为 pos_i，则由于当前平衡树的树根斜率最大，所以有 $pos_i \leq mid_i$。

以下为示意图：



此时，考虑加入绿色部分：

- 由于绿色部分的斜率也 $\geq$ 平衡树斜率最小的树根的斜率；
- 当前向量再与绿色部分合并完后，斜率一定 $\geq$ 平衡树斜率最小的树根的斜率。

这与假设矛盾，因此原命题成立。

由于找斜率最大、最小的树根需要用可删堆维护，这部分时间复杂度带一个 $O(\log \log n)$。

时间复杂度

- 时间复杂度：$O(n \log^2 n \log \log n)$
- 空间复杂度：$O(n \log n)$

综上，我们可以在 $O(n \log n(\log n + \log a))$ 的时间复杂度，$O(n(\log n + \log a))$ 的空间复杂度，或 $O(n \log^2 n \log \log n)$ 的时间复杂度，$O(n \log n)$ 的空间复杂度内解决本题。