DP 杂题选讲

朱羿恺

北京大学

June 24, 2024

目录

「Dwango Programming Contest 6th」Cookie Distribution

- 有 n 个孩子，在接下来的 k 天里，会给这些孩子一些曲奇。
- 在第 i 天，将会等概率选取 a_i 个孩子并给选中的孩子一人一个曲奇。
- 定义 k 天后，孩子们的高兴值为 $\prod_{i=1}^{n} c_i$，其中 c_i 表示第 i 个孩子手里的曲奇数量。
- 求高兴值的期望乘上 $\prod_{i=1}^{n} \binom{n}{a_i}$ 后的值，答案对 10^9+7 取模。
- $1 \le n \le 2000, 1 \le k \le 20$

solution

- 考虑组合意义，$\prod_{i=1}^{n} c_i$ 相当于每个孩子还要从手中的曲奇中再选一个。

solution

- 考虑组合意义，$\prod_{i=1}^{n} c_i$ 相当于每个孩子还要从手中的曲奇中再选一个。
- 考虑据此 DP，令 $dp_{i,j}$ 表示前 i 天里已经有 j 个孩子选过曲奇了。

solution

- 考虑组合意义，$\prod_{i=1}^{n} c_i$ 相当于每个孩子还要从手中的曲奇中再选一个。
- 考虑据此 DP，令 $dp_{i,j}$ 表示前 i 天里已经有 j 个孩子选过曲奇了。
- 转移就考虑枚举当前天另外有几个孩子选到了曲奇，然后剩下的曲奇随便分。

solution

- 考虑组合意义，$\prod_{i=1}^{n} c_i$ 相当于每个孩子还要从手中的曲奇中再选一个。
- 考虑据此 DP，令 $dp_{i,j}$ 表示前 i 天里已经有 j 个孩子选过曲奇了。
- 转移就考虑枚举当前天另外有几个孩子选到了曲奇，然后剩下的曲奇随便分。
    - $dp_{i,j+k} \leftarrow dp_{i,j+k} + dp_{i-1,j} \times \binom{n-j}{k} \times \binom{n-k}{a_i-k}$

solution

- 考虑组合意义，$\prod_{i=1}^{n} c_i$ 相当于每个孩子还要从手中的曲奇中再选一个。
- 考虑据此 DP，令 $dp_{i,j}$ 表示前 i 天里已经有 j 个孩子选过曲奇了。
- 转移就考虑枚举当前天另外有几个孩子选到了曲奇，然后剩下的曲奇随便分。
  - $dp_{i,j+k} \leftarrow dp_{i,j+k} + dp_{i-1,j} \times \binom{n-j}{k} \times \binom{n-k}{a_i-k}$
- 时间复杂度 $O(n^2k)$。

- 有 n 个社员。第 i 个社员会在时刻 S_i 时刻离开公司，会在 T_i 时刻回到公司。保证不会同时有 2 个人同时离开或者回到公司。
- 该公司大楼有一个大门作为入口，所有社员必须从这个门进出公司。在公司内部可以自由的打开或者关闭这个门。但是在公司外的话，必须有钥匙才能打开或者关闭这个门。在时刻 0，门是关闭的。因此，第 i 个社员能在 T_i 时刻回到公司，当且仅当在 T_i 时刻的时候门开着，或者他有钥匙。
- 进门的社员和出门且有钥匙的社员可以选择是否去关门。当没有钥匙的社员离开时，他们无法锁门。
- 你有 K 把钥匙，你需要把这些钥匙给其中 K 个社员，使得第 i 个社员在 T_i 时刻都能够进入公司，并且从 0 时刻到 M 时刻，门被关着的时间最大。
- $n \le 2000$

solution

- 对每个时间间隔分类讨论：

solution

- 对每个时间间隔分类讨论：
  - i 进 $i+1$ 出，这个间隔可以直接算进答案；

solution

- 对每个时间间隔分类讨论：
  - i 进 $i+1$ 出，这个间隔可以直接算进答案；
  - i 进 $i+1$ 进，只有 $i+1$ 有钥匙这个间隔才能有贡献；

solution

- 对每个时间间隔分类讨论：
  - i 进 $i+1$ 出，这个间隔可以直接算进答案；
  - i 进 $i+1$ 进，只有 $i+1$ 有钥匙这个间隔才能有贡献；
  - i 出 $i+1$ 出，只有 i 有钥匙这个间隔才能有贡献；

solution

- 对每个时间间隔分类讨论：
  - i 进 $i+1$ 出，这个间隔可以直接算进答案；
  - i 进 $i+1$ 进，只有 $i+1$ 有钥匙这个间隔才能有贡献；
  - i 出 $i+1$ 出，只有 i 有钥匙这个间隔才能有贡献；
  - i 出 $i+1$ 进，需要两个人都有钥匙这个间隔才能有贡献。

solution

- 对每个时间间隔分类讨论：
  - i 进 $i+1$ 出，这个间隔可以直接算进答案；
  - i 进 $i+1$ 进，只有 $i+1$ 有钥匙这个间隔才能有贡献；
  - i 出 $i+1$ 出，只有 i 有钥匙这个间隔才能有贡献；
  - i 出 $i+1$ 进，需要两个人都有钥匙这个间隔才能有贡献。
- 可以发现只有一种情况是和两端的人都有关，考虑按这种情况给对应的两个人连边的话，这样产生的图将会是若干条链。

solution

- 对每个时间间隔分类讨论：
  - i 进 $i+1$ 出，这个间隔可以直接算进答案；
  - i 进 $i+1$ 进，只有 $i+1$ 有钥匙这个间隔才能有贡献；
  - i 出 $i+1$ 出，只有 i 有钥匙这个间隔才能有贡献；
  - i 出 $i+1$ 进，需要两个人都有钥匙这个间隔才能有贡献。
- 可以发现只有一种情况是和两端的人都有关，考虑按这种情况给对应的两个人连边的话，这样产生的图将会是若干条链。
- 对每条链分别 DP，最后合并即可。

solution

- 对每个时间间隔分类讨论：
  - i 进 $i+1$ 出，这个间隔可以直接算进答案；
  - i 进 $i+1$ 进，只有 $i+1$ 有钥匙这个间隔才能有贡献；
  - i 出 $i+1$ 出，只有 i 有钥匙这个间隔才能有贡献；
  - i 出 $i+1$ 进，需要两个人都有钥匙这个间隔才能有贡献。
- 可以发现只有一种情况是和两端的人都有关，考虑按这种情况给对应的两个人连边的话，这样产生的图将会是若干条链。
- 对每条链分别 DP，最后合并即可。
- 时间复杂度 $O(n^2)$。

「JOISC 2020 Day4」治疗计划

- JOI 国有 n 个房屋，每个房屋有一个居民住在里面。
- 一开始，所有居民都感染了病毒。为了解决这个问题，有 m 个治疗方案被提出。第 i 个治疗方案的花费为 C_i。如果执行计划 i，则会发生以下事件：
  - 在第 T_i 天的晚上，$[L_i, R_i]$ 内的居民都会被治愈。
- 病毒按如下方式传染相邻的居民：
  - 如果在某天的早晨，居民 x 被病毒感染，那么在同一天的中午，居民 $x-1$ 和居民 $x+1$ 就会被感染。
- 一个已经被治愈的居民可以再次被病毒感染。
- 你需要选取某些方案，使得在所有被选中的方案全部执行后，没有居民感染病毒。求最小可能花费。
- $1 \le n \le 10^9, 1 \le m \le 10^5$

solution

- 考虑 DP，令 f_i 表示 T_i 时刻 $[1, R_i]$ 的人都治好了的最小花费。也就是我们是对人考虑的，每次会把治好的前缀往右延伸一点。

solution

- 考虑 DP，令 f_i 表示 T_i 时刻 $[1, R_i]$ 的人都治好了的最小花费。也就是我们是对人考虑的，每次会把治好的前缀往右延伸一点。
- 考虑转移：

$$dp_i = \min_{R_j - L_i + 1 \geq |T_i - T_j|} (dp_j + c_i)$$

solution

- 考虑 DP，令 f_i 表示 T_i 时刻 $[1, R_i]$ 的人都治好了的最小花费。也就是我们是对人考虑的，每次会把治好的前缀往右延伸一点。
- 考虑转移：

$$dp_i = \min_{R_j - L_i + 1 \geq |T_i - T_j|} (dp_j + c_i)$$

- 由于还受到时间的影响，很难设计一个直接的转移方式。

solution

- 考虑 DP，令 f_i 表示 T_i 时刻 $[1, R_i]$ 的人都治好了的最小花费。也就是我们是对人考虑的，每次会把治好的前缀往右延伸一点。
- 考虑转移：

$$dp_i = \min_{R_j - L_i + 1 \geq |T_i - T_j|} (dp_j + c_i)$$

- 由于还受到时间的影响，很难设计一个直接的转移方式。
- 于是考虑类似于 dij 的更新方式，用堆维护所有被更新过的状态，每次取出堆中最小的去转移。

solution

- 考虑 DP，令 f_i 表示 T_i 时刻 $[1, R_i]$ 的人都治好了的最小花费。也就是我们是对人考虑的，每次会把治好的前缀往右延伸一点。
- 考虑转移：

$$dp_i = \min_{R_j - L_i + 1 \geq |T_i - T_j|} (dp_j + c_i)$$

- 由于还受到时间的影响，很难设计一个直接的转移方式。
- 于是考虑类似于 dij 的更新方式，用堆维护所有被更新过的状态，每次取出堆中最小的去转移。
- 注意到一个点 j 如果当前能被转移到，且之前没被转移过，这个点的值就是 $f_i + C_j$，因为每次取出的是最小的且 i 转移到 j 需要再加上的值只和 j 有关。

solution

- 考虑 DP，令 f_i 表示 T_i 时刻 $[1, R_i]$ 的人都治好了的最小花费。也就是我们是对人考虑的，每次会把治好的前缀往右延伸一点。
- 考虑转移：

$$dp_i = \min_{R_j - L_i + 1 \geq |T_i - T_j|} (dp_j + c_i)$$

- 由于还受到时间的影响，很难设计一个直接的转移方式。
- 于是考虑类似于 dij 的更新方式，用堆维护所有被更新过的状态，每次取出堆中最小的去转移。
- 注意到一个点 j 如果当前能被转移到，且之前没被转移过，这个点的值就是 $f_i + C_j$，因为每次取出的是最小的且 i 转移到 j 需要再加上的值只和 j 有关。
- 因此考虑用线段树维护还没被转移的点，每次暴力找出能转移到且还转移到的点更新就行。

solution

- 考虑 DP，令 f_i 表示 T_i 时刻 $[1, R_i]$ 的人都治好了的最小花费。也就是我们是对人考虑的，每次会把治好的前缀往右延伸一点。
- 考虑转移：

$$dp_i = \min_{R_j - L_i + 1 \geq |T_i - T_j|} (dp_j + c_i)$$

- 由于还受到时间的影响，很难设计一个直接的转移方式。
- 于是考虑类似于 dij 的更新方式，用堆维护所有被更新过的状态，每次取出堆中最小的去转移。
- 注意到一个点 j 如果当前能被转移到，且之前没被转移过，这个点的值就是 $f_i + C_j$，因为每次取出的是最小的且 i 转移到 j 需要再加上的值只和 j 有关。
- 因此考虑用线段树维护还没被转移的点，每次暴力找出能转移到且还转移到的点更新就行。
- 时间复杂度 $O(m \log m)$。

「JOISC 2020 Day3」星座 3

- 有一张 $N \times N$ 的星空图，将左起第 X 列，下起第 Y 行的像素点称为像素 (X, Y)。
- 画面里有白色的大楼，黄色的星星，黑色的空格。第 i 列从最下方到自下数起第 A_i 行都是白色的大楼。有 M 个星星，第 j 个星星位于像素点 (X_j, Y_j)。此外，所有的像素点都是黑色。
- 若一个长方形区域可以称作星座，则满足以下条件：
  1. 不含白色像素点。
  2. 至少存在两个星星。
- 你需要删除一些星星，使得没有星座存在。将第 j 个星星涂成黑色会使照片的不自然度增加 C_j，最初不自然度为 0。求不自然度的最小值。
- $1 \le N, M \le 2 \times 10^5$

solution

- 先转为最大化保留的。

solution

- 先转为最大化保留的。
- 建出关于 A_i 的笛卡尔树，然后从叶子开始往上考虑。

solution

- 先转为最大化保留的。
- 建出关于 A_i 的笛卡尔树，然后从叶子开始往上考虑。
- 首先对于叶子的那一列，显然至多只能保留一个。

solution

- 先转为最大化保留的。
- 建出关于 A_i 的笛卡尔树，然后从叶子开始往上考虑。
- 首先对于叶子的那一列，显然至多只能保留一个。
- 然后每次合并两个孩子的时候，注意到我们只需要关心两个孩子子树里保留的最高的星星。

solution

- 先转为最大化保留的。
- 建出关于 A_i 的笛卡尔树，然后从叶子开始往上考虑。
- 首先对于叶子的那一列，显然至多只能保留一个。
- 然后每次合并两个孩子的时候，注意到我们只需要关心两个孩子子树里保留的最高的星星。
- 于是考虑设计状态 $f_{u,i}$ 表示考虑了 u 子树对应列的所有星星，保留的最高的星星高度为 i 的答案。

solution

- 先转为最大化保留的。
- 建出关于 A_i 的笛卡尔树，然后从叶子开始往上考虑。
- 首先对于叶子的那一列，显然至多只能保留一个。
- 然后每次合并两个孩子的时候，注意到我们只需要关心两个孩子子树里保留的最高的星星。
- 于是考虑设计状态 $f_{u,i}$ 表示考虑了 u 子树对应列的所有星星，保留的最高的星星高度为 i 的答案。
- 合并两个子树就是:

- 先转为最大化保留的。
- 建出关于 A_i 的笛卡尔树，然后从叶子开始往上考虑。
- 首先对于叶子的那一列，显然至多只能保留一个。
- 然后每次合并两个孩子的时候，注意到我们只需要关心两个孩子子树里保留的最高的星星。
- 于是考虑设计状态 $f_{u,i}$ 表示考虑了 u 子树对应列的所有星星，保留的最高的星星高度为 i 的答案。
- 合并两个子树就是：
  - $f_{u,i} = \max(f_{ls,i} + \max_{k \le A_u} f_{rs,k}, f_{rs,i} + \max_{k \le A_u} f_{ls,k})$

solution

- 先转为最大化保留的。
- 建出关于 A_i 的笛卡尔树，然后从叶子开始往上考虑。
- 首先对于叶子的那一列，显然至多只能保留一个。
- 然后每次合并两个孩子的时候，注意到我们只需要关心两个孩子子树里保留的最高的星星。
- 于是考虑设计状态 $f_{u,i}$ 表示考虑了 u 子树对应列的所有星星，保留的最高的星星高度为 i 的答案。
- 合并两个子树就是：
  - $f_{u,i} = \max(f_{ls,i} + \max_{k \le A_u} f_{rs,k}, f_{rs,i} + \max_{k \le A_u} f_{ls,k})$
  - 对于 u 对应列上的一个星星 (X_j, Y_j)，有转移：

$$f_{u,y} = \max(f_{u,y}, \max_{k \le A_u} f_{rs,k} + \max_{k \le A_u} f_{ls,k} + C_j)$$

solution

- 先转为最大化保留的。
- 建出关于 A_i 的笛卡尔树，然后从叶子开始往上考虑。
- 首先对于叶子的那一列，显然至多只能保留一个。
- 然后每次合并两个孩子的时候，注意到我们只需要关心两个孩子子树里保留的最高的星星。
- 于是考虑设计状态 $f_{u,i}$ 表示考虑了 u 子树对应列的所有星星，保留的最高的星星高度为 i 的答案。
- 合并两个子树就是：
  - $f_{u,i} = \max(f_{ls,i} + \max\limits_{k \le A_u} f_{rs,k}, f_{rs,i} + \max\limits_{k \le A_u} f_{ls,k})$
  - 对于 u 对应列上的一个星星 (X_j, Y_j)，有转移：

$$f_{u,y} = \max(f_{u,y}, \max_{k \le A_u} f_{rs,k} + \max_{k \le A_u} f_{ls,k} + C_j)$$

- 用线段树合并维护 DP 即可，时间复杂度 $O(n \log n)$。

AGC038E Gachapon

- 有一个随机数生成器，其会以 $\frac{A_i}{\sum_{j=0}^{n-1} A_j}$ 的概率生成 i，其中 $i \in [n]$。
- 这个随机数生成器会不断生成随机数，直到 $\forall i \in [n]$，i 至少被生成了 B_i 次。
- 求期望生成随机数的次数，答案对 998244353 取模。
- $1 \leq n, \sum_{i=1}^{n} A_i, \sum_{i=1}^{n} B_i \leq 400$

solution

- 考虑 min-max 容斥，问题转化为，枚举一个 $[n]$ 的子集，计算该子集中存在一个数被满足要求时步数的期望，考虑再转化为 i 次还没达到条件的概率的和。

solution

- 考虑 min-max 容斥，问题转化为，枚举一个 $[n]$ 的子集，计算该子集中存在一个数被满足要求时步数的期望，考虑再转化为 i 次还没达到条件的概率的和。
- 也就是对于每一个未结束的局面算出到达该局面的概率，该局面对于答案的贡献是概率乘上保持该局面的概率。

solution

- 考虑 min-max 容斥，问题转化为，枚举一个 $[n]$ 的子集，计算该子集中存在一个数被满足要求时步数的期望，考虑再转化为 i 次还没达到条件的概率的和。
- 也就是对于每一个未结束的局面算出到达该局面的概率，该局面对于答案的贡献是概率乘上保持该局面的概率。
- 考察一下一个未结束的局面的贡献的具体式子：

$$(-1)^{|S|-1} \times \left(\sum_{i=0}^{\infty} \left(\frac{\sum_{i \notin S} A_i}{\sum_{i=1}^{n} A_i} \right)^i \right) \times \frac{(\sum_{i \in S} x_i)!}{\prod_{i \in S} x_i!} \times \prod_{i \in S} \left(\frac{A_i}{\sum_{j \in S} A_j} \right)^{x_i}$$

solution

- 考虑 min-max 容斥，问题转化为，枚举一个 $[n]$ 的子集，计算该子集中存在一个数被满足要求时步数的期望，考虑再转化为 i 次还没达到条件的概率的和。
- 也就是对于每一个未结束的局面算出到达该局面的概率，该局面对于答案的贡献是概率乘上保持该局面的概率。
- 考察一下一个未结束的局面的贡献的具体式子：

$$(-1)^{|S|-1} \times \left(\sum_{i=0}^{\infty} \left(\frac{\sum_{i \notin S} A_i}{\sum_{i=1}^{n} A_i} \right)^i \right) \times \frac{(\sum_{i \in S} x_i)!}{\prod_{i \in S} x_i!} \times \prod_{i \in S} \left(\frac{A_i}{\sum_{j \in S} A_j} \right)^{x_i}$$

- 其中 x_i 表示 i 的出现次数。

solution

- 考虑 min-max 容斥，问题转化为，枚举一个 $[n]$ 的子集，计算该子集中存在一个数被满足要求时步数的期望，考虑再转化为 i 次还没达到条件的概率的和。
- 也就是对于每一个未结束的局面算出到达该局面的概率，该局面对于答案的贡献是概率乘上保持该局面的概率。
- 考察一下一个未结束的局面的贡献的具体式子：

$$(-1)^{|S|-1} \times \left(\sum_{i=0}^{\infty} \left(\frac{\sum_{i \notin S} A_i}{\sum_{i=1}^{n} A_i}\right)^i\right) \times \frac{(\sum_{i \in S} x_i)!}{\prod_{i \in S} x_i!} \times \prod_{i \in S} \left(\frac{A_i}{\sum_{j \in S} A_j}\right)^{x_i}$$

- 其中 x_i 表示 i 的出现次数。
- 注意到最后贡献系数只和选中的集合的 A_i 和以及 x_i 和有关。

solution

- 考虑 min-max 容斥，问题转化为，枚举一个 $[n]$ 的子集，计算该子集中存在一个数被满足要求时步数的期望，考虑再转化为 i 次还没达到条件的概率的和。
- 也就是对于每一个未结束的局面算出到达该局面的概率，该局面对于答案的贡献是概率乘上保持该局面的概率。
- 考察一下一个未结束的局面的贡献的具体式子：

$$(-1)^{|S|-1} \times \left(\sum_{i=0}^{\infty} \left(\frac{\sum_{i \notin S} A_i}{\sum_{i=1}^{n} A_i} \right)^i \right) \times \frac{(\sum_{i \in S} x_i)!}{\prod_{i \in S} x_i!} \times \prod_{i \in S} \left(\frac{A_i}{\sum_{j \in S} A_j} \right)^{x_i}$$

- 其中 x_i 表示 i 的出现次数。
- 注意到最后贡献系数只和选中的集合的 A_i 和以及 x_i 和有关。
- 状态里记存这两维即可，直接转移复杂度就是 $O(\sum_i A_i (\sum_i B_i)^2)$ 的。

AGC033D Complexity

- 给定一个 N 行 M 列的 01 矩阵。
- 我们定义一个字符矩阵的凌乱度为：
  - 若这个字符矩阵中所有字符都相同，则凌乱度为 0。
  - 否则，则考虑所有的沿水平或者竖直方向的直线，将字符矩阵分成两个不为空的部分，设两个部分的凌乱度分别为 a 和 b，则整个字符矩阵的凌乱度为 $\max(a, b) + 1$ 的最小值。
- 求给出的字符矩阵的凌乱度是多少。
- $1 \le N, M \le 185$

solution

- 一个直接的想法是设 $g_{l,r,u,d}$ 表示左上角为 (u,l) 右下角为 (d,r) 的子矩形的凌乱度。

solution

- 一个直接的想法是设 $g_{l,r,u,d}$ 表示左上角为 (u,l) 右下角为 (d,r) 的子矩形的凌乱度。
- 但是这样状态数就有 $O(n^2m^2)$，没有前途。

solution

- 一个直接的想法是设 $g_{l,r,u,d}$ 表示左上角为 (u,l) 右下角为 (d,r) 的子矩形的凌乱度。
- 但是这样状态数就有 $O(n^2m^2)$，没有前途。
- 但是注意到一个 $n\times m$ 的矩形的凌乱度至多为 $\log(n)+\log(m)$，且显然在固定 l,u,d 后，DP 值是关于 r 单调的。

solution

- 一个直接的想法是设 $g_{l,r,u,d}$ 表示左上角为 (u,l) 右下角为 (d,r) 的子矩形的凌乱度。
- 但是这样状态数就有 $O(n^2m^2)$，没有前途。
- 但是注意到一个 $n \times m$ 的矩形的凌乱度至多为 $\log(n)+\log(m)$，且显然在固定 l,u,d 后，DP 值是关于 r 单调的。
- 考虑交换状态和 DP 的值！令 $f_{v,l,u,d}$ 表示最大的 r 使得 $g_{l,r,u,d}=v$。

solution

- 一个直接的想法是设 $g_{l,r,u,d}$ 表示左上角为 (u,l) 右下角为 (d,r) 的子矩形的凌乱度。
- 但是这样状态数就有 $O(n^2m^2)$，没有前途。
- 但是注意到一个 $n \times m$ 的矩形的凌乱度至多为 $\log(n)+\log(m)$，且显然在固定 l,u,d 后，DP 值是关于 r 单调的。
- 考虑交换状态和 DP 的值！令 $f_{v,l,u,d}$ 表示最大的 r 使得 $g_{l,r,u,d}=v$。
- 考虑转移：

$$f_{v+1,l,u,d}=\max\left(f_{v,f_{v,l,u,d}+1,u,d},\max_{mid=u}^{d-1}(\min(f_{v,l,u,mid},f_{v,l,mid+1,d}))\right)$$

solution

- 一个直接的想法是设 $g_{l,r,u,d}$ 表示左上角为 (u,l) 右下角为 (d,r) 的子矩形的凌乱度。
- 但是这样状态数就有 $O(n^2m^2)$，没有前途。
- 但是注意到一个 $n\times m$ 的矩形的凌乱度至多为 $\log(n)+\log(m)$，且显然在固定 l,u,d 后，DP 值是关于 r 单调的。
- 考虑交换状态和 DP 的值！令 $f_{v,l,u,d}$ 表示最大的 r 使得 $g_{l,r,u,d}=v$。
- 考虑转移：

$$f_{v+1,l,u,d}=\max\left(f_{v,f_{v,l,u,d}+1,u,d},\max_{mid=u}^{d-1}\left(\min(f_{v,l,u,mid},f_{v,l,mid+1,d})\right)\right)$$

- 注意到 $f_{v,l,u,d}$ 也显然具有单调性，因此转移可以使用三分优化，或者直接双指针。

solution

- 一个直接的想法是设 $g_{l,r,u,d}$ 表示左上角为 (u,l) 右下角为 (d,r) 的子矩形的凌乱度。
- 但是这样状态数就有 $O(n^2m^2)$，没有前途。
- 但是注意到一个 $n\times m$ 的矩形的凌乱度至多为 $\log(n)+\log(m)$，且显然在固定 l,u,d 后，DP 值是关于 r 单调的。
- 考虑交换状态和 DP 的值！令 $f_{v,l,u,d}$ 表示最大的 r 使得 $g_{l,r,u,d}=v$。
- 考虑转移：

$$f_{v+1,l,u,d}=\max\left(f_{v,f_{v,l,u,d}+1,u,d},\max_{mid=u}^{d-1}(\min(f_{v,l,u,mid},f_{v,l,mid+1,d}))\right)$$

- 注意到 $f_{v,l,u,d}$ 也显然具有单调性，因此转移可以使用三分优化，或者直接双指针。
- 复杂度分别是 $O(n^3\log n),O(n^3\log^2 n)$ 的，都能过。

- 给一棵大小为 N 的树 G。
- 考虑在 G 中添加零条或更多条边，得到图 H。
- 求满足以下条件的不同的图 H 的个数，模数为 998244353 。
  1. H 不包含自循环或多条边。
  2. G 和 H 的直径相等。
  3. 对于 H 中的每一对没有直接连接边的顶点对，增加一条直接连接它们的边会减小图的直径。
- $3 \le n \le 2 \times 10^5$

solution

- 首先可以发现 H 的直径是唯一的，否则显然可以加边不减少图的直径。

solution

- 首先可以发现 H 的直径是唯一的，否则显然可以加边不减少图的直径。
- 设这条直径两个端点为 s, t，考虑建出任意一棵以 s 为根的 BFS 树，令 dep_i 表示 i 表示 i 在 BFS 树上的深度，那么两个点两个点 (u, v) 如果在 H 有边必须满足 $|dep_u - dep_v| \leq 1$，且对于满足 $|dep_u - dep_v| \leq 1$ 的点对在 H 中必须有边。

solution

- 首先可以发现 H 的直径是唯一的，否则显然可以加边不减少图的直径。
- 设这条直径两个端点为 s,t，考虑建出任意一棵以 s 为根的 BFS 树，令 dep_i 表示 i 表示 i 在 BFS 树上的深度，那么两个点两个点 (u,v) 如果在 H 有边必须满足 $|dep_u - dep_v| \leq 1$，且对于满足 $|dep_u - dep_v| \leq 1$ 的点对在 H 中必须有边。
- 注意到一个合法的 H 将会对应两种合法的 dep 分配方案，考虑转为计数合法的 dep 分配方案。

solution

- 但是直径可能有很多条，意味着可能有很多 x，但是注意到若直径长度 L 为偶数，则所有直径的中点将为同一个点（设这个中点为 mid），考虑据此稍微改下 dep_i 合法的条件：

solution

- 但是直径可能有很多条，意味着可能有很多 x，但是注意到若直径长度 L 为偶数，则所有直径的中点将为同一个点（设这个中点为 mid），考虑据此稍微改下 dep_i 合法的条件：
  - $dep_{mid} = 0$；

solution

- 但是直径可能有很多条，意味着可能有很多 x，但是注意到若直径长度 L 为偶数，则所有直径的中点将为同一个点（设这个中点为 mid），考虑据此稍微改下 dep_i 合法的条件：
  - $dep_{mid} = 0$；
  - 恰好存在一个点 x 满足 $dep_x = -L/2$；

solution

- 但是直径可能有很多条，意味着可能有很多 x，但是注意到若直径长度 L 为偶数，则所有直径的中点将为同一个点（设这个中点为 mid），考虑据此稍微改下 dep_i 合法的条件：
  - $dep_{mid} = 0$；
  - 恰好存在一个点 x 满足 $dep_x = -L/2$；
  - 恰好存在一个点 y 满足 $dep_y = L/2$；

solution

- 但是直径可能有很多条，意味着可能有很多 x，但是注意到若直径长度 L 为偶数，则所有直径的中点将为同一个点（设这个中点为 mid），考虑据此稍微改下 dep_i 合法的条件：
  - $dep_{mid} = 0$；
  - 恰好存在一个点 x 满足 $dep_x = -L/2$；
  - 恰好存在一个点 y 满足 $dep_y = L/2$；
  - 对于在 G 中有边的点对 (u, v)，满足 $|dep_u - dep_v| \leq 1$。

solution

- 但是直径可能有很多条，意味着可能有很多 x，但是注意到若直径长度 L 为偶数，则所有直径的中点将为同一个点（设这个中点为 mid），考虑据此稍微改下 dep_i 合法的条件：
  - $dep_{mid} = 0$；
  - 恰好存在一个点 x 满足 $dep_x = -L/2$；
  - 恰好存在一个点 y 满足 $dep_y = L/2$；
  - 对于在 G 中有边的点对 (u, v)，满足 $|dep_u - dep_v| \le 1$。
- 由此考虑设计 DP，令 $f_{i,x,y}$ 表示 u 子树内满足 $dep_v - dep_u = -L/2 - dep_u$ 的 v 数量为 x，满足 $dep_v - dep_u = L/2 - dep_u$ 的 v 数量为 y 的方案数。

solution

- 但是直径可能有很多条，意味着可能有很多 x，但是注意到若直径长度 L 为偶数，则所有直径的中点将为同一个点（设这个中点为 mid），考虑据此稍微改下 dep_i 合法的条件：
  - $dep_{mid} = 0$；
  - 恰好存在一个点 x 满足 $dep_x = -L/2$；
  - 恰好存在一个点 y 满足 $dep_y = L/2$；
  - 对于在 G 中有边的点对 (u, v)，满足 $|dep_u - dep_v| \leq 1$。
- 由此考虑设计 DP，令 $f_{i,x,y}$ 表示 u 子树内满足 $dep_v - dep_u = -L/2 - dep_u$ 的 v 数量为 x，满足 $dep_v - dep_u = L/2 - dep_u$ 的 v 数量为 y 的方案数。
- 而 x, y 可以和 2 取 min，转移也只需枚举 (u, v) 的取值（$\{-1, 0, 1\}$），因此复杂度为线性。

solution

- 但是直径可能有很多条，意味着可能有很多 x，但是注意到若直径长度 L 为偶数，则所有直径的中点将为同一个点（设这个中点为 mid），考虑据此稍微改下 dep_i 合法的条件：
    - $dep_{mid} = 0$；
    - 恰好存在一个点 x 满足 $dep_x = -L/2$；
    - 恰好存在一个点 y 满足 $dep_y = L/2$；
    - 对于在 G 中有边的点对 (u, v)，满足 $|dep_u - dep_v| \leq 1$。
- 由此考虑设计 DP，令 $f_{i,x,y}$ 表示 u 子树内满足 $dep_v - dep_u = -L/2 - dep_u$ 的 v 数量为 x，满足 $dep_v - dep_u = L/2 - dep_u$ 的 v 数量为 y 的方案数。
- 而 x, y 可以和 2 取 $\min$，转移也只需枚举 (u, v) 的取值（$\{-1, 0, 1\}$），因此复杂度为线性。
- L 是奇数也是同理，只需要将直径中间的边断开后对两边分别 DP 然后合并。

「PA 2021」Od deski do deski

- 给定 n, m，求满足以下限制的长度为 n 的序列数目：
  1. 每个元素在 $[1, m]$ 之间；
  2. 一次操作定义为删除一个长度至少为 2 且区间两端相等的区间，该序列需要在若干次操作内被删空。
- 答案对 $10^9 + 7$ 取模。
- $1 \leq n \leq 3000$, $1 \leq m \leq 10^9$

solution

- 考虑如何判断一个序列能否被清空。

solution

- 考虑如何判断一个序列能否被清空。
- 这就是一个简单的 DP，令 f_i 表示能否删除 $[1, i]$，l 能更新 r 只需满足 $a_{l+1} = a_r$。

solution

- 考虑如何判断一个序列能否被清空。
- 这就是一个简单的 DP，令 f_i 表示能否删除 $[1,i]$，l 能更新 r 只需满足 $a_{l+1}=a_r$。
- 注意到 f 的转移我们只需要关心哪些颜色已经可以被转移到了，因此可以设计 DP $g_{i,j,0/1}$ 表示考虑了序列的前 i 个数，已经有 j 个颜色可以被更新到了，$f_i=0/1$ 的方案数。

solution

- 考虑如何判断一个序列能否被清空。
- 这就是一个简单的 DP，令 f_i 表示能否删除 $[1, i]$，l 能更新 r 只需满足 $a_{l+1} = a_r$。
- 注意到 f 的转移我们只需要关心哪些颜色已经可以被转移到了，因此可以设计 DP $g_{i,j,0/1}$ 表示考虑了序列的前 i 个数，已经有 j 个颜色可以被更新到了，$f_i = 0/1$ 的方案数。
- 有转移：

solution

- 考虑如何判断一个序列能否被清空。
- 这就是一个简单的 DP，令 f_i 表示能否删除 $[1, i]$，l 能更新 r 只需满足 $a_{l+1} = a_r$。
- 注意到 f 的转移我们只需要关心哪些颜色已经可以被转移到了，因此可以设计 DP $g_{i,j,0/1}$ 表示考虑了序列的前 i 个数，已经有 j 个颜色可以被更新到了，$f_i = 0/1$ 的方案数。
- 有转移：
  - $g_{i,j,0} = g_{i-1,j,0} \times (m - j) + g_{i-1,j-1,1} \times (m - j + 1)$

solution

- 考虑如何判断一个序列能否被清空。
- 这就是一个简单的 DP，令 f_i 表示能否删除 $[1, i]$，l 能更新 r 只需满足 $a_{l+1} = a_r$。
- 注意到 f 的转移我们只需要关心哪些颜色已经可以被转移到了，因此可以设计 DP $g_{i,j,0/1}$ 表示考虑了序列的前 i 个数，已经有 j 个颜色可以被更新到了，$f_i = 0/1$ 的方案数。
- 有转移：
  - $g_{i,j,0} = g_{i-1,j,0} \times (m - j) + g_{i-1,j-1,1} \times (m - j + 1)$
  - $g_{i,j,1} = (g_{i-1,j,0} + g_{i-1,j,1}) \times j$

- 考虑如何判断一个序列能否被清空。
- 这就是一个简单的 DP，令 f_i 表示能否删除 $[1, i]$，l 能更新 r 只需满足 $a_{l+1} = a_r$。
- 注意到 f 的转移我们只需要关心哪些颜色已经可以被转移到了，因此可以设计 DP $g_{i,j,0/1}$ 表示考虑了序列的前 i 个数，已经有 j 个颜色可以被更新到了，$f_i = 0/1$ 的方案数。
- 有转移：
  - $g_{i,j,0} = g_{i-1,j,0} \times (m - j) + g_{i-1,j-1,1} \times (m - j + 1)$
  - $g_{i,j,1} = (g_{i-1,j,0} + g_{i-1,j,1}) \times j$
- 时间复杂度 $O(n^2)$。

「PKUSC 2024」独立

- 给定一棵 n 个节点的树和一个正整数 m，每个节点的权值在 $[1, m]$ 中等概率独立随机。
- 求这棵树的最大独立集的期望乘上 m^n 后的结果。
- 答案对 998244353 取模。
- $1 \leq n \leq 2000, 1 \leq m \leq 10^8$

solution

- 一个直接的想法的是直接设 $f_{u,i,j}, g_{u,i,j}$ 表示表示求最大独立集那个 DP 的 $dp_{u,0} = i, dp_{u,1} = j$ 的方案数和最大独立集之和。

solution

- 一个直接的想法的是直接设 $f_{u,i,j}, g_{u,i,j}$ 表示表示求最大独立集那个 DP 的 $dp_{u,0} = i, dp_{u,1} = j$ 的方案数和最大独立集之和。
- 注意到如果 $dp_{u,1} \le dp_{u,0}$，我们可以直接令 $dp_{u,1} = dp_{u,0}$，因此可以考虑转为直接记录 $dp_{u,1} - dp_{u,0}$，这样 DP 维度就直接降到了两维，且第二维的大小为 m。

solution

- 一个直接的想法的是直接设 $f_{u,i,j}, g_{u,i,j}$ 表示表示求最大独立集那个 DP 的 $dp_{u,0} = i, dp_{u,1} = j$ 的方案数和最大独立集之和。
- 注意到如果 $dp_{u,1} \le dp_{u,0}$，我们可以直接令 $dp_{u,1} = dp_{u,0}$，因此可以考虑转为直接记录 $dp_{u,1} - dp_{u,0}$，这样 DP 维度就直接降到了两维，且第二维的大小为 m。
- 写出具体转移，假设我们想要合并 u, v 的信息，$\forall i, j \in [0, m]$：

solution

- 一个直接的想法的是直接设 $f_{u,i,j}, g_{u,i,j}$ 表示表示求最大独立集那个 DP 的 $dp_{u,0} = i, dp_{u,1} = j$ 的方案数和最大独立集之和。
- 注意到如果 $dp_{u,1} \le dp_{u,0}$，我们可以直接令 $dp_{u,1} = dp_{u,0}$，因此可以考虑转为直接记录 $dp_{u,1} - dp_{u,0}$，这样 DP 维度就直接降到了两维，且第二维的大小为 m。
- 写出具体转移，假设我们想要合并 u, v 的信息，$\forall i, j \in [0, m]$：
  - $f'_{u,k} \leftarrow f'_{u,k} + f_{u,i} \times f_{v,j}$

solution

- 一个直接的想法的是直接设 $f_{u,i,j}, g_{u,i,j}$ 表示表示求最大独立集那个 DP 的 $dp_{u,0} = i, dp_{u,1} = j$ 的方案数和最大独立集之和。
- 注意到如果 $dp_{u,1} \le dp_{u,0}$，我们可以直接令 $dp_{u,1} = dp_{u,0}$，因此可以考虑转为直接记录 $dp_{u,1} - dp_{u,0}$，这样 DP 维度就直接降到了两维，且第二维的大小为 m。
- 写出具体转移，假设我们想要合并 u, v 的信息，$\forall i, j \in [0, m]$：
  - $f'_{u,k} \leftarrow f'_{u,k} + f_{u,i} \times f_{v,j}$
  - $g'_{u,k} \leftarrow g'_{u,k} + f_{u,i} \times (g_{v,j} + j \times f_{v,j}) + f_{v,j} \times g_{u,i}$

- 一个直接的想法的是直接设 $f_{u,i,j}, g_{u,i,j}$ 表示表示求最大独立集那个 DP 的 $dp_{u,0} = i, dp_{u,1} = j$ 的方案数和最大独立集之和。
- 注意到如果 $dp_{u,1} \le dp_{u,0}$，我们可以直接令 $dp_{u,1} = dp_{u,0}$，因此可以考虑转为直接记录 $dp_{u,1} - dp_{u,0}$，这样 DP 维度就直接降到了两维，且第二维的大小为 m。
- 写出具体转移，假设我们想要合并 u, v 的信息，$\forall i, j \in [0, m]$：
  - $f'_{u,k} \leftarrow f'_{u,k} + f_{u,i} \times f_{v,j}$
  - $g'_{u,k} \leftarrow g'_{u,k} + f_{u,i} \times (g_{v,j} + j \times f_{v,j}) + f_{v,j} \times g_{u,i}$
  - 其中 $k = \max(i - j, 0)$

solution

- 一个直接的想法的是直接设 $f_{u,i,j}, g_{u,i,j}$ 表示表示求最大独立集那个 DP 的 $dp_{u,0} = i, dp_{u,1} = j$ 的方案数和最大独立集之和。
- 注意到如果 $dp_{u,1} \le dp_{u,0}$，我们可以直接令 $dp_{u,1} = dp_{u,0}$，因此可以考虑转为直接记录 $dp_{u,1} - dp_{u,0}$，这样 DP 维度就直接降到了两维，且第二维的大小为 m。
- 写出具体转移，假设我们想要合并 u, v 的信息，$\forall i, j \in [0, m]$：
  - $f'_{u,k} \leftarrow f'_{u,k} + f_{u,i} \times f_{v,j}$
  - $g'_{u,k} \leftarrow g'_{u,k} + f_{u,i} \times (g_{v,j} + j \times f_{v,j}) + f_{v,j} \times g_{u,i}$
  - 其中 $k = \max(i - j, 0)$
- 最终答案为 $\sum_{i=0}^{m} g_{rt,i} + f_{rt,i} \times i$。

solution

- 一个直接的想法的是直接设 $f_{u,i,j}, g_{u,i,j}$ 表示表示求最大独立集那个 DP 的 $dp_{u,0} = i, dp_{u,1} = j$ 的方案数和最大独立集之和。
- 注意到如果 $dp_{u,1} \le dp_{u,0}$，我们可以直接令 $dp_{u,1} = dp_{u,0}$，因此可以考虑转为直接记录 $dp_{u,1} - dp_{u,0}$，这样 DP 维度就直接降到了两维，且第二维的大小为 m。
- 写出具体转移，假设我们想要合并 u, v 的信息，$\forall i, j \in [0, m]$：
  - $f'_{u,k} \leftarrow f'_{u,k} + f_{u,i} \times f_{v,j}$
  - $g'_{u,k} \leftarrow g'_{u,k} + f_{u,i} \times (g_{v,j} + j \times f_{v,j}) + f_{v,j} \times g_{u,i}$
  - 其中 $k = \max(i - j, 0)$
- 最终答案为 $\sum_{i=0}^{m} g_{rt,i} + f_{rt,i} \times i$。
- 直接做就可以得到一个 $O(nm^2)$ 的做法。

solution

- 继续分析，我们大胆猜想 $f_{u,x}, g_{u,x}$ 对于每个 u 分别是关于 x 的 siz_u, siz_u+1 次多项式！

solution

- 继续分析，我们大胆猜想 $f_{u,x}, g_{u,x}$ 对于每个 u 分别是关于 x 的 $siz_u, siz_u + 1$ 次多项式！
- 实际上我们可以根据归纳证明这个结论。

solution

- 继续分析，我们大胆猜想 $f_{u,x}, g_{u,x}$ 对于每个 u 分别是关于 x 的 $siz_u, siz_u + 1$ 次多项式！
- 实际上我们可以根据归纳证明这个结论。
- 于是我们可以考虑对于每个 u 只存 $f_{u,0\ldots siz_u}$ 和 $f_{u,m-siz_u+1\ldots m}$，对于每次转移：

solution

- 继续分析，我们大胆猜想 $f_{u,x}, g_{u,x}$ 对于每个 u 分别是关于 x 的 siz_u, siz_u+1 次多项式！
- 实际上我们可以根据归纳证明这个结论。
- 于是我们可以考虑对于每个 u 只存 $f_{u,0\ldots siz_u}$ 和 $f_{u,m-siz_u+1\ldots m}$，对于每次转移：
  - 首先是 $f'_{u,0}, g'_{u,0}$ 需要单独处理，$f'_{u,0}=\sum_{i=0}^{m}\left(f_{u,i}\times\left(\sum_{j=i}^{m}f_{v,j}\right)\right)$，可以考虑先求出这个式子前几个 i 的贡献，然后插值，$g'_{u,0}$ 是同理的。

solution

- 继续分析，我们大胆猜想 $f_{u,x}, g_{u,x}$ 对于每个 u 分别是关于 x 的 $siz_u, siz_u + 1$ 次多项式！
- 实际上我们可以根据归纳证明这个结论。
- 于是我们可以考虑对于每个 u 只存 $f_{u,0\dots siz_u}$ 和 $f_{u,m-siz_u+1\dots m}$，对于每次转移：
  - 首先是 $f'_{u,0}, g'_{u,0}$ 需要单独处理，$f'_{u,0} = \sum_{i=0}^{m}\left(f_{u,i} \times \left(\sum_{j=i}^{m} f_{v,j}\right)\right)$，可以考虑先求出这个式子前几个 i 的贡献，然后插值，$g'_{u,0}$ 是同理的。
  - 然后考虑计算 $f'_{u,i}$ 的最后几位，注意到这部分就是 $f_{u,i}$ 的后几位和 $f_{v,i}$ 的前几位做减法卷积，可以直接 FFT。

solution

- 继续分析，我们大胆猜想 $f_{u,x}, g_{u,x}$ 对于每个 u 分别是关于 x 的 $siz_u, siz_u + 1$ 次多项式！
- 实际上我们可以根据归纳证明这个结论。
- 于是我们可以考虑对于每个 u 只存 $f_{u,0\dots siz_u}$ 和 $f_{u,m-siz_u+1\dots m}$，对于每次转移：
  - 首先是 $f'_{u,0}, g'_{u,0}$ 需要单独处理，$f'_{u,0} = \sum_{i=0}^{m}\left(f_{u,i} \times \left(\sum_{j=i}^{m} f_{v,j}\right)\right)$，可以考虑先求出这个式子前几个 i 的贡献，然后插值，$g'_{u,0}$ 是同理的。
  - 然后考虑计算 $f'_{u,i}$ 的最后几位，注意到这部分就是 $f_{u,i}$ 的后几位和 $f_{v,i}$ 的前几位做减法卷积，可以直接 FFT。
  - 然后就是转移的时候还有最后计算 $f'_{u,i}$ 的前几位，我们都需要快速计算这样一个问题：已知一个多项式在一个区间的上的值，需要快速得到这个多项式在另外一个区间上的值。这是简单的，把插值的式子写出来可以发现就是一个卷积。

solution

- 继续分析，我们大胆猜想 $f_{u,x}, g_{u,x}$ 对于每个 u 分别是关于 x 的 siz_u, siz_u+1 次多项式！
- 实际上我们可以根据归纳证明这个结论。
- 于是我们可以考虑对于每个 u 只存 $f_{u,0\dots siz_u}$ 和 $f_{u,m-siz_u+1\dots m}$，对于每次转移：
    - 首先是 $f'_{u,0}, g'_{u,0}$ 需要单独处理，$f'_{u,0}=\sum_{i=0}^{m}\left(f_{u,i}\times\left(\sum_{j=i}^{m}f_{v,j}\right)\right)$，可以考虑先求出这个式子前几个 i 的贡献，然后插值，$g'_{u,0}$ 是同理的。
    - 然后考虑计算 $f'_{u,i}$ 的最后几位，注意到这部分就是 $f_{u,i}$ 的后几位和 $f_{v,i}$ 的前几位做减法卷积，可以直接 FFT。
    - 然后就是转移的时候还有最后计算 $f'_{u,i}$ 的前几位，我们都需要快速计算这样一个问题：已知一个多项式在一个区间的上的值，需要快速得到这个多项式在另外一个区间上的值。这是简单的，把插值的式子写出来可以发现就是一个卷积。
- 从而一次转移的复杂度至多为 $O(n\log n)$，总复杂度 $O(n^2\log n)$。

Good Luck & Have Fun!