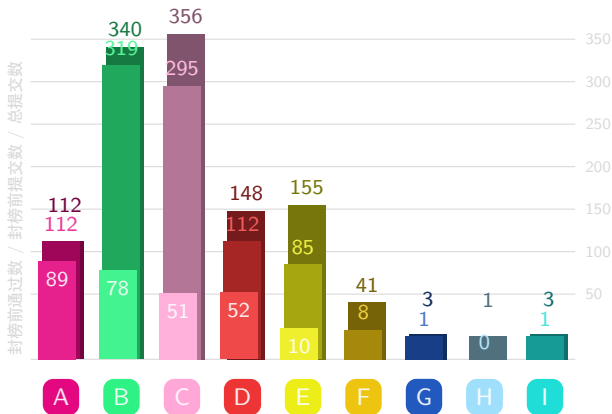


第十八届“快手杯” 北京航空航天大学程序设计竞赛 决赛题目解析

Dec 17, 2023

统计



rnm，退钱！

求给定的 n 条记录的和即可。
时间复杂度 $O(n)$ 。

魔术之尘

贪心地考虑，优先选择合成需要魔术之尘最少的卡组。将 B_i 排序后求前缀和，对于每个 A_i 二分前缀和中不大于 A_i 的最后一个位置。

时间复杂度 $O((n + m) \log m)$ 。

更小但是异或之后至少有 x 个 1

首先要保证 b 尽可能接近 a ，因此我们要让 b 跟 a 高位相同的部分尽可能多。假设 a 的从低到高第 $x-1$ 位（下标从 0 开始）是 1，那么我们可以让 b 的 $x-1$ 以上的位都与 a 相同， $x-1$ 及以下的位都与 a 不同，这样得到的 b 显然就是答案。否则，如果 a 的第 $x-1$ 位是 0，那么如果像上面那样做会导致 b 比 a 大。因此找到 a 的 $x-1$ 位往上的第一个 1，让 b 的这个位是 0，然后 b 下面的位就可以随便填了，在满足题目限制的条件下贪心即可。时间复杂度 $O(n)$ 。

可能需要特判的一些情况和一些细节： x 比 a 的位数大、 x 是 0、 a 是 0、 b 可能需要删去前导零。

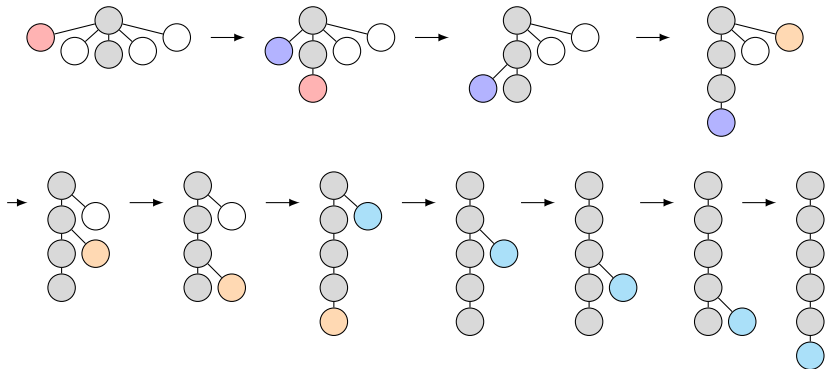
生活在树上

对于一棵 n 个点的树，根到树上所有点距离和最小时树的形态为菊花图，最大时树的形态为链。

以菊花图为初始形态，每次选定非链上的一点不断向链的远端调整，可以使得根到树上所有点距离和连续增加。

时间复杂度 $O(n)$ 。

生活在树上



宝石合成

容易发现，贪心的每次将等级最小的两个或者三个（在连续该等级的宝石只有 3 个时）宝石合成一定不会让结果变差。接下来只需要模拟这个过程即可。可以采用优先队列维护当前每个宝石的等级以及该宝石是由哪个区间的宝石合成出来的。

时间复杂度 $O(n \log n)$ 。

二分查找

注意到每个位置可能的值是一个连续的区间，只需要在此基础上找到一个合法的排列即可。此为经典贪心问题，从小到大遍历每一个值，每次选择当前最小值大于该值且最大值最小的位置放置该值，出现矛盾则说明无解。

时间复杂度 $O(n \log n)$ 。

What if ...?

题目要求的是 A 数组的期望，可以将其转化为计数。 x 数组的所有情况有 n^m 种， A 在每个情况下的数字之和除以情况数量，就是期望。

而在每个情况中 A 数组的数等价于这个分支的进入次数。

What if ...?

首先可以枚举分支 k ，对每个分支计算在所有情况下的进入次数之和。

对于 A_k ，想要在枚举 i 的时候进入这个分支，就需要满足在 k 分支前面的条件全部不成立，而在 k 分支的条件成立。

对于前面的分支， x 是随机的，而比较符号是确定的，我们需要让所有条件都不成立。

What if ...?

可以统计 $=, <, >$ 各出现了几次, 假设分别为 a, b, c 次, 那么所有情况的进入次数之和为:

$$ans_k \times n^m = \sum_{i=1}^n (n-1)^a \times i^b \times (n-i+1)^c \times g(k) \times n^{m-k}$$

其中 $g(k)$ 表示满足当前分支的 x_k 的数量, 由 op_k 决定, $=, <, >$ 分别对应 $1, n-i, i-1$ 。

n^{m-k} 表示后面的分支可以任意选择 x , 不会影响当前分支的状态。

What if ...?

注意当 $k = m + 1$ ，即在 else 分支的时候比较特殊，此时不需要考虑当前和后面的条件，进入次数之和为：

$$ans_k \times n^m = \sum_{i=1}^n (n-1)^a \times i^b \times (n-i+1)^c$$

如果预处理幂次的运算结果，这样我们就得到了一个 $O(nm)$ 的算法，但是不足以通过此题。

What if ...?

不难发现，如果把式子在 $\sum$ 里面的每一项都展开，可以得到一个关于 i 的不超过 m 次的多项式。形如：

$$c_m i^m + c_{m-1} i^{m-1} + \cdots + c_0 i^0$$

和 $\sum$ 结合，再按每一项拆开，就得到了：

$$\begin{aligned} & \sum_{i=1}^n c_m i^m + c_{m-1} i^{m-1} + \cdots + c_0 i^0 \\ &= c_m \sum_{i=1}^n i^m + c_{m-1} \sum_{i=1}^n i^{m-1} + \cdots + c_0 \sum_{i=1}^n i^0 \end{aligned}$$

What if ...?

记 $S_l = \sum_{i=1}^n i^l$, 如果我们能对每个 $0 \leq l \leq m$ 预处理出 S_l , 就可以直接枚举 l 来计算这个式子了。

计算 S_l 有很多种方法, 一种比较简单的可以通过的方法是递推。一种递推公式为:

$$S_l = \frac{n^{l+1}}{l+1} - \frac{1}{l+1} \sum_{i=0}^{l-1} (-1)^{l-i} C_{l+1}^i S_i$$

具体证明不再赘述, 可以通过二项式定理展开。

What if ...?

递推预处理的时间复杂度为 $O(m^2)$ 。还可以使用拉格朗日差值等方法更快速地求出 S_l ，但瓶颈在于枚举计算答案。

计算答案需要枚举分支和多项式次数，所以计算部分复杂度为 $O(m^2)$ ，总复杂度为 $O(m^2)$ 。

Rolling Star

首先可以认为每门已修课程的学分均为 1，接下来只需确定每一学分的成绩。先列出稍后可能用到的 GPA 计算公式：

$$g(s) = 4 - \frac{3(100 - s)^2}{1600} \quad (60 \leq s \leq 100)$$

考虑动态规划。令 $f(c, s, g)$ 表示当前考虑 c 学分，成绩和为 $s = \sum_{i=1}^c s_i$ ， $\frac{1600}{3}$ 倍的 GPA 和为 $g = \sum_{i=1}^c \frac{1600}{3} \cdot g(s_i)$ 时是否存在解。枚举下一学分的成绩 s_{c+1} 即可转移：

$$f(c, s, g) \rightarrow f(c + 1, s + s_{c+1}, g + 1600 \cdot g(s_{c+1}))$$

Rolling Star

注意到 $0 \leq c \leq n$, $60c \leq s \leq 100c$, $\frac{1600}{3}c \leq g \leq \frac{6400}{3}c$, 无法直接保存所有状态。可以先将各维减去偏移量, 使得 $0 \leq s \leq 40c$, $0 \leq g \leq 1600c$ 。但即便如此最多仍有 $48000n^3$ 个状态。

我们有如下结论: 对于任意数列 $a_1, a_2, \dots, a_n$, 存在其排列 $a'_1, a'_2, \dots, a'_n$ 使得:

$$\left| \sum_{i=1}^k a'_i - \frac{k}{n} \sum_{i=1}^n a_i \right| \leq \max_{1 \leq i \leq n} a_i$$

对任意 $1 \leq k \leq n$ 成立。于是我们可以将 g 的有效值控制在均值附近, 使得该维最多有 3200 个状态。

Rolling Star

实际上，由于成绩和与 GPA 和是相互制约的，在动态规划时有效状态数量并不多。利用 Bitset 优化动态规划可以达到 $O(\frac{40^2 \cdot 3200 \cdot n^2}{w})$ 的复杂度，其中 $w = 64$ 。结合 Meet in the middle 可以进一步优化。

请问您今天要来点取模吗？

考虑求 $S(m) = \sum_{k=0}^m (((k \bmod A_1) \bmod A_2) \cdots) \bmod A_n$, 答案即为 $S(R) - S(L-1)$ 。

令 $A_0 = \infty$, 对于 $0 \leq i \leq n$, 按区间维护 $[0, m]$ 对 A_i 取模后的情况。具体来说, 最开始仅有区间 $[0, m]$ 。在对某个 A_1 取模后, $[0, m]$ 先分裂为 $[0, A_1 - 1], [A_1, 2A_1 - 1], \dots, [kA_1, m]$, 在模意义下成为 $\lfloor \frac{n}{A_1} \rfloor$ 个区间 $[0, A_1 - 1]$, 以及区间 $[0, m \bmod A_1]$ 。

请问您今天要来点取模吗？

对于一次取模，一个区间 $[0, r]$ 会成为若干个 $[0, A_i - 1]$ 与 $[0, r \bmod A_i]$ ，其中 $[0, A_i - 1]$ 可以合并，于是一次取模最多增加一个区间。

对于一个区间 $[0, r]$ ，对其有影响的取模最多 $O(\log r)$ 次。
利用数据结构维护区间信息，时间复杂度 $O(n \log^2 R)$ 。