

transport

由于所有点的和为定值，所以我们写出方差的式子 $\frac{\sum (x_i - \bar{x})^2}{n}$ ，其中 $n, \sum x$ 为定值。因此我们希望最小化 $\sum_{i=1}^n (x_i^2 - 2x_i\bar{x} + \bar{x}^2)$ ，其中后两项都是固定的，所以我们要最小化 $\sum_{i=1}^n x_i^2$ 。考虑随意赋一组值然后调整。假设我们给 $x_i + 1, x_j - 1$ ，则对平方和的影响为 $(2x_i + 1) - (2x_j - 1) = 2(x_i - x_j) + 2$ 。因此若存在两个数 $x_i + 1 < x_j$ ，这样调整一定更优，另一方面，若不存在这样的一对数，则不调整最优。所以最优方案一定是一些点上的值为 $\lfloor \frac{s}{n} \rfloor$ ，另一些为这个 $+1$ ，其中 $s = \sum a_i$ 。这也是符合直觉的。

下面考虑怎么求出最小代价。此时记 $v = \lfloor \frac{s}{n} \rfloor, m = s - nv$ ，则我们需要在 m 个点上放 $v + 1$ ，在 $n - m$ 个点上放 v 。考虑如果确定了一种方案怎么计算其代价。我们对于每条边分别计算，若一条边某一侧的子树中原来的权值之和与新的权值之和的差为 d ，则这条边至少会带来 dw 的贡献。而可以从下往上贪心来达到这个理论最小值。所以我们希望最小化 $\sum dw$ 。

这样就很好做了，记 $dp_{i,j}$ 表示以 i 为根的子树中有 j 个点的权值为 $v + 1$ 时子树内的最小代价。处理出子树大小以及子树中初始权值的和就可以转移。转移时枚举原来选了 j 个 $v + 1$ ，新子树中选了 k 个 $v + 1$ 可以 $O(1)$ 计算贡献。这是经典的树上背包，复杂度 $O(n^2)$ 。

or

先考虑去掉 l 的限制，即先考虑 $l = 0$ 怎么做。此时找到 r 的最高位，假设这是第 k 位。那么首先任意 $0 \leq v < 2^k$ 都是可以取到的，因为 $v < r$ ，所以我们只需要考虑第 k 位为 1 的数有哪些能被取到。此时我们发现，我们首先选择 2^k 这个数将第 k 位变成 1，这一定是可以的，因为 $r \geq 2^k$ 。下面我们发现，对于任意的 $2^k + v (0 \leq v < 2^k)$ ，由于 $v < r$ ，所以直接选择对应的 v 就可以凑出 $2^k + v$ 。这样我们证明了此时可以取到任意的 $0 \leq v < 2^{k+1}$ 。

下面回到原题，首先 l, r 最高的若干位完全相同的可以直接忽略，因为它们全部都是一样的，所以任意的取法得到的结果在这些位上也都是一样的。假设 h 为最高的 l, r 不同的位，我们分两种结果讨论。若结果的第 h 位为 0，则任意 $l \leq v < 2^h$ 都是可以的，而且显然只有这些是可以的。下面主要分析结果的第 h 位为 1 的情形。

我们同样先选择 2^h 这个数将这一位变成 1，下面我们只需要考虑更低的位。此时问题复杂的原因是 l 带来一段可以选择的后缀， r 带来一段可以选择的前缀。这个问题其实可以直接设计出 dp 进行转移，但这样做不够简洁。我们考虑挖掘一些更有用的性质。首先对于任意 $l \leq v < 2^h$ ， $2^h + v$ 一定可以被凑出来，直接选择 v 即可。那么现在我们只需要考虑 $v < l$ 的部分。由于按位或有一个很好的性质就是 $a \text{ or } b \geq \max(a, b)$ ，所以若我们希望最终两个数或起来 $< l$ ，这两个数都必须 $< l$ 。这意味着我们现在需要考虑的部分只能是由若干 $2^h \leq v \leq r$ 的 v 或起来得到的。由于我们前面已经去掉了最高位，这个问题就形如最开始我们考虑的 $l = 0$ 的问题，是很容易解决的。这样我们可以算出仅使用 $\geq 2^h$ 的数或起来能得到的区间，将这两个区间取一个并即是答案。复杂度 $O(\log n)$ 。

number

枚举 gcd，假设为 g 。直接计算一些数的 $\gcd = g$ 的 lcm 之积这样的问题是很困难的，需要进行复杂的容斥。但我们考虑能不能把容斥放在前面。注意到若我们只要求所有数都是 g 的倍数而不强制它们的 $\gcd = g$ ，这个问题会简单很多。先假设这是好做的，即我们设 $f(d) = \prod_{\forall i, d|x_i} \text{lcm}(x_i)$ ，以及 $g(d) = \prod_{d=\gcd(x_i)} \text{lcm}(x_i)$ ，则答案为 $\prod_d g(d)^d$ ，所以我们需要算出所有 $g(d)$ 。而 $f(d) = \prod_{d|n} g(n)$ ，这看起来是一个莫比乌斯反演的变式，只不过莫比乌斯反演用来计算和，而我们需要计算的是积。但我们容易发现求得了 f 之后要求 g ，并不一定需要局限于使用莫比乌斯反演的式子。考虑从大往小枚

举 d ，假设我们已经确定了所有 $n > d$ 的 $g(n)$ 。那么我们可以将式子写成 $g(d) = \frac{f(d)}{\prod_{d|n, n \neq d} g(n)}$ ，由于 $f(d)$ 的这些 lcm 显然都不是 998244353 的倍数，所以所有的 $g(d)$ 也显然不是 998244353 的倍数，因此直接取倒数是没有问题的。这样我们就在 $O(m \log m)$ 的时间内由 f 推到了 g 。下面我们只需要能够求出所有的 $f(d)$ 即可。

考虑这个式子 $\prod_{\forall i, d|x_i} \text{lcm}(x_i)$ ，由于是要求乘积，所以可以对每个质因子分开考虑。先将所有排列都具有的因数 d 计入答案，我们只需要计算总排列个数，即 $\lfloor \frac{m}{d} \rfloor^n$ 。下面我们将所有这样的排列中的 x_i 除掉 d ，记 $lim = \lfloor \frac{m}{d} \rfloor$ ，则现在要求所有 $x_i \in [1, lim]$ ，其它没有限制。接着我们考虑所有质因子的贡献。先将式子改写成 $\prod_x \prod_{p,c} (p^c)^{t(p^c, x)}$ ，其中 $t(p^c, x) = [\text{lcm}(x_i) | p^c, \text{lcm}(x_i) \nmid p^{c-1}]$ 。将外面的乘积放到指数上变成求和，则我们要算 $\prod_{p,c} (p^c)^{s(p^c)}$ ，其中 $s(p^c) = \sum_x t(p^c, x)$ 。若我们记 $s'(p^c) = \sum_x t'(p^c, x)$ ，其中 $t'(p^c, x) = [\text{lcm}(x_i) | p^c]$ ，则 $s(p^c) = s'(p^c) - s'(p^{c-1})$ 。而 s' 的计算是容易的，具体来说，lcm 是所有数之因此指数的 max，所以 $\text{lcm} | p^c$ 相当于 $\forall i, x_i | p^c$ 。一个位置上这样的数的个数是 $lim - \lfloor \frac{lim}{p^c} \rfloor$ ，所以总方案数为 $(lim - \lfloor \frac{lim}{p^c} \rfloor)^n$ 。这样我们枚举 p, c ，求出 $s'(p^c)$ ，推回 $s(p^c)$ 即可求出最终的答案。

分析一下复杂度，枚举 g 之后 $lim = \frac{m}{g}$ ，所以 lim 的总和为 $O(m \log m)$ ，而我们枚举 p, c 都是 $O(lim)$ 级别，其中还要带上计算快速幂的复杂度，总复杂度 $O(m \log m \log n)$ 。可以对相同的 lim 做记忆化，因为后面的部分和 d 没有关系，可以优化一些常数。

最后需要注意对指数的计算，即 s, s' 都是对 $mod - 1$ 取模，但由于这里只需要支持加减乘法，所以模数不是质数是没有关系的。

graph

若按照最短路将这些点分层，则每一层中的点的编号都是连续的。因此容易想到记 $dp_{i,j,k}$ 表示考虑到前 i 个点，最后一层有 j 个点剩余度数为 1，有 k 个点剩余度数为 2。这个 dp 的转移至少也需要 $O(n)$ 的复杂度，这样一来就至少是 $O(n^4)$ ，所以首先需要优化状态。

i 是必不可少的，向后的度数也是必不可少的，但为什么我们要记录两个状态来表示度数呢？因为后面转移的时候，点可以任意向前连边，但那 k 个度数为 2 的点内部没有顺序，所以要除掉 2^k 。但这件事可以在上一步转移的时候将系数考虑进去。具体来说，我们新加一层的时候是加入了一段区间，每个点先去掉一个度数，然后其中有 x 个点度数为 1， y 个点度数为 2。在这内部连边之后，最终会剩余 j 个点度数为 1， k 个点度数为 2，记这个方案数为 $f_{x,y,j,k}$ 。在这个过程中，下一步的转移需要乘上 2^{-k} 的系数，所以我们需要单独维护 j, k 。但我们考虑记 $f_{x,y,s}$ ，表示若考虑所有原来 $j + 2k = s$ 的状态， $f_{x,y,j,k} \times 2^{-k}$ 的和是多少。这样我们在原来的 dp 中可以只记录 $dp_{i,s}$ 表示考虑前 i 个点，向后的度数之和为 s 时，前面的方案数乘上转移系数之和是多少。

考虑转移 $f_{x,y,s}$ 。若 $x \neq 0$ ，则考虑第一个度数为 1 的点和哪个点连边了。否则考虑第一个度数为 2 的点和哪些点连边了。在令一个度数为 2 的点不连边的时候乘上 $\frac{1}{2}$ 的系数即可。讨论之后整体的转移是容易 $O(1)$ 完成的，需要注意度数为 2 的点要一下转移一整个，不能一条一条出边转移，否则无法处理重边。这样我们得到了一个 $O(n^3)$ 的做法，可以通过 300，但这个做法是空间复杂度为 $O(n^3)$ ，无法通过 1000。

为了优化掉空间，考虑和刚刚类似的分段转移的方法，即我们在 f 中也不记录度数为 1 和 2 的点分别的个数，而是先考虑度数为 2 的点的连边，再考虑度数为 1 的点的连边。具体来说，我们先将所有度数为 2 的点看作两个点，然后在这些点直接随意两两配对，最后剩下一些点。但这个配对的过程可能会导致不合法情况的出现，所以我们考虑使用容斥解决这些不合法情况。不合法情况有两种，自环和重边，且只出现在度数为 2 的点上（自环就是自己拆出的两个点之间连边，重边就是字面意思）。我们钦定有

k 个度数为 2 的点不合法, 则枚举其中 l 个连成自环, 剩下 $k - l$ 个连成重边, 其方案数可以通过组合数计算得到。这个可以 $O(n^2)$ 预处理一个 c_k , 表示有 k 个点不合法的容斥系数乘以方案数之和。下面在转移的时候, 设当前分别有 c_1, c_2 个度数为 1, 2 的点, 枚举不合法点个数 k , 则剩余还有 $c_1 + 2(c_2 - k)$ 个点, 枚举最终剩余的度数 d , 则剩余的 $c_1 + 2(c_2 - k) - d$ 个点直接任意连即可, 这个方案数也可以预处理出来。直接实现复杂度是 $O(n^4)$, 还需要优化。后面枚举 k, d 都是不可省略的, 但可以发现这个部分和选取的段中的具体情况无关, 如果我们要转移到 $f_{j,d}$ 的话, 只需要对所有 i, k , 在 $c_1 + 2(c_2 - k)$ 的位置处加上其系数, 最终枚举 d 可以在 i 之外枚举。这样我们先枚举 i, j, k , 然后再枚举 j, k, d 做两步转移就可以将复杂度优化至 $O(n^3)$ 。