计数问题选讲

Harry27182

2025 年 1 月 27 日

- 给定 n, k，问有多少个序列组 $(A_0, A_1, ...A_n)$ 满足：序列 A_i 的长度为 i；所有元素都在 $[1, k]$ 中，A_i 是 A_{i+1} 的子序列且 A_i 的字典序小于 A_{i+1}。
- $n, k \leq 300$。

- 显然 A_i 是在 A_{i-1} 的基础上插入一个数得到的。所以问题就转化为了求符合条件的长度为 n 的序列数量。

- 显然 A_i 是在 A_{i-1} 的基础上插入一个数得到的。所以问题就转化为了求符合条件的长度为 n 的序列数量。
- 假设 s_i 是最后一个插入序列的元素，那么问题可以等价于 s_i 需要大于 i 之后第一个 $\neq s_i$ 的元素。以此可以归纳类推。

- 显然 A_i 是在 A_{i-1} 的基础上插入一个数得到的。所以问题就转化为了求符合条件的长度为 n 的序列数量。
- 假设 s_i 是最后一个插入序列的元素，那么问题可以等价于 s_i 需要大于 i 之后第一个 $\neq s_i$ 的元素。以此可以归纳类推。
- 但是，如果有一段相同的元素，那么它们的删除顺序不同会得到同样的结果，而这是我们不希望看到的。如果有一段相同的元素，钦定需要先删除后面的即可。

- 设 $dp_{i,j}$ 表示当前长度为 i，用了 $[1,j]$ 的元素的方案数。考虑找到第一个填 1 的位置进行转移。

- 设 $dp_{i,j}$ 表示当前长度为 i，用了 $[1,j]$ 的元素的方案数。考虑找到第一个填 1 的位置进行转移。
- 如果不存在这种位置，那么有 $dp_{i,j} = dp_{i,j-1}$。

- 设 $dp_{i,j}$ 表示当前长度为 i，用了 $[1,j]$ 的元素的方案数。考虑找到第一个填 1 的位置进行转移。
- 如果不存在这种位置，那么有 $dp_{i,j} = dp_{i,j-1}$。
- 否则找到这个位置 k 并枚举其填入时间 p，那么要求 k 以后的都必须在 p 以后被加入。有 $dp_{i,j} = \sum_k \sum_p dp_{k-1,j-1} dp_{i-k,j} \binom{i-p}{i-k}$，复杂度 $O(n^4)$

- 设 $dp_{i,j}$ 表示当前长度为 i，用了 $[1,j]$ 的元素的方案数。考虑找到第一个填 1 的位置进行转移。
- 如果不存在这种位置，那么有 $dp_{i,j} = dp_{i,j-1}$。
- 否则找到这个位置 k 并枚举其填入时间 p，那么要求 k 以后的都必须在 p 以后被加入。有 $dp_{i,j} = \sum_k \sum_p dp_{k-1,j-1} dp_{i-k,j} \binom{i-p}{i-k}$，复杂度 $O(n^4)$
- 发现组合数与 j 无关，可以预处理，复杂度 $O(n^3)$。

- 给定一个长度为 n 的棋盘，第 i 列只有从下到上前 h_i 个位置。
- 求有多少种放置象棋中车的方案，使得所有位置均能被车覆盖。能被覆盖指在同一行或同一列连通。
- $n \leq 400(4000), h_i \leq n$。

- 每次找到最小的位置，然后下面的一块处理之后劈成两半，这两半就是独立的了。所以说可以套路的考虑笛卡尔树。

- 每次找到最小的位置，然后下面的一块处理之后劈成两半，这两半就是独立的了。所以说可以套路的考虑笛卡尔树。
- 必须全部覆盖这种问题需要套路地考虑容斥，设 $dp_{u,i}$ 表示当前在笛卡尔树上考虑 u 子树，其中有 i 列是可以有棋子的，其他列是钦定不能有棋子的对应的方案数。

- 每次找到最小的位置，然后下面的一块处理之后劈成两半，这两半就是独立的了。所以说可以套路的考虑笛卡尔树。
- 必须全部覆盖这种问题需要套路地考虑容斥，设 $dp_{u,i}$ 表示当前在笛卡尔树上考虑 u 子树，其中有 i 列是可以有棋子的，其他列是钦定不能有棋子的对应的方案数。
- 转移显然可以树形背包，对于当前新加入的行，对应的方案数需要乘一个系数，考虑当前是否存在若干列被钦定不能有棋子，如果存在那么转移系数为 2^i-1，本行必须填入棋子。否则系数为 2^i。复杂度 $O(n^2)$。

- 上述做法真的正确吗？

- 上述做法真的正确吗？
- 显然不正确，因为这个转移中是默认了没有被钦定的列是有棋子的，但我们做不到这一点。

- 上述做法真的正确吗？
- 显然不正确，因为这个转移中是默认了没有被钦定的列是有棋子的，但我们做不到这一点。
- 我们再进行一次容斥，在未被第一次钦定不能有棋子的列中再钦定一些列没有棋子。或者说，第一次选择若干列有棋子，其他列没有棋子。第二次在选择有棋子的列中钦定若干没有棋子。

- 上述做法真的正确吗？
- 显然不正确，因为这个转移中是默认了没有被钦定的列是有棋子的，但我们做不到这一点。
- 我们再进行一次容斥，在未被第一次钦定不能有棋子的列中再钦定一些列没有棋子。或者说，第一次选择若干列有棋子，其他列没有棋子。第二次在选择有棋子的列中钦定若干没有棋子。
- 所以需要记录 $dp_{u,i,j}$ 表示有 i 列有棋子，其中 j 列容斥钦定没有棋子的方案数。需要对 i,j 两维分别背包，复杂度 $O(n^4)$。

- 上述做法真的正确吗？
- 显然不正确，因为这个转移中是默认了没有被钦定的列是有棋子的，但我们做不到这一点。
- 我们再进行一次容斥，在未被第一次钦定不能有棋子的列中再钦定一些列没有棋子。或者说，第一次选择若干列有棋子，其他列没有棋子。第二次在选择有棋子的列中钦定若干没有棋子。
- 所以需要记录 $dp_{u,i,j}$ 表示有 i 列有棋子，其中 j 列容斥钦定没有棋子的方案数。需要对 i,j 两维分别背包，复杂度 $O(n^4)$。
- 发现转移系数只和 $i-j$ 有关，所以可以少记录一维，复杂度 $O(n^2)$，可以通过。

ARC093F

- 给定 2^n 个人，按照满二叉树的形态进行淘汰赛。
- 你是 1 号，存在 m 个人打得过你，剩下人都打不过你。其他人之间的比赛编号小的胜利。
- 问有多少种方案你最后能赢。
- $n, m \leq 16$。

- 考虑容斥，钦定 1 号点的若干场比赛输掉，剩下的不做限制。

- 考虑容斥，钦定 1 号点的若干场比赛输掉，剩下的不做限制。
- 令 $f(S,T)$ 表示深度为 S 集合内的场次钦定输掉，输的是 T 集合内的人的方案数。

- 考虑容斥，钦定 1 号点的若干场比赛输掉，剩下的不做限制。
- 令 $f(S,T)$ 表示深度为 S 集合内的场次钦定输掉，输的是 T 集合内的人的方案数。
- $f(S,T)$ 是可以 dp 简单算出的，按照 T 集合内编号从大到小的顺序放入，可以确定目前能选入当前子树被新元素打败的人数，组合数计算方案数即可。

- 考虑容斥，钦定 1 号点的若干场比赛输掉，剩下的不做限制。
- 令 $f(S,T)$ 表示深度为 S 集合内的场次钦定输掉，输的是 T 集合内的人的方案数。
- $f(S,T)$ 是可以 dp 简单算出的，按照 T 集合内编号从大到小的顺序放入，可以确定目前能选入当前子树被新元素打败的人数，组合数计算方案数即可。
- 复杂度 $O(2^{nm}(n+m))$。

- 贡献是在 T 集合的每个元素之间独立的。并且在计算 T 集合中第 i 个元素时，并不关心前 $i-1$ 个元素，只需要关心他们的最小值。

- 贡献是在 T 集合的每个元素之间独立的。并且在计算 T 集合中第 i 个元素时，并不关心前 $i-1$ 个元素，只需要关心他们的最小值。
- 所以可以优化 dp 了，设 $dp_{i,S}$ 表示考虑了前 i 个人，当前选了的深度集合为 S 的方案数。复杂度 $O(2^n nm)$。

- 定义一个长度为 n，有 k 种颜色的序列是好的，当且仅当存在一个长度为 k 的子区间包含全部 k 种颜色。
- 一个好的序列的权值定义为一个长度为 m 的序列 A 在原序列中的出现次数。求所有好的序列的权值和。
- $n \leq 25000, k \leq 400$。

- 首先可以考虑用不考虑限制条件的答案，容斥掉不合法的部分。前半部分显然是 $k^{n-m}(n-m+1)$。对后者分类讨论。

- 首先可以考虑用不考虑限制条件的答案，容斥掉不合法的部分。前半部分显然是 $k^{n-m}(n-m+1)$。对后者分类讨论。
- 如果 $m \geq k$，并且 A 序列中存在合法部分。显然不需要容斥。

- 首先可以考虑用不考虑限制条件的答案，容斥掉不合法的部分。前半部分显然是 $k^{n-m}(n-m+1)$。对后者分类讨论。
- 如果 $m \geq k$，并且 A 序列中存在合法部分。显然不需要容斥。
- 如果 $m < k$，且 A 序列两两不同。那么可以发现，A 序列长什么样子是不重要的。只需要统计所有不合法串中长度为 m 的互不相同子串数量即可。

- 首先可以考虑用不考虑限制条件的答案，容斥掉不合法的部分。前半部分显然是 $k^{n-m}(n-m+1)$。对后者分类讨论。
- 如果 $m \geq k$，并且 A 序列中存在合法部分。显然不需要容斥。
- 如果 $m < k$，且 A 序列两两不同。那么可以发现，A 序列长什么样子是不重要的。只需要统计所有不合法串中长度为 m 的互不相同子串数量即可。
- 设 $f_{i,j}$ 表示长度为 i，最长合法后缀长度为 j 的方案数，$g_{i,j}$ 表示对应的权值和。转移显然。复杂度 $O(nk)$。

- 对于剩下的情况，A 中一定有重复元素。考虑枚举一个 A 出现的位置进行统计贡献。

- 对于剩下的情况，A 中一定有重复元素。考虑枚举一个 A 出现的位置进行统计贡献。
- 条件转化为 A 序列两边不出现长度为 k 的合法区间。所以可以进行类似的 dp，合并的时候卷积卷起来即可。复杂度 $O(nk)$。

qoj5367

- 给定一棵 n 个点的树，求有多少排列 p 满足 $dep(lca(p_i, p_{i+1}) \leq dep(lca(p_{i+1}, p_{i+2}))$。
- $n \leq 80$。

- 问题是满足可递归性的，首先操作 1 的不同子树，直到只有一个子树内有点，递归到这个子树解决。

- 问题是满足可递归性的，首先操作 1 的不同子树，直到只有一个子树内有点，递归到这个子树解决。
- 设 $dp_{u,i}$ 表示 u 子树内还剩下 i 个未确定标号的点的方案数。考虑转移。

- 问题是满足可递归性的，首先操作 1 的不同子树，直到只有一个子树内有点，递归到这个子树解决。
- 设 $dp_{u,i}$ 表示 u 子树内还剩下 i 个未确定标号的点的方案数。考虑转移。
- 选择一个子树 v 从 $dp_{v,j}$ 转移过来。转移的时候对于其他子树，其内部元素可以看做相同，且相邻两个不能选择相同子树。

- 问题是满足可递归性的，首先操作 1 的不同子树，直到只有一个子树内有点，递归到这个子树解决。
- 设 $dp_{u,i}$ 表示 u 子树内还剩下 i 个未确定标号的点的方案数。考虑转移。
- 选择一个子树 v 从 $dp_{v,j}$ 转移过来。转移的时候对于其他子树，其内部元素可以看做相同，且相邻两个不能选择相同子树。
- 考虑容斥，处理 $f_{i,j,k}$ 表示考虑了前 i 个子树，选了 j 个节点，钦定 k 个相邻相同的方案数。转移背包，系数是组合数。

- 问题是满足可递归性的，首先操作 1 的不同子树，直到只有一个子树内有点，递归到这个子树解决。
- 设 $dp_{u,i}$ 表示 u 子树内还剩下 i 个未确定标号的点的方案数。考虑转移。
- 选择一个子树 v 从 $dp_{v,j}$ 转移过来。转移的时候对于其他子树，其内部元素可以看做相同，且相邻两个不能选择相同子树。
- 考虑容斥，处理 $f_{i,j,k}$ 表示考虑了前 i 个子树，选了 j 个节点，钦定 k 个相邻相同的方案数。转移背包，系数是组合数。
- 处理好 f 数组之后再对当前子树进行处理，转移类似。最后将 f 数组的值转移进 dp 数组。复杂度 $O(n^5)$。

- 定义 $f(S) = \max_{T \subset S, |T|=m} \prod_{x \in T} x$。给定参数 m 和集合 S，求 $\sum_{T \subset S, |T| \geq m} f(T)$。位置不同的两个相等元素看做不同。
- $|S| \leq 600$。

- 首先考虑对于一个确定的集合 S 如何计算。在认为集合中没有 0 的情况下，发现是选择绝对值最大的前 m 个数的，根据正负需要进行一点调整。

- 首先考虑对于一个确定的集合 S 如何计算。在认为集合中没有 0 的情况下，发现是选择绝对值最大的前 m 个数的，根据正负需要进行一点调整。
- 考虑对于每个 $|T| = m$ 的集合 T，计算其对答案的贡献。对于一个 T 能否做出贡献，显然只需要关心绝对值最小的正数和负数即可。

- 首先考虑对于一个确定的集合 S 如何计算。在认为集合中没有 0 的情况下，发现是选择绝对值最大的前 m 个数的，根据正负需要进行一点调整。
- 考虑对于每个 $|T| = m$ 的集合 T，计算其对答案的贡献。对于一个 T 能否做出贡献，显然只需要关心绝对值最小的正数和负数即可。
- 令 $dp_{i,j,k,0/1}$ 表示绝对值从大到小排序后第 i 个选进去了，选了 j 个，上一个与 i 符号不同的是 k，选了奇数个还是偶数个负数的所有方案对应的绝对值乘积之和。

- 首先考虑对于一个确定的集合 S 如何计算。在认为集合中没有 0 的情况下，发现是选择绝对值最大的前 m 个数的，根据正负需要进行一点调整。
- 考虑对于每个 $|T| = m$ 的集合 T，计算其对答案的贡献。对于一个 T 能否做出贡献，显然只需要关心绝对值最小的正数和负数即可。
- 令 $dp_{i,j,k,0/1}$ 表示绝对值从大到小排序后第 i 个选进去了，选了 j 个，上一个与 i 符号不同的是 k，选了奇数个还是偶数个负数的所有方案对应的绝对值乘积之和。
- 前缀和优化转移可以做到 $O(n^3)$。

- 对于 $dp_{i,m,k,0}$ 的贡献系数，显然就是 2^{n-i}。下面考虑 $dp_{i,m,k,1}$ 的情况。

- 对于 $dp_{i,m,k,0}$ 的贡献系数，显然就是 2^{n-i}。下面考虑 $dp_{i,m,k,1}$ 的情况。
- 如果后面都不选，就不得不将负数计入答案，特殊处理一下、

- 对于 $dp_{i,m,k,0}$ 的贡献系数，显然就是 2^{n-i}。下面考虑 $dp_{i,m,k,1}$ 的情况。
- 如果后面都不选，就不得不将负数计入答案，特殊处理一下、
- 考虑其他选进去的正负是否都有，如果只有一个有是好算的，否则要考虑哪个更优，容易发现这个东西对于每一个最大的正数位置有单调性，所以可以双指针扫一遍，去求出每一个做出贡献的情况的范围。单次 $O(n)$，总复杂度 $O(n^3)$。

- 对于 $dp_{i,m,k,0}$ 的贡献系数，显然就是 2^{n-i}。下面考虑 $dp_{i,m,k,1}$ 的情况。
- 如果后面都不选，就不得不将负数计入答案，特殊处理一下、
- 考虑其他选进去的正负是否都有，如果只有一个有是好算的，否则要考虑哪个更优，容易发现这个东西对于每一个最大的正数位置有单调性，所以可以双指针扫一遍，去求出每一个做出贡献的情况的范围。单次 $O(n)$，总复杂度 $O(n^3)$。
- 最后注意一个 corner case，选择数量比 m 多但仍然是负数，这里只有 m 是奇数且全选负数有可能达到，额外处理一下即可。

- 有一个长度为 n，值域在 $[1, c]$ 的序列 a，设长度为 m 的序列 b 满足 $b_i = \min_{j=l_i}^{r_i} a_j$。求 a 任取可以得到多少种不同的 b。
- $n \leq 100, c \leq 10^8$。

- 发现很难对 b 序列进行计数，考虑在 a 序列与 b 序列间建立一种一一对应的对应关系，然后对满足能够对应一种 b 序列的 a 序列计数。

- 发现很难对 b 序列进行计数，考虑在 a 序列与 b 序列间建立一种一一对应的对应关系，然后对满足能够对应一种 b 序列的 a 序列计数。
- 首先考虑对于 $c = 2$ 的情况，我们钦定能填 2 的位置就不填 1。那么一段区间能填 2，当且仅当被这段区间包含的所有 $[l_i, r_i]$ 的并能覆盖整个区间。所以可以简单 dp。

- 发现很难对 b 序列进行计数，考虑在 a 序列与 b 序列间建立一种一一对应的对应关系，然后对满足能够对应一种 b 序列的 a 序列计数。
- 首先考虑对于 $c = 2$ 的情况，我们钦定能填 2 的位置就不填 1。那么一段区间能填 2，当且仅当被这段区间包含的所有 $[l_i, r_i]$ 的并能覆盖整个区间。所以可以简单 dp。
- 以此类推，我们对于每个 b 序列，对每个位置尝试最大化 a。考虑从大到小对于每个 x 计算，设 $dp_{x,l,r}$ 表示用 $[x, c]$ 填了 $[l, r]$，转移枚举第一个填 x 的位置，注意判断转移条件。复杂度 $O(n^3 c)$。

- 发现很难对 b 序列进行计数，考虑在 a 序列与 b 序列间建立一种一一对应的对应关系，然后对满足能够对应一种 b 序列的 a 序列计数。
- 首先考虑对于 $c = 2$ 的情况，我们钦定能填 2 的位置就不填 1。那么一段区间能填 2，当且仅当被这段区间包含的所有 $[l_i, r_i]$ 的并能覆盖整个区间。所以可以简单 dp。
- 以此类推，我们对于每个 b 序列，对每个位置尝试最大化 a。考虑从大到小对于每个 x 计算，设 $dp_{x,l,r}$ 表示用 $[x, c]$ 填了 $[l, r]$，转移枚举第一个填 x 的位置，注意判断转移条件。复杂度 $O(n^3c)$。
- 答案是关于 c 的 n 次多项式，拉格朗日插值即可做到 $O(n^4)$。

- 有 nm 本书，第 i 本书权值为 a_i。他想把这些书分成若干无序的组，每组数量都是 m 的倍数。一个分组方案的贡献为所有组权值和的乘积。求所有方案的贡献和。
- $n \leq 1500, m \leq 100$。

- 显然要把贡献用乘法分配律拆出来。设一个分了 b 组的贡献系数为 $w(b)$。$w(b)$ 是将 $nm-b$ 个有标号数分成 b 组，每组元素个数都是 m 的倍数 -1 的方案数。

- 显然要把贡献用乘法分配律拆出来。设一个分了 b 组的贡献系数为 $w(b)$。$w(b)$ 是将 $nm-b$ 个有标号数分成 b 组，每组元素个数都是 m 的倍数 -1 的方案数。
- 转化为将 nm 个数分成 b 组，每组有一个元素无标号，其他有标号的方案数。令 $f_{i,j}$ 表示前 i 个集合，选了 jm 个数的方案数。转移 $f_{i,j}=f_{i-1,k}\frac{1}{((j-k)m-1)!}$。复杂度 $O(n^3)$。

- 处理完 $w(b)$ 之后可以进行 dp，令 $dp_{i,j}$ 表示前 i 个，选了 j 个的乘积之和，容易 $O(n^2)$ dp。总复杂度 $O(n^3)$。

- 处理完 $w(b)$ 之后可以进行 dp，令 $dp_{i,j}$ 表示前 i 个，选了 j 个的乘积之和，容易 $O(n^2)$ dp。总复杂度 $O(n^3)$。
- 考虑优化预处理的 dp，设阈值 B，对于集合大小 $\leq Bm$ 和 $> Bm$ 分别处理。前者 $k \leq B$，后者 $i \leq \frac{n}{B}$。合并的时候做一个卷积即可。取 $B = \sqrt{n}$ 得到最优复杂度 $O(n^2\sqrt{n})$。

- 把 NOI2023 书本的消耗体力变成第 i 摞书的磨损值加上两摞书的重量之和。
- $\sum n^2 \leq 10^8$。

- 用二叉树刻画合并过程，发现重量部分的贡献可以用 $\sum_i dep_i w_i$ 来刻画。所以如果确定了所有的 dep_i，那么按照 dep 从小到大匹配 w 从大到小显然最优。所以只需要考虑树的形态。

- 用二叉树刻画合并过程，发现重量部分的贡献可以用 $\sum_i dep_i w_i$ 来刻画。所以如果确定了所有的 dep_i，那么按照 dep 从小到大匹配 w 从大到小显然最优。所以只需要考虑树的形态。
- 考虑磨损值的贡献，可以看做对于二叉树进行长链剖分后，除了根节点所在长链其他链的 $2^l en - 1$ 之和。

- 用二叉树刻画合并过程，发现重量部分的贡献可以用 $\sum_i dep_i w_i$ 来刻画。所以如果确定了所有的 dep_i，那么按照 dep 从小到大匹配 w 从大到小显然最优。所以只需要考虑树的形态。
- 考虑磨损值的贡献，可以看做对于二叉树进行长链剖分后，除了根节点所在长链其他链的 $2^l en - 1$ 之和。
- 如果我们在某一层确定了若干个 $d_1 \sim d_k$ 表示每个子树的深度，考虑如何匹配。匹配的作用是截停一条长链，所以截停的越长越好，也就是按照 d 从大到小排序后相邻两两匹配。

- 考虑如何计算贡献。容易发现，对于第 i 层的第 x 个叶子节点，它的贡献是 $2^{lowbit(x)}$。这样就可以形式化刻画贡献。

- 考虑如何计算贡献。容易发现，对于第 i 层的第 x 个叶子节点，它的贡献是 $2^{lowbit(x)}$。这样就可以形式化刻画贡献。
- 有一个暴力的 dp 是设 $dp_{i,j,k}$ 表示从上到下第 i 层，当前层 j 个节点，k 个叶子的答案。复杂度 $O(n^4)$。

- 考虑如何计算贡献。容易发现，对于第 i 层的第 x 个叶子节点，它的贡献是 $2^{lowbit(x)}$。这样就可以形式化刻画贡献。
- 有一个暴力的 dp 是设 $dp_{i,j,k}$ 表示从上到下第 i 层，当前层 j 个节点，k 个叶子的答案。复杂度 $O(n^4)$。
- 我们发现记录 i 是没有必要的，可以进一步拆 $dep_i w_i$ 的贡献，也就是每下放一层统计一次 w_i 的贡献。

- 考虑如何计算贡献。容易发现，对于第 i 层的第 x 个叶子节点，它的贡献是 $2^{lowbit(x)}$。这样就可以形式化刻画贡献。
- 有一个暴力的 dp 是设 $dp_{i,j,k}$ 表示从上到下第 i 层，当前层 j 个节点，k 个叶子的答案。复杂度 $O(n^4)$。
- 我们发现记录 i 是没有必要的，可以进一步拆 $dep_i w_i$ 的贡献，也就是每下放一层统计一次 w_i 的贡献。
- $dp_{i,j}$ 表示当前用了 i 个叶子，当前层 j 个节点的贡献。转移在这一层钦定一个叶子或者下放一层。复杂度 $O(n^2)$。

- 对于一个集合 A，将其元素按照任意顺序排列得到 a_i，记 $b_i = \lfloor \frac{\sum_{j=0}^{i} a_j}{10} \rfloor$，定义 $f(A) = \max |B|$。
- 给定可重集合 A，求其所有本质不同的子集 B 的 $f(B)$ 之和。
- $n \leq 2000, a_i \leq 12$。

- 对于 $a_i \le 11$ 的情况，显然有上界 $\min(\frac{|S|}{10}, n)$，并且可以贪心取到这个上界。

- 对于 $a_i \le 11$ 的情况，显然有上界 $\min(\frac{|S|}{10}, n)$，并且可以贪心取到这个上界。
- 如果当前前缀和末位不是 9 就放 11，否则随便放，如果最后剩下 11 就能取到 n，否则取到 $\frac{S}{10}$，直接 dp 即可做到 $O(n^2|a|)$。

- 对于 $a_i \le 11$ 的情况，显然有上界 $\min(\frac{|S|}{10}, n)$，并且可以贪心取到这个上界。
- 如果当前前缀和末位不是 9 就放 11，否则随便放，如果最后剩下 11 就能取到 n，否则取到 $\frac{S}{10}$，直接 dp 即可做到 $O(n^2|a|)$。
- 对于 $a_i \le 12$ 的情况，上述上界不再正确，因为有些数并不再能贡献有效进位。

- 对于 $a_i \le 11$ 的情况，显然有上界 $\min(\frac{|S|}{10}, n)$，并且可以贪心取到这个上界。
- 如果当前前缀和末位不是 9 就放 11，否则随便放，如果最后剩下 11 就能取到 n，否则取到 $\frac{S}{10}$，直接 dp 即可做到 $O(n^2|a|)$。
- 对于 $a_i \le 12$ 的情况，上述上界不再正确，因为有些数并不再能贡献有效进位。
- 这里不能贡献有效进位的数其实是 1，前缀和末位是偶数的时候是不能放 1 的，所以每个 1 需要和一个奇数一起带来进位，否则只能两个一起带来一个进位。也就是上界为 $\min(\frac{S}{10}, n - \frac{\max(0, c_1 - (c_3+c_5+c_7+c_9+c_{11}))}{2})$。

- 暴力进行 dp 是 $O(n^3|a|)$ 的，考虑优化。

- 暴力进行 dp 是 $O(n^3|a|)$ 的，考虑优化。
- 首先对于 $3,5,7,9,11$ 和 $2,4,6,8,10,12$ 这两部分的转移，都是可以预处理 $f_{i,j}$ 和 $g_{i,j}$ 并利用简单前缀和做到 $O(n^2)$ 的。问题在于在 f 中转移 1。

- 暴力进行 dp 是 $O(n^3|a|)$ 的，考虑优化。
- 首先对于 $3,5,7,9,11$ 和 $2,4,6,8,10,12$ 这两部分的转移，都是可以预处理 $f_{i,j}$ 和 $g_{i,j}$ 并利用简单前缀和做到 $O(n^2)$ 的。问题在于在 f 中转移 1。
- 发现转移的形式是在一些斜率为 1 或 2 的直线上转移，可以对每条这种直线记录前缀和数组。复杂度 $O(n^2|a|)$，可以通过。

- 给定 n, V，求 $\sum_{T\ n} f(T)$，$f(T)$ 为给 T 每个节点在 $[1, V]$ 中赋权的合法方案数。定义一个方案是合法的，当且仅当对于所有连通块 S 满足 $\mathrm{mex}_{i\in S} a_i = \min_{i\notin S} a_i$。
- $n \le 150$。

- 分析性质。显然 0 只能有恰好一个，继续分析 1 也是恰好一个。再继续分析，发现 2 要么也是恰好一个，要么全都位于 01 形成的虚树内部。

- 分析性质。显然 0 只能有恰好一个，继续分析 1 也是恰好一个。再继续分析，发现 2 要么也是恰好一个，要么全都位于 01 形成的虚树内部。
- 继续归纳，可以得到在 $\leq i$ 的点形成的虚树上，i 要么不超过一个，要么没有虚树上度数为 1 的点。这个就是充要条件。

- 分析性质。显然 0 只能有恰好一个，继续分析 1 也是恰好一个。再继续分析，发现 2 要么也是恰好一个，要么全都位于 01 形成的虚树内部。
- 继续归纳，可以得到在 $\leq i$ 的点形成的虚树上，i 要么不超过一个，要么没有虚树上度数为 1 的点。这个就是充要条件。
- 设 $dp_{i,j,k}$ 表示考虑点权 $\leq i$ 的虚树，虚树大小为 j，$\leq i$ 的节点数为 k 的方案数。转移考虑新加入一个点挂在虚树外面扩大虚树大小，或者在虚树内部若干空着的点填上 i。复杂度 $O(n^4)$。

致谢

谢谢大家！