中国计算机学会 China Computer Federation

组合计数问题的常用技巧

彭博

广州大学附属中学

2021 年 5 月 21 日

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧

5 综合应用

中国计算机学会 China Computer Federation

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧

5 综合应用

要讲什么

如题所示，要讲组合计数。

要讲什么

如题所示，要讲组合计数。

但是组合计数是一个很大的话题，所以这里仅仅涉及一些常用技巧。由于听课同学的基础可能参差不齐，选题不会用到太多知识点。基础的前置知识有：

要讲什么

如题所示，要讲组合计数。
但是组合计数是一个很大的话题，所以这里仅仅涉及一些常用技巧。由于听课同学的基础可能参差不齐，选题不会用到太多知识点。基础的前置知识有：

1. 加法原理、乘法原理。
2. 简单 DP 技巧。
3. 简单逻辑推理能力。

要讲什么

如题所示，要讲组合计数。

但是组合计数是一个很大的话题，所以这里仅仅涉及一些常用技巧。由于听课同学的基础可能参差不齐，选题不会用到太多知识点。基础的前置知识有：

1. 加法原理、乘法原理。
2. 简单 DP 技巧。
3. 简单逻辑推理能力。

相信大家已经看出这是一节不可能掉线的课了。

为什么要讲

在对我的学弟的观察中，我发现很多刚刚开始接触计数题的同学会对计数题怀有“本能的恐惧”(包括我也曾经是这样)。

为什么要讲

在对我的学弟的观察中，我发现很多刚刚开始接触计数题的同学会对计数题怀有“本能的恐惧”(包括我也曾经是这样)。
所以这次讲课以总结组合计数问题的一些常用技巧为主，希望同学们听完之后能在面对计数问题时不再手足无措。

为什么要讲

在对我的学弟的观察中，我发现很多刚刚开始接触计数题的同学会对计数题怀有“本能的恐惧”(包括我也曾经是这样)。
所以这次讲课以总结组合计数问题的一些常用技巧为主，希望同学们听完之后能在面对计数问题时不再手足无措。
~~也希望大家好好听讲，不要掉线。~~

为什么要讲

在对我的学弟的观察中，我发现很多刚刚开始接触计数题的同学会对计数题怀有“本能的恐惧”(包括我也曾经是这样)。

所以这次讲课以总结组合计数问题的一些常用技巧为主，希望同学们听完之后能在面对计数问题时不再手足无措。

~~也希望大家好好听讲，不要掉线。~~

下面将会介绍若干种技巧，并配有对应的例题。题目按照对应技巧排序，难度乱序，不幸掉线的同学可以在下一题迅速重连。

为什么要讲

中国计算机学会 China Computer Federation

在对我的学弟的观察中，我发现很多刚刚开始接触计数题的同学会对计数题怀有“本能的恐惧”(包括我也曾经是这样)。
所以这次讲课以总结组合计数问题的一些常用技巧为主，希望同学们听完之后能在面对计数问题时不再手足无措。
~~也希望大家好好听讲，不要掉线。~~
下面将会介绍若干种技巧，并配有对应的例题。题目按照对应技巧排序，难度乱序，不幸掉线的同学可以在下一题迅速重连。
如无特殊说明，答案均模任意大质数。

1 前言

2 寻找唯一性
何为唯一
最优方案唯一
不合法元素唯一

3 判定合法

4 其他技巧

5 综合应用

1 前言

2 寻找唯一性
- 何为唯一
- 最优方案唯一
- 不合法元素唯一

3 判定合法

4 其他技巧

5 综合应用

计数中，最常出现的问题便是数重或数漏。比如要求数合法的元素个数，但一个元素可能有多种合法的方式，那么对合法方案计数就会数重。

一种解决方法是，把合法元素唯一对应到一种合法方案上，相当于添加了限制条件。这样就不会出问题了。

1 前言

2 寻找唯一性
何为唯一
最优方案唯一
不合法元素唯一

3 判定合法

4 其他技巧

5 综合应用

CF1264D2 Beautiful Bracket Sequence

递归定义合法括号序列及其深度：

- 空串是合法括号序列，深度为 0 。
- 如果 s 是合法括号序列，那么 (s) 也是合法括号序列，深度为 s 的深度加一。
- 如果 s,t 是合法括号序列，那么 st 也是合法括号序列，深度为 s,t 的深度的较大值。

定义任意一个括号序列的深度为其所有合法括号子序列的深度的最大值。

给出一个包含 ()? 的字符串，? 可以任意替换为) 或 (，求出每一种替换方案的深度之和。

$n \le 10^6$

CF1264D2 Beautiful Bracket Sequence

考虑对于一个固定的串如何计算其深度。
容易发现选出的子序列必然长度为 $2k$，前 k 个字符是 (，后 k 个字符是)，因为额外选出的字符不会使答案更优。
那么就必然在原串中可以找到一个位置，把左边的 (和右边的) 全部选进去，然后把较多的那一边删掉一些。
考虑从左往右枚举这个位置，那么左边 (的个数 c_1 单调不降，右边) 的个数 c_2 单调不增，并且每走一步一定会恰好使得 $c_1 := c_1 + 1$ 或 $c_2 := c_2 - 1$ 其中之一发生。
那么一定在 $c_1 = c_2$ 的时候取到 $\max(c_1, c_2)$ 的最大值，且这个位置存在且唯一。

CF1264D2 Beautiful Bracket Sequence

现在我们把每一个串映射到了唯一的一个位置，然后考虑如何计数。

容易想到枚举这个位置。设左边有 l_1 个?，l_2 个 (，右边同理分别有 r_1, r_2 个?,)。

先假设 r_1 个? 全部变成)，那么设 $d = r_1 + r_2 - l_2$ 表示左右的个数之差。现在在左边选一个? 就表示给左边加一个 (，右边选一个) 就表示给右边减一个)。

所以贡献是

$$\sum_{i=0}^{d} \binom{l_1}{i} \binom{r_1}{d-i} (i + l_2)$$

化简为 $l_2\binom{l_1+r_1}{d} + l_1\binom{l_1+r_1-1}{d-1}$，可以 $O(1)$ 计算。总复杂度 $O(n)$。

1 前言

2 寻找唯一性
- 何为唯一
- 最优方案唯一
- 不合法元素唯一

3 判定合法

4 其他技巧

5 综合应用

LOJ#3211. 「CSP-S 2019」Emiya 家今天的饭

给定一个 $n \times m$ 的表格，每个位置写有非负整数 $a_{i,j}$ 。
你需要在表格中选若干个位置，满足

- 至少选取一个位置；
- 每行最多选一个位置；
- 设总共选了 c 个位置，那么不存在一列选的个数超过 $\lfloor \frac{c}{2} \rfloor$ 。

对于一种合法方案，它的权值是所有选的位置的 $a_{i,j}$ 的乘积。
求所有合法方案的权值之和。
$n \le 100, m \le 2000$

LOJ#3211. 「CSP-S 2019」Emiya 家今天的饭

显然，在三个限制中，只有最后这个限制比较难处理。

LOJ#3211. 「CSP-S 2019」Emiya 家今天的饭

显然，在三个限制中，只有最后这个限制比较难处理。
但是注意到显然只有一列可能打破这个限制，所以只需要用总方案数减去每一列打破限制的方案数，就得到了合法的方案数。

LOJ#3211. 「CSP-S 2019」Emiya 家今天的饭

显然，在三个限制中，只有最后这个限制比较难处理。
但是注意到显然只有一列可能打破这个限制，所以只需要用总方案数减去每一列打破限制的方案数，就得到了合法的方案数。
对于固定的一列，容易用 $O(n^2)$ 的 DP 求出它超过一半的方案数，所以总复杂度 $O(n^2m)$ 。

AGC052C Nondivisible Prefix Sums

给定质数 P 和长度 n 。
对于 $(P-1)^n$ 个值域在 $[1, P-1]$ 的数列，称它是好的，当且仅当存在一种方法把它重排，使得每个前缀和都不是 P 的倍数。
求好的数列个数。
$1 \le n \le 5000, 3 \le P \le 10^8$

AGC052C Nondivisible Prefix Sums

如果不要求重排，那么容易发现答案是 $(P-1)(P-2)^{n-1}$ 。然而重排的方案太多，很难保证不数重。

AGC052C Nondivisible Prefix Sums

如果不要求重排，那么容易发现答案是 $(P-1)(P-2)^{n-1}$ 。然而重排的方案太多，很难保证不数重。
考虑一个很简单的构造：把数列任意排序，然后从前往后扫，遇到某个前缀和 $sum_i = 0$ 时，就从 i 后面选择一个 $a_j \neq a_i$ ，然后交换 a_i, a_j 。但是做到最后可能会发现只剩下一种数，于是找不到 $a_j \neq a_i$ ，于是构造就失败了。

AGC052C Nondivisible Prefix Sums

如果不要求重排，那么容易发现答案是 $(P-1)(P-2)^{n-1}$ 。然而重排的方案太多，很难保证不数重。
考虑一个很简单的构造：把数列任意排序，然后从前往后扫，遇到某个前缀和 $sum_i = 0$ 时，就从 i 后面选择一个 $a_j \neq a_i$ ，然后交换 a_i, a_j 。但是做到最后可能会发现只剩下一种数，于是找不到 $a_j \neq a_i$ ，于是构造就失败了。
这提示我们，不合法的数列可能仅仅是因为一种元素出现太多次。当然，如果数列中所有元素加起来是 P 的倍数那么肯定也不合法。

AGC052C Nondivisible Prefix Sums

把数列中出现次数最多的元素提出来。因为 P 是质数，所以不妨假设它是 1 。

AGC052C Nondivisible Prefix Sums

把数列中出现次数最多的元素提出来。因为 P 是质数，所以不妨假设它是 1 。

为了让 1 不在最后出现太多次，我们尝试在前面尽可能把它消掉。初始可以直接放 $P-1$ 个 1 ，然后每放一个 x ，就可以放 $P-x$ 个 1 。如果到最后还剩下 1 那么就显然不合法了。

AGC052C Nondivisible Prefix Sums

把数列中出现次数最多的元素提出来。因为 P 是质数，所以不妨假设它是 1 。
为了让 1 不在最后出现太多次，我们尝试在前面尽可能把它消掉。初始可以直接放 $P-1$ 个 1 ，然后每放一个 x ，就可以放 $P-x$ 个 1 。如果到最后还剩下 1 那么就显然不合法了。
然后我们证明，如果用这样的方法可以把 1 消掉，并且总和不是 P 的倍数，那么就存在一种重排的方案。

AGC052C Nondivisible Prefix Sums

把数列中出现次数最多的元素提出来。因为 P 是质数，所以不妨假设它是 1 。

为了让 1 不在最后出现太多次，我们尝试在前面尽可能把它消掉。初始可以直接放 $P-1$ 个 1 ，然后每放一个 x ，就可以放 $P-x$ 个 1 。如果到最后还剩下 1 那么就显然不合法了。

然后我们证明，如果用这样的方法可以把 1 消掉，并且总和不是 P 的倍数，那么就存在一种重排的方案。

使用一开始那个构造的思路。设当前的前缀和为 cur ，剩下的数中出现次数最多的是 x 。如果 $P \nmid (x + cur)$ 那么直接把 x 加入，否则另取 $y \neq x$ 且 y 的出现次数在剩下的元素中最大，先加入 y 再加入 x 。

AGC052C Nondivisible Prefix Sums

什么时候这个算法会挂呢？做到最后发现还剩下至少两个 x ，且 $P \mid (x + cur)$ ，且没有剩下其他元素了。但是注意到，对于任意一个元素 t ，如果初始时它的出现次数最多，那么在过程中它的出现次数不会比最多的少 2 。

AGC052C Nondivisible Prefix Sums

什么时候这个算法会挂呢？做到最后发现还剩下至少两个 x ，且 $P \mid (x + cur)$ ，且没有剩下其他元素了。但是注意到，对于任意一个元素 t ，如果初始时它的出现次数最多，那么在过程中它的出现次数不会比最多的少 2 。

所以 x 就是出现次数最多的元素，即 $x = 1$ 。但是这样的话那么构造出来的数列就和前面尽量消掉 1 的数列相同了，而前面是消得掉的，所以矛盾。

AGC052C Nondivisible Prefix Sums

从前面的分析可以看出，如果 $P \nmid sum_n$ 但仍然不合法，那么一定存在一个严格众数出现次数过多。

AGC052C Nondivisible Prefix Sums

从前面的分析可以看出，如果 $P \nmid sum_n$ 但仍然不合法，那么一定存在一个严格众数出现次数过多。
所以只需要用总方案数减去 $(P-1)$ 乘 1 出现次数过多的方案即可。只需要一个简单背包。

1. 前言
2. 寻找唯一性
3. 判定合法
   - 例一
   - 例二
4. 其他技巧
5. 综合应用

1 前言

2 寻找唯一性

3 判定合法

例一

例二

4 其他技巧

5 综合应用

AGC012F Prefix Median

给定长度为 $2n-1$ 的数列 a （可能有相同元素），可以将它任意打乱顺序，然后定义 b_i 为前 $2i-1$ 个数的中位数。求能生成多少种数列 b 。

$n \le 50$

AGC012F Prefix Median

考虑怎样得到一个序列 b 。有这样一个贪心：每个时刻，如果当前的 b_i 还未出现就把它加入 a 中，然后用还没有放进 b 的最大最小值调整中位数。显然一次调整只能让中位数移动一格。

AGC012F Prefix Median

考虑怎样得到一个序列 b 。有这样一个贪心：每个时刻，如果当前的 b_i 还未出现就把它加入 a 中，然后用还没有放进 b 的最大最小值调整中位数。显然一次调整只能让中位数移动一格。
也就是说，已经被放入的数集长成这样：中间是一堆零散的作为中位数出现过的数，两边是连续的前缀后缀。

AGC012F Prefix Median

考虑怎样得到一个序列 b 。有这样一个贪心：每个时刻，如果当前的 b_i 还未出现就把它加入 a 中，然后用还没有放进 b 的最大最小值调整中位数。显然一次调整只能让中位数移动一格。
也就是说，已经被放入的数集长成这样：中间是一堆零散的作为中位数出现过的数，两边是连续的前缀后缀。
在此基础上，容易得到合法序列 b 的必要条件：

AGC012F Prefix Median

考虑怎样得到一个序列 b 。有这样一个贪心：每个时刻，如果当前的 b_i 还未出现就把它加入 a 中，然后用还没有放进 b 的最大最小值调整中位数。显然一次调整只能让中位数移动一格。

也就是说，已经被放入的数集长成这样：中间是一堆零散的作为中位数出现过的数，两边是连续的前缀后缀。

在此基础上，容易得到合法序列 b 的必要条件：

1. 每个位置都有上下界，分别是全部取最大和全部取最小所得到的中位数。
2. 不存在 $j < i$ 满足 $\min(b_{i-1}, b_i) < b_j < \max(b_{i-1}, b_i)$ ，即不能一次移动两步。

AGC012F Prefix Median

考虑怎样得到一个序列 b 。有这样一个贪心：每个时刻，如果当前的 b_i 还未出现就把它加入 a 中，然后用还没有放进 b 的最大最小值调整中位数。显然一次调整只能让中位数移动一格。

也就是说，已经被放入的数集长成这样：中间是一堆零散的作为中位数出现过的数，两边是连续的前缀后缀。

在此基础上，容易得到合法序列 b 的必要条件：

1. 每个位置都有上下界，分别是全部取最大和全部取最小所得到的中位数。
2. 不存在 $j < i$ 满足 $\min(b_{i-1}, b_i) < b_j < \max(b_{i-1}, b_i)$ ，即不能一次移动两步。

可以用上面的构造来证明这是充分条件。

AGC012F Prefix Median

注意到条件中只关注相对大小关系，所以可以在确定上下界之后就可以忽略相同的数。注意上界单调不升，下界单调不降，所以只需要关心当前区间中有哪些数出现，不需要考虑外面的。

AGC012F Prefix Median

注意到条件中只关注相对大小关系，所以可以在确定上下界之后就可以忽略相同的数。注意上界单调不升，下界单调不降，所以只需要关心当前区间中有哪些数出现，不需要考虑外面的。
另外，我们只关心大小关系而不关心具体的值，所以只需要记录区间中有几个数，当某个数被排除在区间外时才确定它的值。

AGC012F Prefix Median

注意到条件中只关注相对大小关系，所以可以在确定上下界之后就可以忽略相同的数。注意上界单调不升，下界单调不降，所以只需要关心当前区间中有哪些数出现，不需要考虑外面的。

另外，我们只关心大小关系而不关心具体的值，所以只需要记录区间中有几个数，当某个数被排除在区间外时才确定它的值。

设 $dp_{i,j,k}$ 表示确定了前 i 个位置，有 j 个不同的数现在仍在合法区间中，当前的中位数是第 k 个，的方案数。随便转移。

1 前言

2 寻找唯一性

3 判定合法
   例一
   例二

4 其他技巧

5 综合应用

AGC043D Merge Triplets

给定 n ，求长度为 $3n$ 的能被以下方法生成的排列个数。

- 生成 n 个长度为 3 的数列，使得每个 1 到 $3n$ 中的数恰好被用一次。
- 生成一个初始为空的数列 P ，重复以下操作 $3n$ 次：
  - 在所有非空数列的开头中选取最小的元素 x 。
  - 把 x 加入到 P 的末尾，并从原数列删去。

$n \leq 2000$

AGC043D Merge Triplets

显然造排列的方式就是归并排序，而归并排序的性质：把每一个序列按前缀最大值分成若干段，然后把段按段头的大小排序。

AGC043D Merge Triplets

显然造排列的方式就是归并排序，而归并排序的性质：把每一个序列按前缀最大值分成若干段，然后把段按段头的大小排序。所以判断一个序列合法的方式：从第一个开始，每次把比段头小的数全部加入这一段。那么合法当且仅当每一段长度不超过 3 ，且 2 的个数小于等于 1 的个数。

AGC043D Merge Triplets

显然造排列的方式就是归并排序，而归并排序的性质：把每一个序列按前缀最大值分成若干段，然后把段按段头的大小排序。

所以判断一个序列合法的方式：从第一个开始，每次把比段头小的数全部加入这一段。那么合法当且仅当每一段长度不超过 3，且 2 的个数小于等于 1 的个数。

设 $f_{i,j}$ 表示把 i 个数分成若干段，使得 1 的个数减 2 的个数是 j，的方案数。由于最大的那个数一定是最后一段的段头，所以枚举它所在的段的长度是 1/2/3 即可转移。

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧
   - 差分
   - 贡献的分配

5 综合应用

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧
- 差分
- 贡献的分配

5 综合应用

为什么要差分

中国计算机学会 China Computer Federation

差分大家都很熟悉，但是计数的时候为什么要差分呢？
所谓差分，其实只是这样一个式子：

$$x = \sum_{0 \le i < x} 1$$

这样就可以在外面枚举 i ，然后只关心结果与 i 的大小关系。有时可以极大地简化问题。
这是一个很常见的 trick ，不过由于篇幅关系只有一道题。

LOJ#3463. 「WC2021」表达式求值

~~相信大家都很熟悉这题。~~

定义 $\mathbb{Z}$ 上的二元运算符 $<, >$，分别返回两者取 $\min$ 或 $\max$。

给出一个只包含 <,>,?,(,),0 到 9 的表达式 E，其中 0 到 9 的数字 i 表示变量 x_i，? 表示暂未确定这个位置填 $<$ 还是 $>$。

有 n 次询问，每次给出这些变量的值，求对于所有把? 替换为 $<$ 或 $>$ 的方案中，表达式的返回值的和。

$|E|, n \le 5 \times 10^4$

LOJ#3463. 「WC2021」表达式求值

中国计算机学会 China Computer Federation

显然返回值只能是 $x_0, \cdots, x_9$ 中的一个，而是哪个又只和它们的大小关系有关，于是获得了一个 $O(10!|E| + 10n)$ 的做法。

LOJ#3463. 「WC2021」表达式求值

显然返回值只能是 $x_0, \cdots, x_9$ 中的一个，而是哪个又只和它们的大小关系有关，于是获得了一个 $O(10!|E| + 10n)$ 的做法。

因为只和大小关系有关，所以考虑差分。对于一个 i ，把比它大的 x 变为 1 ，比它小的变为 0 ，那么只关心返回值是 1 还是 0 。

LOJ#3463. 「WC2021」表达式求值

显然返回值只能是 $x_0, \cdots, x_9$ 中的一个，而是哪个又只和它们的大小关系有关，于是获得了一个 $O(10!|E| + 10n)$ 的做法。
因为只和大小关系有关，所以考虑差分。对于一个 i，把比它大的 x 变为 1，比它小的变为 0，那么只关心返回值是 1 还是 0。
但是此时只有 2^{10} 种不同的 01 串，所以预处理的复杂度显著降低。总复杂度 $O(2^{10}|E| + 10n)$。

LOJ#3463. 「WC2021」表达式求值

显然返回值只能是 $x_0, \cdots, x_9$ 中的一个，而是哪个又只和它们的大小关系有关，于是获得了一个 $O(10!|E| + 10n)$ 的做法。

因为只和大小关系有关，所以考虑差分。对于一个 i ，把比它大的 x 变为 1 ，比它小的变为 0 ，那么只关心返回值是 1 还是 0 。但是此时只有 2^{10} 种不同的 01 串，所以预处理的复杂度显著降低。总复杂度 $O(2^{10}|E| + 10n)$ 。

中国计算机学会 China Computer Federation

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧
- 差分
- 贡献的分配

5 综合应用

也许算是一种配凑法。

ARC082E ConvexScore

给定平面上 n 个两两不相同的点，对于一个点集 S ，如果它不在一条直线上（即凸包存在），那么设它的凸包大小为 k ，它的权值就是 $2^{|S|-k}$ 。

求所有凸包存在的点集的权值之和。

$n \leq 200$

ARC082E ConvexScore

看到 $2^{n-|S|}$，容易想到子集个数。

ARC082E ConvexScore

看到 $2^{n-|S|}$，容易想到子集个数。
于是可以想到枚举点集，然后贡献到这个点集的凸包上。

ARC082E ConvexScore

看到 $2^{n-|S|}$，容易想到子集个数。
于是可以想到枚举点集，然后贡献到这个点集的凸包上。
于是就转化成有多少个点集存在凸包，改成多少个点集没有凸包，然后就很好做了。

LOJ#3053. 「十二省联考 2019」希望

给定一棵 n 个点的树，问有多少个长度为 k 的连通块有序列 $\{s_i\}_{i=1}^{k}$ （即任意选出 k 个树上连通块），使得存在一个点 x ，使得 $\forall y \in s_i, dis(x,y) \le L$ ，其中 L 给定。

$n \le 10^6, k \le 10$

因为进一步的优化与本文内容无关，所以可以认为数据范围是 $n \le 3000$ 。

LOJ#3053. 「十二省联考 2019」希望

容易发现，对于一组确定的 $\{s\}$，合法的点 x 一定组成了一个树上连通块。

LOJ#3053. 「十二省联考 2019」希望

容易发现，对于一组确定的 $\{s\}$，合法的点 x 一定组成了一个树上连通块。

对于这类题有一个技巧，是注意到一个树上连通块满足 $|V|-|E|=1$，而空集满足 $|V|-|E|=0$。所以只需要用合法的点数减去合法的边数即可。

LOJ#3053. 「十二省联考 2019」希望

容易发现，对于一组确定的 $\{s\}$，合法的点 x 一定组成了一个树上连通块。

对于这类题有一个技巧，是注意到一个树上连通块满足 $|V|-|E|=1$，而空集满足 $|V|-|E|=0$。所以只需要用合法的点数减去合法的边数即可。

以点数为例，我们需要对每个点求出 f_x 表示距离 x 不超过 L 的连通块个数，然后对 f_x^k 求和。

LOJ#3053. 「十二省联考 2019」希望

容易发现，对于一组确定的 $\{s\}$ ，合法的点 x 一定组成了一个树上连通块。

对于这类题有一个技巧，是注意到一个树上连通块满足 $|V|-|E|=1$ ，而空集满足 $|V|-|E|=0$ 。所以只需要用合法的点数减去合法的边数即可。

以点数为例，我们需要对每个点求出 f_x 表示距离 x 不超过 L 的连通块个数，然后对 f_x^k 求和。

设 $dn_{x,i}, up_{x,i}$ 分别表示子树内和子树外，距离 x 最远的点不超过 i ，的连通块个数，即可做到 $O(nL)$ 。

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧

5 综合应用
- 例一
- 例二
- 备选一
- 备选二

你已经具有基本的数数能力了，一起来做几道题吧！

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧

5 综合应用
- 例一
- 例二
- 备选一
- 备选二

LOJ#2719. 「NOI2018」冒泡排序（简化）

给定一个长度为 n 的排列 q 。
定义一个排列是好的，当且仅当它的最长下降子序列长度不超过 2 。
求字典序严格大于 q 的好的排列有多少个。
$n \leq 6 \times 10^5, \sum n \leq 2 \times 10^6$

LOJ#2719. 「NOI2018」冒泡排序（简化）

我们不管字典序的限制，先写出一个 DP：$dp_{i,j}$ 表示考虑了前 i 个，之前最大值是 j，的方案数。转移就考虑下一个位置是填一个比最大值更大的数，或是填还没有填的数里面最小的数。显然合法的转移只有这两种。

LOJ#2719. 「NOI2018」冒泡排序（简化）

我们不管字典序的限制，先写出一个 DP：$dp_{i,j}$ 表示考虑了前 i 个，之前最大值是 j，的方案数。转移就考虑下一个位置是填一个比最大值更大的数，或是填还没有填的数里面最小的数。显然合法的转移只有这两种。
那么就是 $dp_{i,j} \to dp_{i+1,k}, k \geq j$ 。我们假装 $dp_{i+1,i}$ 这里也有一个虚点，那么就可以把转移看成往右走一步，然后往上走若干步。初始从 $dp_{0,0} = 1$ 开始走，第一步必须向右然后向上若干步。最后答案是 $dp_{n,n}$。

LOJ#2719. 「NOI2018」冒泡排序（简化）

我们不管字典序的限制，先写出一个 DP：$dp_{i,j}$ 表示考虑了前 i 个，之前最大值是 j，的方案数。转移就考虑下一个位置是填一个比最大值更大的数，或是填还没有填的数里面最小的数。显然合法的转移只有这两种。

那么就是 $dp_{i,j} \to dp_{i+1,k}, k \geq j$ 。我们假装 $dp_{i+1,i}$ 这里也有一个虚点，那么就可以把转移看成往右走一步，然后往上走若干步。初始从 $dp_{0,0} = 1$ 开始走，第一步必须向右然后向上若干步。最后答案是 $dp_{n,n}$。

观察这个 DP 式子，发现把 $dp_{i,j}$ 看做点 (i,j)，那么它的值就是从 $(1,1)$ 走到 (i,j)，只能向右或向上，并且不能摸到 $y = x - 2$ 这一条直线，的方案数。

不能碰到某条直线的这个套路我们非常熟悉，就是把起点对于它对称一下，方案数减掉，就可以了。

LOJ#2719. 「NOI2018」冒泡排序（简化）

我们不管字典序的限制，先写出一个 DP：$dp_{i,j}$ 表示考虑了前 i 个，之前最大值是 j，的方案数。转移就考虑下一个位置是填一个比最大值更大的数，或是填还没有填的数里面最小的数。显然合法的转移只有这两种。

那么就是 $dp_{i,j} \to dp_{i+1,k}, k \geq j$ 。我们假装 $dp_{i+1,i}$ 这里也有一个虚点，那么就可以把转移看成往右走一步，然后往上走若干步。初始从 $dp_{0,0} = 1$ 开始走，第一步必须向右然后向上若干步。最后答案是 $dp_{n,n}$。

观察这个 DP 式子，发现把 $dp_{i,j}$ 看做点 (i,j)，那么它的值就是从 $(1,1)$ 走到 (i,j)，只能向右或向上，并且不能摸到 $y = x - 2$ 这一条直线，的方案数。

不能碰到某条直线的这个套路我们非常熟悉，就是把起点对于它对称一下，方案数减掉，就可以了。

于是我们有 $dp_{n,n} = \binom{2n-2}{n-1} - \binom{2n-2}{n-3}$。

LOJ#2719. 「NOI2018」冒泡排序（简化）

那么现在再来看一看字典序的限制，容易想到枚举前面几项相等，有一项更大，然后后面随便填。

LOJ#2719. 「NOI2018」冒泡排序（简化）

那么现在再来看一看字典序的限制，容易想到枚举前面几项相等，有一项更大，然后后面随便填。
这个怎么处理呢？设第 i 位不同，已经填完的数的最大值为 mx，那么这一位就必须填大于 $\max(mx, q_i)$ 的数。为什么？如果小于 mx，那么就必须是未出现里面最小的数，就一定小于等于 a_i 了，就不合法了。

LOJ#2719. 「NOI2018」冒泡排序（简化）

那么现在再来看一看字典序的限制，容易想到枚举前面几项相等，有一项更大，然后后面随便填。

这个怎么处理呢？设第 i 位不同，已经填完的数的最大值为 mx，那么这一位就必须填大于 $\max(mx, q_i)$ 的数。为什么？如果小于 mx，那么就必须是未出现里面最小的数，就一定小于等于 a_i 了，就不合法了。

这相当于限定起点是 $(i, \max(q_i, mx) + 1)$，仍然是走到 (n, n)，所以方案数也很容易算。

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧

5 综合应用
- 例一
- 例二
- 备选一
- 备选二

AGC041F Histogram Rooks

给定一个 n 列的棋盘，第 i 列只有下面 h_i 行的格子有效。要在上面放棋子（车），一个棋子可以覆盖同行同列的所有格子（但不能穿过无效格子）。问有多少种放棋子的方法使得每个有效格子都被覆盖。

$n \leq 400$

AGC041F Histogram Rooks

容斥 0：枚举哪些格子没有被覆盖，然后对于一个位置如果行列都没有被钦定未被覆盖的格子，那么方案数是 2 。
显然这个容斥的复杂度非常爆炸，而且也不是很好 DP 。

AGC041F Histogram Rooks

容斥 0：枚举哪些格子没有被覆盖，然后对于一个位置如果行列都没有被钦定未被覆盖的格子，那么方案数是 2 。
显然这个容斥的复杂度非常爆炸，而且也不是很好 DP 。
容斥 1：枚举哪些列里面有格子被钦定未被覆盖，那么这些列都不能放棋子了。设集合为 S 。
那么此时考虑一个极长行连续段的贡献，假设里面有 p 个格子在 S 里，那么贡献有两种：

1. 这 p 个格子都没有钦定未覆盖，所以剩下的格子可以随便放，方案数 2^{len-p} 。
2. 枚举里面有几个格子被钦定，方案数是 $\sum_{i=1}^{p}\binom{p}{i}(-1)^i = -[p \neq 0]$ 。

但是这样算出来的结果仍然是错的，因为某一些在 S 里的列可能还是没有钦定的格子。

AGC041F Histogram Rooks

容斥 2：在容斥 1 的基础上再套一个容斥，枚举集合 $T \subseteq S$，表示 T 里面没有被钦定的格子。

AGC041F Histogram Rooks

容斥 2：在容斥 1 的基础上再套一个容斥，枚举集合 $T \subseteq S$，表示 T 里面没有被钦定的格子。
设一个极长行连续段里面有 q 个格子在 T 里面，那么第二种的贡献变成了 $-[p \neq q]$。
注意到此时贡献只和 $p, [p \neq q]$ 有关，所以 DP 的状态数可能不会太大，来尝试设计一个 DP。

AGC041F Histogram Rooks

容斥 2：在容斥 1 的基础上再套一个容斥，枚举集合 $T \subseteq S$，表示 T 里面没有被钦定的格子。
设一个极长行连续段里面有 q 个格子在 T 里面，那么第二种的贡献变成了 $-[p \neq q]$。
注意到此时贡献只和 $p, [p \neq q]$ 有关，所以 DP 的状态数可能不会太大，来尝试设计一个 DP。
每次从 h 最小的位置切开来，下面会形成一个矩形，上面会被分裂成一些子问题，并且子问题形成树形结构，并且矩形里的行连续段的 p, q 就是由上面的加起来，再算上独有的列。
于是可以设 $dp_{x,p,0/1}$，p 表示 S 的大小，0/1 表示是否有 $p \neq q$。

AGC041F Histogram Rooks

容斥 2：在容斥 1 的基础上再套一个容斥，枚举集合 $T \subseteq S$ ，表示 T 里面没有被钦定的格子。
设一个极长行连续段里面有 q 个格子在 T 里面，那么第二种的贡献变成了 $-[p \neq q]$ 。
注意到此时贡献只和 $p, [p \neq q]$ 有关，所以 DP 的状态数可能不会太大，来尝试设计一个 DP 。
每次从 h 最小的位置切开来，下面会形成一个矩形，上面会被分裂成一些子问题，并且子问题形成树形结构，并且矩形里的行连续段的 p, q 就是由上面的加起来，再算上独有的列。
于是可以设 $dp_{x,p,0/1}$ ，p 表示 S 的大小，0/1 表示是否有 $p \neq q$ 。
转移就是无脑从儿子用背包合并过来。对于独有的列也可以看做是一个特殊的儿子。
然后此时每一行的方案数都是相同的，直接快速幂即可。
复杂度 $O(n^2 \log n)$ ，预处理一些东西可以变成 $O(n^2)$ 。

中国计算机学会
China Computer Federation

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧

5 综合应用
- 例一
- 例二
- 备选一
- 备选二

CF848E Days of Floral Colours

有 $2n$ 个点围成一个环，要给它们染色。称一个染色方案是合法的，当且仅当

- 恰好染了 n 种颜色，每种颜色两个点。颜色无序。
- 如果点 x, y 同色，那么 x, y 对面的两个点也要同色。
- 两个同色点的距离只能是 $1, 2, n$ ，其中距离定义为劣弧的长度，比如相邻两个点的距离是 1 。

对于一个染色方案，定义它的权值为从距离为 n 的同色点处切开之后每一段的长度的乘积。求所有合法的染色方案的权值之和。如果不存在距离为 n 的同色点那么权值为 0 ；如果两对距离为 n 的同色点相邻那么权值也为 0 。

求出所有合法染色方案的权值和。

$n \leq 50000$

CF848E Days of Floral Colours

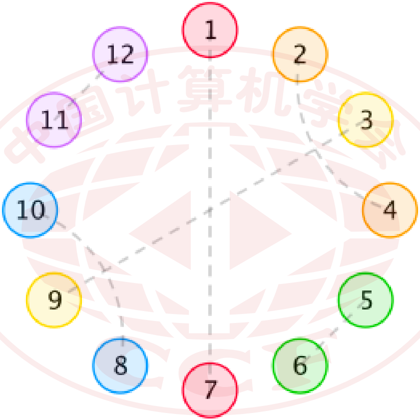

图 1: 一个合法方案

CF848E Days of Floral Colours

显然这个图形有很强的对称性，我们可以从某一对同色且距离为 n 的花的位置切开，分成两个完全对称的半圆，然后对半圆计数。

CF848E Days of Floral Colours

显然这个图形有很强的对称性，我们可以从某一对同色且距离为 n 的花的位置切开，分成两个完全对称的半圆，然后对半圆计数。切的位置选 1 之后的第一个，这样我们只需要最后枚举经过 1 的连续段长度即可。

CF848E Days of Floral Colours

显然这个图形有很强的对称性，我们可以从某一对同色且距离为 n 的花的位置切开，分成两个完全对称的半圆，然后对半圆计数。切的位置选 1 之后的第一个，这样我们只需要最后枚举经过 1 的连续段长度即可。

那么怎样对半圆计数呢？

CF848E Days of Floral Colours

显然这个图形有很强的对称性，我们可以从某一对同色且距离为 n 的花的位置切开，分成两个完全对称的半圆，然后对半圆计数。切的位置选 1 之后的第一个，这样我们只需要最后枚举经过 1 的连续段长度即可。

那么怎样对半圆计数呢？

对于某一段，它的方案数很像斐波那契数列，但是由于有向前后（即非这一段）的点连边的情况，不是很好看。

CF848E Days of Floral Colours

我们不妨换一种思路，不要在 DP 的时候一次加一段，而是一次加一个点，并记录最后两个位置是否有被匹配掉。

CF848E Days of Floral Colours

我们不妨换一种思路，不要在 DP 的时候一次加一段，而是一次加一个点，并记录最后两个位置是否有被匹配掉。
那么又有一个问题：现在一段的权值是 len^2，怎么处理这个东西。

CF848E Days of Floral Colours

我们不妨换一种思路，不要在 DP 的时候一次加一段，而是一次加一个点，并记录最后两个位置是否有被匹配掉。
那么又有一个问题：现在一段的权值是 len^2 ，怎么处理这个东西。
考虑用组合意义解决：这相当于是每一段里面要放一个红球一个蓝球，我们只需要再记录两维表示当前这一段红球/蓝球是否有放即可。

CF848E Days of Floral Colours

我们不妨换一种思路，不要在 DP 的时候一次加一段，而是一次加一个点，并记录最后两个位置是否有被匹配掉。
那么又有一个问题：现在一段的权值是 len^2 ，怎么处理这个东西。
考虑用组合意义解决：这相当于是每一段里面要放一个红球一个蓝球，我们只需要再记录两维表示当前这一段红球/蓝球是否有放即可。
最后，注意有可能半圆的两端的两朵花匹配上了，所以有两种 DP 初值，需要分别做 DP 。
总复杂度 $O(4^k n)$ ，其中 k 的大小视实现技巧而定。

中国计算机学会 China Computer Federation

1 前言

2 寻找唯一性

3 判定合法

4 其他技巧

5 综合应用
- 例一
- 例二
- 备选一
- 备选二

AGC021E Ball Eat Chameleons

有 n 只变色龙，初始颜色全部为蓝色。
给它们按某种顺序喂 K 个球，每个球的颜色都是红色或蓝色，一个球可以选择喂给任意一只变色龙。
一只变色龙吃掉一个球之后，如果它吃的红球数不等于蓝球数，那么它会变成较多的那种颜色，否则不变色。
问有多少种球的颜色序列，使得存在一种喂球的方案，把球全部喂完之后所有变色龙都是红色。
$n, K \leq 5 \times 10^5$

AGC021E Ball Eat Chameleons

考虑怎样判断一个序列是否合法。

AGC021E Ball Eat Chameleons

考虑怎样判断一个序列是否合法。
想到一个贪心策略：把第 n 个变色龙单独拿出来；拿到红球时，如果前 $n-1$ 个还有没变色的那就直接吃，否则喂给第 n 个；拿到蓝球时，如果前 $n-1$ 个有红色变色龙可以吃掉这个而不变色的那就吃掉，否则也喂给第 n 个。

AGC021E Ball Eat Chameleons

考虑怎样判断一个序列是否合法。
想到一个贪心策略：把第 n 个变色龙单独拿出来；拿到红球时，如果前 $n-1$ 个还有没变色的那就直接吃，否则喂给第 n 个；拿到蓝球时，如果前 $n-1$ 个有红色变色龙可以吃掉这个而不变色的那就吃掉，否则也喂给第 n 个。
容易发现这一定是最优策略。

AGC021E Ball Eat Chameleons

显然最终前 $n-1$ 个变色龙要么吃的红球比蓝球多一个（称这种情况为浪费），要么相等。设有 A 个红球，B 个蓝球，显然有 $A \geq B, A \geq n$ 。

AGC021E Ball Eat Chameleons

显然最终前 $n-1$ 个变色龙要么吃的红球比蓝球多一个（称这种情况为浪费），要么相等。设有 A 个红球，B 个蓝球，显然有 $A \geq B, A \geq n$ 。
我们只关心第 n 个变色龙最终的颜色。显然，如果它吃的红球个数大于蓝球那么一定合法；否则在相等的时候最后吃的球必须是蓝球。

AGC021E Ball Eat Chameleons

显然最终前 $n-1$ 个变色龙要么吃的红球比蓝球多一个（称这种情况为浪费），要么相等。设有 A 个红球，B 个蓝球，显然有 $A \geq B, A \geq n$ 。
我们只关心第 n 个变色龙最终的颜色。显然，如果它吃的红球个数大于蓝球那么一定合法；否则在相等的时候最后吃的球必须是蓝球。
考虑浪费的球数 m 。

AGC021E Ball Eat Chameleons

如果 $A - m \neq B$ 那么直接比较大小关系。

AGC021E Ball Eat Chameleons

如果 $A-m\neq B$ 那么直接比较大小关系。

否则，假设 $A\neq B, m\neq 0$ 。如果最后一只变色龙最后吃的球是蓝球，那么吃这个球的时候前 $n-1$ 个变色龙一定还有至少 m 只没有吃过红球。这也说明此时第 n 只变色龙也没有吃红球，所以最终第 n 只变色龙只吃了一个蓝球。这显然是矛盾的，所以最后吃的球一定是红球，即一定不合法。

AGC021E Ball Eat Chameleons

如果 $A-m\neq B$ 那么直接比较大小关系。
否则，假设 $A\neq B, m\neq 0$ 。如果最后一只变色龙最后吃的球是蓝球，那么吃这个球的时候前 $n-1$ 个变色龙一定还有至少 m 只没有吃过红球。这也说明此时第 n 只变色龙也没有吃红球，所以最终第 n 只变色龙只吃了一个蓝球。这显然是矛盾的，所以最后吃的球一定是红球，即一定不合法。
否则就只能是 $A=B, m=0$ 。这种情况稍微特殊一点，最后再讨论。

AGC021E Ball Eat Chameleons

考虑怎么求出 m 。容易想到把红球看做左括号，蓝球看做右括号，匹配出 w 对，就有 $m = \max(n-1-w, 0)$ 。而 w 很容易计算，和最小前缀和什么的有关系。

AGC021E Ball Eat Chameleons

考虑怎么求出 m 。容易想到把红球看做左括号，蓝球看做右括号，匹配出 w 对，就有 $m = \max(n-1-w, 0)$ 。而 w 很容易计算，和最小前缀和什么的有关系。

所以当 $A \neq B$ 时，只需要 $A - B > \max(n-1-w, 0)$ ，即 $A - B > n-1-w$ 。

AGC021E Ball Eat Chameleons

考虑怎么求出 m 。容易想到把红球看做左括号，蓝球看做右括号，匹配出 w 对，就有 $m = \max(n-1-w, 0)$ 。而 w 很容易计算，和最小前缀和什么的有关系。
所以当 $A \neq B$ 时，只需要 $A - B > \max(n-1-w, 0)$ ，即 $A - B > n - 1 - w$ 。
而 $A = B$ 时需要满足什么条件呢？找到前 $n-1$ 对括号，剩下的都喂给第 n 个变色龙。如果第 K 个球是红球那么显然不合法，否则考虑它是否组成了第 $n-1$ 个括号。如果它不是第 $n-1$ 个括号的右括号，即喂给了第 n 个变色龙，那么显然是合法的；否则喂给第 n 个变色龙的球必须一个括号都组不出来，即 'BBB...BRRR..R' ，肯定不合法。

AGC021E Ball Eat Chameleons

考虑怎么求出 m 。容易想到把红球看做左括号，蓝球看做右括号，匹配出 w 对，就有 $m=\max(n-1-w,0)$ 。而 w 很容易计算，和最小前缀和什么的有关系。
所以当 $A\neq B$ 时，只需要 $A-B>\max(n-1-w,0)$ ，即 $A-B>n-1-w$ 。
而 $A=B$ 时需要满足什么条件呢？找到前 $n-1$ 对括号，剩下的都喂给第 n 个变色龙。如果第 K 个球是红球那么显然不合法，否则考虑它是否组成了第 $n-1$ 个括号。如果它不是第 $n-1$ 个括号的右括号，即喂给了第 n 个变色龙，那么显然是合法的；否则喂给第 n 个变色龙的球必须一个括号都组不出来，即 'BBB...BRRR..R' ，肯定不合法。
综上，$A>B$ 时只需要能组出 $n-(A-B)$ 个括号，而 $A=B$ 时需要最后一个球是蓝球，并且前 $K-1$ 个球能组出至少 $n-1$ 个括号。

AGC021E Ball Eat Chameleons

考虑怎么求出 m 。容易想到把红球看做左括号，蓝球看做右括号，匹配出 w 对，就有 $m = \max(n-1-w, 0)$ 。而 w 很容易计算，和最小前缀和什么的有关系。
所以当 $A \neq B$ 时，只需要 $A - B > \max(n-1-w, 0)$ ，即 $A - B > n - 1 - w$ 。
而 $A = B$ 时需要满足什么条件呢？找到前 $n-1$ 对括号，剩下的都喂给第 n 个变色龙。如果第 K 个球是红球那么显然不合法，否则考虑它是否组成了第 $n-1$ 个括号。如果它不是第 $n-1$ 个括号的右括号，即喂给了第 n 个变色龙，那么显然是合法的；否则喂给第 n 个变色龙的球必须一个括号都组不出来，即 'BBB...BRRR..R' ，肯定不合法。
综上，$A > B$ 时只需要能组出 $n - (A - B)$ 个括号，而 $A = B$ 时需要最后一个球是蓝球，并且前 $K-1$ 个球能组出至少 $n-1$ 个括号。
这东西用翻折法随便搞搞即可。

Thanks!