dp

demerzel

2019 年 1 月 11 日

概念

dp 是通过把原问题分解为相对简单的子问题的方式求解复杂问题的方法。

dp 常常适用于有重叠子问题和最优子结构性质的问题。

基本结构

1. 状态。用于描述一个子问题，通常来说，状态的定义越简洁越好。
2. 转移。转移用于描述状态之间的关系。

总结

OI 三十五年文化浓缩起来就是动规文化，动规文化就是状态转移精神，学懂了状态转移就学懂了 OI。

序列 dp 的子问题一般来说表现为一段序列的形式，比如 $f(i,j)$ 表示第 i 个元素到第 j 个元素等等。

这属于最常见的一类 dp，状态表示和转移方程一般很容易看出来（假的

题意

有一个长度为 n 的字符串 s。定义一种划分为 k 个字符串构成的序列 $\{t_k\}$，满足:

- 对于 $i > 1$，t_i 是 t_{i-1} 的子串。
- $s = u_1 t_1 u_2 t_2 \cdots u_k t_k u_{k+1}$，其中 u_i 是任意字符串，可以为空。

求所有划分中 k 最大是多少,$n \leq 5 \cdot 10^5$。

分析

- 首先可以得到一个结论，即 t 中字符串的长度一定是连续的。

分析

- 首先可以得到一个结论，即 t 中字符串的长度一定是连续的。
- 显然如果不连续，那么可以把较长的那个字符串删掉一部分。

分析

- 首先可以得到一个结论，即 t 中字符串的长度一定是连续的。
- 显然如果不连续，那么可以把较长的那个字符串删掉一部分。
- 那么根据这个结论，t_k 必定是一个字符，t_i 必定由 t_{i+1} 首或尾加上一个字符得到。

分析

- 首先可以得到一个结论，即 t 中字符串的长度一定是连续的。
- 显然如果不连续，那么可以把较长的那个字符串删掉一部分。
- 那么根据这个结论，t_k 必定是一个字符，t_i 必定由 t_{i+1} 首或尾加上一个字符得到。
- 并且发现只要记录 t_1 的位置和长度就可以表示出一种划分了。

根号做法

- 由于 k 的大小只有根号级别，那么设状态 $f(i,j)$ 表示是否存在一种划分满足从位置 i 开始往后 j 个字符是其 t_1。

根号做法

- 由于 k 的大小只有根号级别，那么设状态 $f(i,j)$ 表示是否存在一种划分满足从位置 i 开始往后 j 个字符是其 t_1。
- 转移也很简单（下面用 $s(i,j)$ 表示 s 第 i 个字符到第 j 个字符形成的字符串）

根号做法

- 由于 k 的大小只有根号级别，那么设状态 $f(i,j)$ 表示是否存在一种划分满足从位置 i 开始往后 j 个字符是其 t_1。
- 转移也很简单（下面用 $s(i,j)$ 表示 s 第 i 个字符到第 j 个字符形成的字符串）
- 枚举 $k \geq i+j$，若 $f(k,j-1)=1$，并且 $s(i,i+j-2)=s(k,k+j-2)$ 或 $s(i+1,i+j-1)=s(k,k+j-2)$，那么 $f(i,j)=1$。

根号做法

- 由于 k 的大小只有根号级别，那么设状态 $f(i,j)$ 表示是否存在一种划分满足从位置 i 开始往后 j 个字符是其 t_1。
- 转移也很简单（下面用 $s(i,j)$ 表示 s 第 i 个字符到第 j 个字符形成的字符串）
- 枚举 $k \geq i+j$，若 $f(k,j-1)=1$，并且 $s(i,i+j-2)=s(k,k+j-2)$ 或 $s(i+1,i+j-1)=s(k,k+j-2)$，那么 $f(i,j)=1$。
- 于是只要把每个 $f(i,j)=1$ 对应的串存到哈希表里，就可以根号做了。

根号做法

- 由于 k 的大小只有根号级别，那么设状态 $f(i,j)$ 表示是否存在一种划分满足从位置 i 开始往后 j 个字符是其 t_1。
- 转移也很简单（下面用 $s(i,j)$ 表示 s 第 i 个字符到第 j 个字符形成的字符串）
- 枚举 $k \geq i+j$，若 $f(k,j-1)=1$，并且 $s(i,i+j-2)=s(k,k+j-2)$ 或 $s(i+1,i+j-1)=s(k,k+j-2)$，那么 $f(i,j)=1$。
- 于是只要把每个 $f(i,j)=1$ 对应的串存到哈希表里，就可以根号做了。
- 加上一些奇怪的优化，跑得飞快。

优化

- 注意到若 $f(i,j)=1$ 则必有 $f(i,j-1)=1$，因此可以把状态简化为 $dp(i)$ 表示最大的 j $f(i,j)=1$。

优化

- 注意到若 $f(i,j)=1$ 则必有 $f(i,j-1)=1$，因此可以把状态简化为 $dp(i)$ 表示最大的 j $f(i,j)=1$。
- 因此可以通过二分来算 $dp(i)$，只要能够快速判断就行了。

优化

- 注意到若 $f(i,j)=1$ 则必有 $f(i,j-1)=1$，因此可以把状态简化为 $dp(i)$ 表示最大的 j $f(i,j)=1$。
- 因此可以通过二分来算 $dp(i)$，只要能够快速判断就行了。
- 要判断从后缀 i 开始的 x 个字符是否可行，只要找到所有的后缀 $j \geq i+x$ 满足 $lcp(i,j) \geq x-1$ 或 $lcp(i+1,j) \geq x-1$，判断这些 j 中是否存在 $dp(j) \geq x-1$，若存在则是可行的。

优化

- 注意到若 $f(i,j)=1$ 则必有 $f(i,j-1)=1$，因此可以把状态简化为 $dp(i)$ 表示最大的 j $f(i,j)=1$。
- 因此可以通过二分来算 $dp(i)$，只要能够快速判断就行了。
- 要判断从后缀 i 开始的 x 个字符是否可行，只要找到所有的后缀 $j \geq i+x$ 满足 $lcp(i,j) \geq x-1$ 或 $lcp(i+1,j) \geq x-1$，判断这些 j 中是否存在 $dp(j) \geq x-1$，若存在则是可行的。
- 观察条件，第一个限制可以用主席树来维护，第二个限制在后缀数组上是一段区间，第三个限制只要求 $dp(j)$ 的最大值就行了。

小优化

- ~~因为我写的两个 log 跑不过~~继续进行优化。

小优化

- ~~因为我写的两个 log 跑不过~~继续进行优化。
- 可以发现 $dp(i+1) \geq dp(i) - 1$，即 $dp(i) \leq dp(i+1) + 1$。

小优化

- ~~因为我写的两个 log 跑不过~~继续进行优化。
- 可以发现 $dp(i+1) \geq dp(i) - 1$，即 $dp(i) \leq dp(i+1) + 1$。
- 也就是说 dp 值的总变化量是 $\leq n$ 的，于是我们不需要二分了，直接反复暴力检查当前的 $dp(i)$ 是否合法就行了。

题意

有一个长度为 $2n$ 的排列 A，其中一些位置已经确定了，另一些位置上是 -1，即没有确定。

现在可以以任意方式补全这个排列，之后定义一个长度为 n 的序列 B 满足 $B_i = min(A_{2i}, A_{2i-1})$。

计算一共有多少种不同的序列 B，$n \leq 300$。

分析

- 先分析所有位置都未确定的情况。方便起见，先假设 B 序列是无序的。

分析

- 先分析所有位置都未确定的情况。方便起见，先假设 B 序列是无序的。
- 把每对数中较小值看作左括号，较大值看作右括号，那么此时的答案就是卡特兰数 C_n，再乘上 $n!$ 就是原答案。

分析

- 先分析所有位置都未确定的情况。方便起见，先假设 B 序列是无序的。
- 把每对数中较小值看作左括号，较大值看作右括号，那么此时的答案就是卡特兰数 C_n，再乘上 $n!$ 就是原答案。
- 这启示我们把问题转化成括号序列。每对数看作一对匹配的括号。

分析

- 先分析所有位置都未确定的情况。方便起见，先假设 B 序列是无序的。
- 把每对数中较小值看作左括号，较大值看作右括号，那么此时的答案就是卡特兰数 C_n，再乘上 $n!$ 就是原答案。
- 这启示我们把问题转化成括号序列。每对数看作一对匹配的括号。
- 那么原序列中两个数都不是 -1 的，可以忽略掉。一个数是 -1 另一个不是的，定义这种数为特殊括号。其他的是普通括号。

分析

- 先分析所有位置都未确定的情况。方便起见，先假设 B 序列是无序的。
- 把每对数中较小值看作左括号，较大值看作右括号，那么此时的答案就是卡特兰数 C_n，再乘上 $n!$ 就是原答案。
- 这启示我们把问题转化成括号序列。每对数看作一对匹配的括号。
- 那么原序列中两个数都不是 -1 的，可以忽略掉。一个数是 -1 另一个不是的，定义这种数为特殊括号。其他的是普通括号。
- 显然，两个特殊括号之间是不能够匹配的，特殊括号只能和普通括号匹配，而普通括号可以随意匹配。

进一步分析

- 先考虑一个暴力，$\binom{2n}{n}$ 枚举左括号是哪些，剩下的就是右括号。再计算这个括号序列对应的 B 序列有多少。

进一步分析

- 先考虑一个暴力，$\binom{2n}{n}$ 枚举左括号是哪些，剩下的就是右括号。再计算这个括号序列对应的 B 序列有多少。
- 首先对于普通左括号和普通右括号的匹配，暂且不给它们分配位置，最后乘一个阶乘就行。

进一步分析

- 先考虑一个暴力，$\binom{2n}{n}$ 枚举左括号是哪些，剩下的就是右括号。再计算这个括号序列对应的 B 序列有多少。
- 首先对于普通左括号和普通右括号的匹配，暂且不给它们分配位置，最后乘一个阶乘就行。
- 可以从后往前扫描，记录当前剩余的普通右括号和特殊右括号的个数，遇到特殊左括号的时候直接匹配普通右括号。

进一步分析

- 先考虑一个暴力，$\binom{2n}{n}$ 枚举左括号是哪些，剩下的就是右括号。再计算这个括号序列对应的 B 序列有多少。
- 首先对于普通左括号和普通右括号的匹配，暂且不给它们分配位置，最后乘一个阶乘就行。
- 可以从后往前扫描，记录当前剩余的普通右括号和特殊右括号的个数，遇到特殊左括号的时候直接匹配普通右括号。
- 遇到普通左括号的时候有两种选择，即匹配普通右括号或特殊右括号，两种情况都要考虑。

进一步分析

- 先考虑一个暴力，$\binom{2n}{n}$ 枚举左括号是哪些，剩下的就是右括号。再计算这个括号序列对应的 B 序列有多少。
- 首先对于普通左括号和普通右括号的匹配，暂且不给它们分配位置，最后乘一个阶乘就行。
- 可以从后往前扫描，记录当前剩余的普通右括号和特殊右括号的个数，遇到特殊左括号的时候直接匹配普通右括号。
- 遇到普通左括号的时候有两种选择，即匹配普通右括号或特殊右括号，两种情况都要考虑。
- 并且当一个普通左括号匹配到特殊右括号的时候，因为特殊右括号的位置是确定的，所以方案数要乘上特殊右括号的数量，表示这个普通左括号可以放在任意特殊右括号的位置。

解法

- 解法已经很明了了，只需设 $f(i,j,k)$ 为考虑完 i 到 $2n$ 的括号，还剩下 j 个普通右括号和 k 个特殊右括号的方案数。

解法

- 解法已经很明了了，只需设 $f(i,j,k)$ 为考虑完 i 到 $2n$ 的括号，还剩下 j 个普通右括号和 k 个特殊右括号的方案数。
- 若第 $i-1$ 个括号是普通括号，则有转移
  - $f(i,j,k) \to f(i-1,j+1,k)$
  - $f(i,j,k) \to f(i-1,j-1,k)$
  - $f(i,j,k) \cdot k \to f(i-1,j,k-1)$

解法

- 解法已经很明了了，只需设 $f(i,j,k)$ 为考虑完 i 到 $2n$ 的括号，还剩下 j 个普通右括号和 k 个特殊右括号的方案数。
- 若第 $i-1$ 个括号是普通括号，则有转移
  - $f(i,j,k) \to f(i-1,j+1,k)$
  - $f(i,j,k) \to f(i-1,j-1,k)$
  - $f(i,j,k) \cdot k \to f(i-1,j,k-1)$
- 若是特殊括号，则
  - $f(i,j,k) \to f(i-1,j-1,k)$
  - $f(i,j,k) \to f(i-1,j,k+1)$

解法

- 解法已经很明了了，只需设 $f(i,j,k)$ 为考虑完 i 到 $2n$ 的括号，还剩下 j 个普通右括号和 k 个特殊右括号的方案数。
- 若第 $i-1$ 个括号是普通括号，则有转移
  - $f(i,j,k) \to f(i-1,j+1,k)$
  - $f(i,j,k) \to f(i-1,j-1,k)$
  - $f(i,j,k) \cdot k \to f(i-1,j,k-1)$
- 若是特殊括号，则
  - $f(i,j,k) \to f(i-1,j-1,k)$
  - $f(i,j,k) \to f(i-1,j,k+1)$
- 最后答案是 $f(1,0,0)$ 乘上一个阶乘。

题意

有一个 01 序列，初始时序列为空。你可以对序列进行两种操作：

1. 在序列末端插入一个 0。
2. 在序列中删去一个子序列，并在序列末端插入一个 1。这里对子序列的选取有一定限制，设子序列中包含 x 个 0，y 个 1，则你选取的子序列必须满足：
   - 子序列不可为空，即 $x+y>0$
   - 当 $y>0$ 时，$x\in A$，这里 A 为给定集合
   - 当 $y=0$ 时，$x\in B$，这里 B 为给定集合

现在，你需要对序列执行 n 次操作（$n\leq 10^5$）。请你求出在所有不同的操作方案中，最终序列长度为 1 的方案有多少种。两种操作方案被视为不同，当且仅当某一次操作的种类不同，或某个第二类操作中选取的子序列不同（子序列不同指的是位置不同，与值无关）。

答案对 998244353 取模。

分析

- 每次操作都会加入一个元素，把第 i 次操作加入的元素叫做 s_i，并且当且仅当加入 1 的时候会删除元素。

分析

- 每次操作都会加入一个元素，把第 i 次操作加入的元素叫做 s_i，并且当且仅当加入 1 的时候会删除元素。
- 当加入 s_i 的时候，把这次操作中删除的 s_j 都认为是 s_i 的儿子。

分析

- 每次操作都会加入一个元素，把第 i 次操作加入的元素叫做 s_i，并且当且仅当加入 1 的时候会删除元素。
- 当加入 s_i 的时候，把这次操作中删除的 s_j 都认为是 s_i 的儿子。
- 那么所有操作实际上使序列中的元素形成了一个森林结构。0 是叶子节点，1 是非叶节点，并且对非叶节点的孩子中有几个叶子节点有一定的限制。

分析

- 每次操作都会加入一个元素，把第 i 次操作加入的元素叫做 s_i，并且当且仅当加入 1 的时候会删除元素。
- 当加入 s_i 的时候，把这次操作中删除的 s_j 都认为是 s_i 的儿子。
- 那么所有操作实际上使序列中的元素形成了一个森林结构。0 是叶子节点，1 是非叶节点，并且对非叶节点的孩子中有几个叶子节点有一定的限制。
- 由于最后要求序列长度为 1，那么森林中只有一棵树。

解法

- 有了这些就可以 dp 了，设 $f(n)$ 表示大小为 n 的树的方案数，$g(n)$ 表示大小为 n 的森林的方案数。

解法

- 有了这些就可以 dp 了，设 $f(n)$ 表示大小为 n 的树的方案数，$g(n)$ 表示大小为 n 的森林的方案数。
- 有初值 $f(0) = g(0) = 0$。有转移

$$f(n) = \sum_{i \in A} \binom{n-1}{i} g(n-i-1) + [n-1 \in B]$$

$$g(n) = \sum_{i} \binom{n-1}{i-1} f(i) g(n-i)$$

解法

- 有了这些就可以 dp 了，设 $f(n)$ 表示大小为 n 的树的方案数，$g(n)$ 表示大小为 n 的森林的方案数。
- 有初值 $f(0) = g(0) = 0$。有转移

$$f(n) = \sum_{i \in A} \binom{n-1}{i} g(n-i-1) + [n-1 \in B]$$

$$g(n) = \sum_{i} \binom{n-1}{i-1} f(i) g(n-i)$$

- 两个转移都是卷积的形式，因此可以分治 FFT 来做。

树形 dp 的子问题结构形成了一棵树。

这属于最常见的一类 dp，状态表示和转移方程一般很容易看出来（假的

题意

定义森林的秩为其邻接矩阵的秩。

有一棵 n 个点的树，每条边可以选择删或不删，请计算所有 2^{n-1} 种方案形成的森林的秩之和。

$n \leq 5 \cdot 10^5$

线代小知识

- 众所周知，矩阵的 k 阶余子式是指从矩阵中选出 k 行 k 列后所产生的 k^2 个交叉点形成的方形矩阵的行列式。

线代小知识

- 众所周知，矩阵的 k 阶余子式是指从矩阵中选出 k 行 k 列后所产生的 k^2 个交叉点形成的方形矩阵的行列式。
- 众所周知，矩阵的秩等于极大线性无关组的大小，也等于阶数最大的非零余子式的阶数。

线代小知识

- 众所周知，矩阵的 k 阶余子式是指从矩阵中选出 k 行 k 列后所产生的 k^2 个交叉点形成的方形矩阵的行列式。
- 众所周知，矩阵的秩等于极大线性无关组的大小，也等于阶数最大的非零余子式的阶数。
- 众所周知，对称矩阵满足所有 i, j 有 $A_{i,j} = A_{j,i}$，对称余子式是指选取的行集合和列集合相等的余子式。

线代小知识

- 众所周知，矩阵的 k 阶余子式是指从矩阵中选出 k 行 k 列后所产生的 k^2 个交叉点形成的方形矩阵的行列式。
- 众所周知，矩阵的秩等于极大线性无关组的大小，也等于阶数最大的非零余子式的阶数。
- 众所周知，对称矩阵满足所有 i, j 有 $A_{i,j} = A_{j,i}$，对称余子式是指选取的行集合和列集合相等的余子式。
- 于是有定理，对称矩阵的秩等于阶数最大的非零对称余子式的阶数。

证明

- 首先任何非零余子式的阶数都是小于等于矩阵的秩的，所以只需找出一个阶等于矩阵的秩的对意称余子式即可。

证明

- 首先任何非零余子式的阶数都是小于等于矩阵的秩的，所以只需找出一个阶等于矩阵的秩的对意称余子式即可。
- 设对称矩阵 A 的秩为 k。设行集合 I 为矩阵 A 中的极大线性无关组，不失一般性，可以令 $I = \{1, 2, 3, \cdots, k\}$。

证明

- 首先任何非零余子式的阶数都是小于等于矩阵的秩的，所以只需找出一个阶等于矩阵的秩的对意称余子式即可。
- 设对称矩阵 A 的秩为 k。设行集合 I 为矩阵 A 中的极大线性无关组，不失一般性，可以令 $I = \{1, 2, 3, \cdots, k\}$。
- 那么子矩阵 $A_{I,*}$ 的列秩等于行秩等于 k。

证明

- 首先任何非零余子式的阶数都是小于等于矩阵的秩的，所以只需找出一个阶等于矩阵的秩的对意称余子式即可。
- 设对称矩阵 A 的秩为 k。设行集合 I 为矩阵 A 中的极大线性无关组，不失一般性，可以令 $I = \{1, 2, 3, \cdots, k\}$。
- 那么子矩阵 $A_{I,*}$ 的列秩等于行秩等于 k。
- 由于对称性，列集合 I 也是一个极大线性无关组。

证明

- 首先任何非零余子式的阶数都是小于等于矩阵的秩的，所以只需找出一个阶等于矩阵的秩的对意称余子式即可。
- 设对称矩阵 A 的秩为 k。设行集合 I 为矩阵 A 中的极大线性无关组，不失一般性，可以令 $I = \{1, 2, 3, \cdots, k\}$。
- 那么子矩阵 $A_{I,*}$ 的列秩等于行秩等于 k。
- 由于对称性，列集合 I 也是一个极大线性无关组。
- 所以所有 $i > k$ 的列向量 $A_{I,i}$ 都可以由所有 $j \leq k$ 的列向量 $A_{I,j}$ 线性组合得到。

证明

- 首先任何非零余子式的阶数都是小于等于矩阵的秩的，所以只需找出一个阶等于矩阵的秩的对意称余子式即可。
- 设对称矩阵 A 的秩为 k。设行集合 I 为矩阵 A 中的极大线性无关组，不失一般性，可以令 $I = \{1, 2, 3, \cdots, k\}$。
- 那么子矩阵 $A_{I,*}$ 的列秩等于行秩等于 k。
- 由于对称性，列集合 I 也是一个极大线性无关组。
- 所以所有 $i > k$ 的列向量 $A_{I,i}$ 都可以由所有 $j \leq k$ 的列向量 $A_{I,j}$ 线性组合得到。
- 又因为子矩阵 $A_{I,*}$ 的列秩为 k，可知所有 $j \leq k$ 的列向量 $A_{I,j}$ 是线性无关的。

证明

- 首先任何非零余子式的阶数都是小于等于矩阵的秩的，所以只需找出一个阶等于矩阵的秩的对意称余子式即可。
- 设对称矩阵 A 的秩为 k。设行集合 I 为矩阵 A 中的极大线性无关组，不失一般性，可以令 $I = \{1, 2, 3, \cdots, k\}$。
- 那么子矩阵 $A_{I,*}$ 的列秩等于行秩等于 k。
- 由于对称性，列集合 I 也是一个极大线性无关组。
- 所以所有 $i > k$ 的列向量 $A_{I,i}$ 都可以由所有 $j \leq k$ 的列向量 $A_{I,j}$ 线性组合得到。
- 又因为子矩阵 $A_{I,*}$ 的列秩为 k，可知所有 $j \leq k$ 的列向量 $A_{I,j}$ 是线性无关的。
- 那么子矩阵 $A_{I,I}$ 是满秩的，即行列式非零。那么定理得证。

分析

- 这个定理的好处是，无向图邻接矩阵的对称余子式可对应到其导出子图的邻接矩阵。而无向图的邻接矩阵显然是对称的。

分析

- 这个定理的好处是，无向图邻接矩阵的对称余子式可对应到其导出子图的邻接矩阵。而无向图的邻接矩阵显然是对称的。
- 先考虑什么情况下森林的邻接矩阵是满秩的。满秩等价于行列式非零。

分析

- 这个定理的好处是，无向图邻接矩阵的对称余子式可对应到其导出子图的邻接矩阵。而无向图的邻接矩阵显然是对称的。
- 先考虑什么情况下森林的邻接矩阵是满秩的。满秩等价于行列式非零。
- $det(A) = \sum_{\pi} \prod_{i=1}^{n} A_{i,\pi_i} \cdot sgn(\pi)$。

分析

- 这个定理的好处是，无向图邻接矩阵的对称余子式可对应到其导出子图的邻接矩阵。而无向图的邻接矩阵显然是对称的。
- 先考虑什么情况下森林的邻接矩阵是满秩的。满秩等价于行列式非零。
- $det(A) = \sum_{\pi} \prod_{i=1}^{n} A_{i,\pi_i} \cdot sgn(\pi)$。
- 若将 i 向 π_i 连边，那么会形成若干个环。然而森林中只存在二元环。所以当且仅当 π 构成了一个完美匹配的时候乘积不为零。显然森林的完美匹配是唯一的，因此可以得出结论：森林的邻接矩阵满秩当且仅当森林存在完美匹配。

分析

- 这个定理的好处是，无向图邻接矩阵的对称余子式可对应到其导出子图的邻接矩阵。而无向图的邻接矩阵显然是对称的。
- 先考虑什么情况下森林的邻接矩阵是满秩的。满秩等价于行列式非零。
- $det(A) = \sum_{\pi} \prod_{i=1}^{n} A_{i,\pi_i} \cdot sgn(\pi)$。
- 若将 i 向 π_i 连边，那么会形成若干个环。然而森林中只存在二元环。所以当且仅当 π 构成了一个完美匹配的时候乘积不为零。显然森林的完美匹配是唯一的，因此可以得出结论：森林的邻接矩阵满秩当且仅当森林存在完美匹配。
- 那么不存在完美匹配的情况，根据上面的定理，此时矩阵的秩等于最大的存在完美匹配的导出子图的大小。

分析

- 这个定理的好处是，无向图邻接矩阵的对称余子式可对应到其导出子图的邻接矩阵。而无向图的邻接矩阵显然是对称的。
- 先考虑什么情况下森林的邻接矩阵是满秩的。满秩等价于行列式非零。
- $det(A) = \sum_{\pi} \prod_{i=1}^{n} A_{i,\pi_i} \cdot sgn(\pi)$。
- 若将 i 向 π_i 连边，那么会形成若干个环。然而森林中只存在二元环。所以当且仅当 π 构成了一个完美匹配的时候乘积不为零。显然森林的完美匹配是唯一的，因此可以得出结论：森林的邻接矩阵满秩当且仅当森林存在完美匹配。
- 那么不存在完美匹配的情况，根据上面的定理，此时矩阵的秩等于最大的存在完美匹配的导出子图的大小。
- 稍有常识的人都能看出，这就等于森林的最大匹配数 $\times 2$。

解法

- 对证明没有兴趣的同学，只需知道结论：森林的秩等于森林的最大匹配数 $\times 2$。

解法

- 对证明没有兴趣的同学，只需知道结论：森林的秩等于森林的最大匹配数 $\times 2$。
- 于是设 $f(i,0/1)$ 表示以 i 为根的子树内，根节点是否被匹配了的所有方案的最大匹配之和，$g(i,0/1)$ 表示方案数。

解法

- 对证明没有兴趣的同学，只需知道结论：森林的秩等于森林的最大匹配数 $\times 2$。
- 于是设 $f(i,0/1)$ 表示以 i 为根的子树内，根节点是否被匹配了的所有方案的最大匹配之和，$g(i,0/1)$ 表示方案数。
- 合并子树时只要枚举这条边是否要断开就可以了，不赘述。

题意

有一棵 n 个点的树，求这棵树的点分方案数。如果两种方案的点分树是相同的那么认为这两种方案相同。

对 1000000007 取模，$n \leq 2000$

分析

- 似乎无从下手，还是尝试子树 dp 吧。那么需要解决如何合并子树的问题。

分析

- 似乎无从下手，还是尝试子树 dp 吧。那么需要解决如何合并子树的问题。
- 将子树 q 合并到其父亲 p 时，要合并两棵树内的点分方案。

分析

- 似乎无从下手，还是尝试子树 dp 吧。那么需要解决如何合并子树的问题。
- 将子树 q 合并到其父亲 p 时，要合并两棵树内的点分方案。
- 由于 p 和 q 是相邻的，因此一旦某次分治时选择了 p 或 q 作为分治中心，之后这两棵树就分开了。

分析

- 似乎无从下手，还是尝试子树 dp 吧。那么需要解决如何合并子树的问题。
- 将子树 q 合并到其父亲 p 时，要合并两棵树内的点分方案。
- 由于 p 和 q 是相邻的，因此一旦某次分治时选择了 p 或 q 作为分治中心，之后这两棵树就分开了。
- 在点分树上考虑，当 p 或 q 的某个祖先 anc 被分治掉时，anc 的不包含 p 或 q 的那些子树的分治方案也确定了。

分析

- 似乎无从下手，还是尝试子树 dp 吧。那么需要解决如何合并子树的问题。
- 将子树 q 合并到其父亲 p 时，要合并两棵树内的点分方案。
- 由于 p 和 q 是相邻的，因此一旦某次分治时选择了 p 或 q 作为分治中心，之后这两棵树就分开了。
- 在点分树上考虑，当 p 或 q 的某个祖先 anc 被分治掉时，anc 的不包含 p 或 q 的那些子树的分治方案也确定了。
- 因此只需要把 p 和 q 的所有祖先按照任意顺序归并，就可以合并这两种点分方案了。

解法

- 根据前面的分析，合并方案时只跟 p 或 q 在点分树上的深度有关。

解法

- 根据前面的分析，合并方案时只跟 p 或 q 在点分树上的深度有关。
- 设 $f(p,k)$ 表示以 p 为根的子树，p 在点分树上深度为 k 的点分方案数。

解法

- 根据前面的分析，合并方案时只跟 p 或 q 在点分树上的深度有关。
- 设 $f(p,k)$ 表示以 p 为根的子树，p 在点分树上深度为 k 的点分方案数。
- 转移:

$$f(p,i)\cdot f(q,j)\cdot\binom{k-1}{i-1}\to f'(p,k)\quad(k\le i+j)$$

f' 表示合并后新的 f 值。

解法

- 根据前面的分析，合并方案时只跟 p 或 q 在点分树上的深度有关。
- 设 $f(p,k)$ 表示以 p 为根的子树，p 在点分树上深度为 k 的点分方案数。
- 转移:

$$f(p,i)\cdot f(q,j)\cdot\binom{k-1}{i-1}\to f'(p,k)\quad(k\le i+j)$$

  f' 表示合并后新的 f 值。
- 通过后缀和等优化就可以做到 n^2。

题意

一个序列长度为 n，每个元素都是 1 到 m 内的整数，且所有 1 到 m 内的整数出现过，则称这是一个挺好序列。

对于一个序列 A，记 $f_A(l,r)$ 为 A 的第 l 个到第 r 个数中最大值的下标（如果有多个最大值，取下标最小的）。

两个序列 A 和 B 同构，当且仅当 A 和 B 长度相等，且对于任意 $i \le j$，均有 $f_A(i,j) = f_B(i,j)$。

给出 $n, m \le 10^5$，求有多少种不同构的挺好序列。答案对 998244353 取模。

分析

- 可以看出两个序列同构当且仅当它们的笛卡尔树相同。这里的笛卡尔树满足左儿子 $<$ 父亲，右儿子 $\leq$ 父亲。

分析

- 可以看出两个序列同构当且仅当它们的笛卡尔树相同。这里的笛卡尔树满足左儿子 $<$ 父亲，右儿子 $\leq$ 父亲。
- 对于第二个条件，首先必有 $n \leq m$。定义笛卡尔树的深度为从根节点出发路径上最多的左儿子数 $+1$。那么这个条件等价于树的深度 $\leq m$，因为只要满足这个条件就可以构造出一种方案。

分析

- 可以看出两个序列同构当且仅当它们的笛卡尔树相同。这里的笛卡尔树满足左儿子 $<$ 父亲，右儿子 $\leq$ 父亲。
- 对于第二个条件，首先必有 $n \leq m$。定义笛卡尔树的深度为从根节点出发路径上最多的左儿子数 $+1$。那么这个条件等价于树的深度 $\leq m$，因为只要满足这个条件就可以构造出一种方案。
- 构造方法如下：将所有点和其右儿子视为同一层，这样会形成不超过 m 个层，从最上面的层开始给节点填数字，依次填入 $m, m-1, m-2, \cdots$，当剩余的层数和剩余的数字一样多的时候，就把剩余的层每层全部都填一个数字。这样每个数字都至少出现了一次。

分析

- 可以看出两个序列同构当且仅当它们的笛卡尔树相同。这里的笛卡尔树满足左儿子 $<$ 父亲，右儿子 $\leq$ 父亲。
- 对于第二个条件，首先必有 $n \leq m$。定义笛卡尔树的深度为从根节点出发路径上最多的左儿子数 $+1$。那么这个条件等价于树的深度 $\leq m$，因为只要满足这个条件就可以构造出一种方案。
- 构造方法如下：将所有点和其右儿子视为同一层，这样会形成不超过 m 个层，从最上面的层开始给节点填数字，依次填入 $m, m-1, m-2, \cdots$，当剩余的层数和剩余的数字一样多的时候，就把剩余的层每层全部都填一个数字。这样每个数字都至少出现了一次。
- 有一个基于括号序列的做法，有兴趣的同学可以自行了解，这里只介绍基于 dp 的做法。

解法

- 设 $f(k,n)$ 表示深度不超过 k 的 n 个点的树有多少。转移时枚举左子树大小

$$f(k,n)=\sum_{i=0}^{n-1} f(k-1,i)\cdot f(k,n-1-i)$$

解法

- 设 $f(k,n)$ 表示深度不超过 k 的 n 个点的树有多少。转移时枚举左子树大小

$$f(k,n)=\sum_{i=0}^{n-1} f(k-1,i)\cdot f(k,n-1-i)$$

- 令多项式 $F_k(x)=\sum_i f(k,i)x^i$，根据转移方程有

$$F_k(x)=xF_k(x)F_{k-1}(x)+1$$

$$F_k(x)=\frac{1}{1-xF_{k-1}(x)}$$

解法

- 设 $f(k,n)$ 表示深度不超过 k 的 n 个点的树有多少。转移时枚举左子树大小

$$f(k,n)=\sum_{i=0}^{n-1} f(k-1,i)\cdot f(k,n-1-i)$$

- 令多项式 $F_k(x)=\sum_i f(k,i)x^i$，根据转移方程有

$$F_k(x)=xF_k(x)F_{k-1}(x)+1$$

$$F_k(x)=\frac{1}{1-xF_{k-1}(x)}$$

- 推测 $F_k(x)$ 可以表示成 $\frac{A_k(x)}{B_k(x)}$ 的形式，那么

$$F_k(x)=\frac{1}{1-x\frac{A_{k-1}(x)}{B_{k-1}(x)}}$$

$$=\frac{B_{k-1}(x)}{B_{k-1}(x)-xA_{k-1}(x)}$$

解法

- 把关系写成矩阵的形式

$$\begin{bmatrix} A_k(x) \\ B_k(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -x & 1 \end{bmatrix} \times \begin{bmatrix} A_{k-1}(x) \\ B_{k-1}(x) \end{bmatrix}$$

解法

- 把关系写成矩阵的形式

$$\begin{bmatrix} A_k(x) \\ B_k(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -x & 1 \end{bmatrix} \times \begin{bmatrix} A_{k-1}(x) \\ B_{k-1}(x) \end{bmatrix}$$

- 矩阵乘法即可，但是这样常数很大。

解法

- 把关系写成矩阵的形式

$$\begin{bmatrix} A_k(x) \\ B_k(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -x & 1 \end{bmatrix} \times \begin{bmatrix} A_{k-1}(x) \\ B_{k-1}(x) \end{bmatrix}$$

- 矩阵乘法即可，但是这样常数很大。
- 有一个利用 $IDFT$ 来进行插值的小技巧，对于所有 $0 \le i \le N, (N = 2^r)$，计算出 $A_m(\omega_N^i)$ 和 $B_m(\omega_N^i)$ 的值，最后 $IDFT$ 求出 $A_m(x)$ 和 $B_m(x)$ 即可。

题意

有 $n \times m$ 的网格图。每秒选择一对相邻格子涂黑（已经涂黑的格子可以再次涂黑），涂黑每一对相邻格子的概率是相等的。

在这 $n \times m$ 个格子中，有一些格子是特殊的。期望多少秒后，所有特殊的格子被涂黑。

$n \leq 6, m \leq 100$

解法

- 要求的是所有特殊格子被染黑的时间的最大值，这不太好算，可以通过 $MinMax$ 容斥来将最大值转化为最小值。

$$Max(S) = \sum_{T \subseteq S, T \neq \emptyset} Min(T)(-1)^{|T|+1}$$

解法

- 要求的是所有特殊格子被染黑的时间的最大值，这不太好算，可以通过 $MinMax$ 容斥来将最大值转化为最小值。

$$Max(S) = \sum_{T \subseteq S, T \neq \emptyset} Min(T)(-1)^{|T|+1}$$

- 最小值就很好算了，只需求出 x 代表有 x 对相邻格子中至少存在一个 T 中的格子，那么 $Min(T)$ 的期望就等于 $\frac{n(m-1)+(n-1)m}{x}$

解法

- 要求的是所有特殊格子被染黑的时间的最大值，这不太好算，可以通过 $MinMax$ 容斥来将最大值转化为最小值。

$$Max(S) = \sum_{T \subseteq S, T \neq \emptyset} Min(T)(-1)^{|T|+1}$$

- 最小值就很好算了，只需求出 x 代表有 x 对相邻格子中至少存在一个 T 中的格子，那么 $Min(T)$ 的期望就等于 $\frac{n(m-1)+(n-1)m}{x}$
- 对于每一对相邻格子，我们在其中较右或较下的格子处考虑。

解法

- 要求的是所有特殊格子被染黑的时间的最大值，这不太好算，可以通过 $MinMax$ 容斥来将最大值转化为最小值。

$$Max(S) = \sum_{T \subseteq S, T \neq \emptyset} Min(T)(-1)^{|T|+1}$$

- 最小值就很好算了，只需求出 x 代表有 x 对相邻格子中至少存在一个 T 中的格子，那么 $Min(T)$ 的期望就等于 $\frac{n(m-1)+(n-1)m}{x}$
- 对于每一对相邻格子，我们在其中较右或较下的格子处考虑。
- 设 $f(i,j,k,x)$ 表示考虑到前 i 列的从上往下第 j 个格子，轮廓线上的格子被选择的情况是 k，已经考虑了的相邻格子对有 x 对至少包含一个被选择的格子，所有这样的方案的 $(-1)^{|T|+1}$ 之和。

解法

- 要求的是所有特殊格子被染黑的时间的最大值，这不太好算，可以通过 $MinMax$ 容斥来将最大值转化为最小值。

$$Max(S) = \sum_{T \subseteq S, T \neq \emptyset} Min(T)(-1)^{|T|+1}$$

- 最小值就很好算了，只需求出 x 代表有 x 对相邻格子中至少存在一个 T 中的格子，那么 $Min(T)$ 的期望就等于 $\frac{n(m-1)+(n-1)m}{x}$
- 对于每一对相邻格子，我们在其中较右或较下的格子处考虑。
- 设 $f(i,j,k,x)$ 表示考虑到前 i 列的从上往下第 j 个格子，轮廓线上的格子被选择的情况是 k，已经考虑了的相邻格子对有 x 对至少包含一个被选择的格子，所有这样的方案的 $(-1)^{|T|+1}$ 之和。
- 转移时，如果 (i,j) 不是特殊格子，那么不能选，否则可以选也可以不选。之后根据 $(i-1,j)(i,j-1)(i,j)$ 三个格子的情况来决定 x 要加上多少。注意如果当前格子选了的话 dp 值要乘 -1。

题意

小 Z 是养鸽子的人。一天，小 Z 给鸽子们喂玉米吃。

一共有 n 只鸽子，小 Z 每秒会等概率选择一只鸽子并给他一粒玉米。一只鸽子饱了当且仅当它吃了的玉米粒数量 $\geq k$。

小 Z 想要你告诉他，期望多少秒之后所有的鸽子都饱了。

$n \leq 1000, k \leq 50$，答案对 998244353 取模。

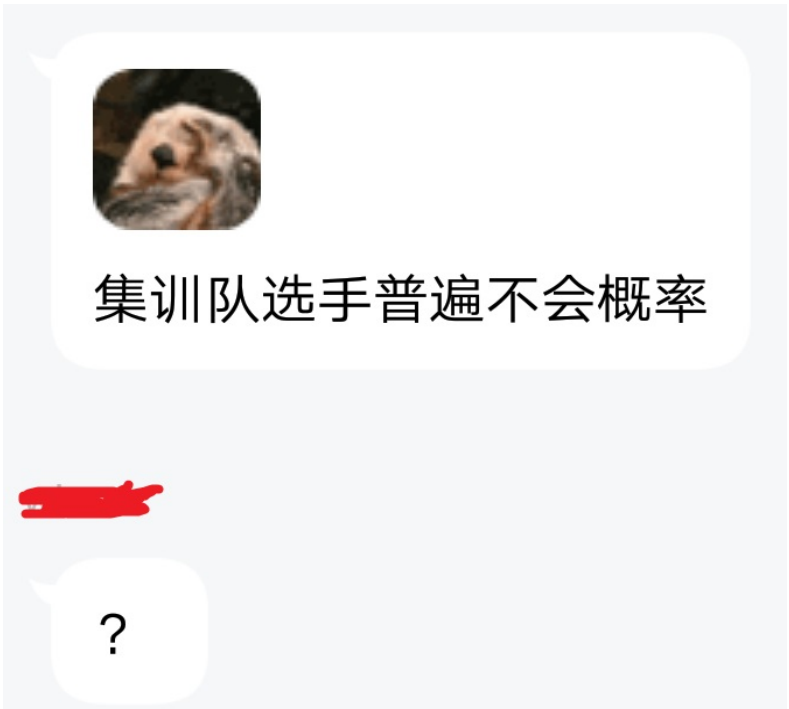
来自出题人的吐槽

这道题是去年全国赛前出的。当时只想到了 $nk^2log(nk)$ 的做法。

圣诞节前一天的集训队作业考到下午两点的时候，我被告知存在 nk^2 的做法。还抱怨这题数据范围太小了 $=\ =$。

然而最后我看榜上 A 掉这题的同学只有一个人写的 nk^2，其他人都写的标算做法。

不过 $nk^2log(nk)$ 和 nk^2 差别不能说太大哈，不算出题事故。



分析

- 先讲我的辣鸡做法。

分析

- 先讲我的辣鸡做法。
- 还是 MinMax 容斥一下，问题变成：在 n 个盒子里随机放球，若前 m 个盒子中有一个盒子的球数达到 k 则结束，问期望。

分析

- 先讲我的辣鸡做法。
- 还是 MinMax 容斥一下，问题变成：在 n 个盒子里随机放球，若前 m 个盒子中有一个盒子的球数达到 k 则结束，问期望。
- 假设某个方案在这 m 个盒子中总共放了 x 个球，那么它对答案的贡献为

$$(\frac{1}{m})^x \cdot E_m(x)$$

其中 $E_m(x)$ 表示在前 m 个盒子中放入 x 个球的期望步数，后面再介绍它怎么算。

分析

- 先讲我的辣鸡做法。
- 还是 MinMax 容斥一下，问题变成：在 n 个盒子里随机放球，若前 m 个盒子中有一个盒子的球数达到 k 则结束，问期望。
- 假设某个方案在这 m 个盒子中总共放了 x 个球，那么它对答案的贡献为

$$(\frac{1}{m})^x \cdot E_m(x)$$

  其中 $E_m(x)$ 表示在前 m 个盒子中放入 x 个球的期望步数，后面再介绍它怎么算。
- 上式的含义是“在前 m 个盒子中放入 x 个球的期望”乘上“放入的 x 个球正好是这个方案的概率”。

分析

- 先讲我的辣鸡做法。
- 还是 MinMax 容斥一下，问题变成：在 n 个盒子里随机放球，若前 m 个盒子中有一个盒子的球数达到 k 则结束，问期望。
- 假设某个方案在这 m 个盒子中总共放了 x 个球，那么它对答案的贡献为

$$(\frac{1}{m})^x \cdot E_m(x)$$

  其中 $E_m(x)$ 表示在前 m 个盒子中放入 x 个球的期望步数，后面再介绍它怎么算。
- 上式的含义是“在前 m 个盒子中放入 x 个球的期望”乘上“放入的 x 个球正好是这个方案的概率”。
- 因此只需统计在 m 个盒子中放入 x 个球的方案数即可。由于最后一步肯定是在一个盒子里放一个球使得该盒子中球数达到 k，因此我们把这个球拿出来特殊考虑。

解法

- 假设放完之后第 i 个盒子里有 a_i 个球，那么其对应的方案数有

$$\frac{(\sum_{i=1}^{m} a_i)!}{\prod_{i=1}^{m} a_i!}$$

解法

- 假设放完之后第 i 个盒子里有 a_i 个球，那么其对应的方案数有

$$\frac{(\sum_{i=1}^{m} a_i)!}{\prod_{i=1}^{m} a_i!}$$

- 对 $\prod_{i=1}^{m} \frac{1}{a_i!}$ 进行 dp 即可。

解法

- 假设放完之后第 i 个盒子里有 a_i 个球，那么其对应的方案数有

$$\frac{(\sum_{i=1}^{m} a_i)!}{\prod_{i=1}^{m} a_i!}$$

- 对 $\prod_{i=1}^{m} \frac{1}{a_i!}$ 进行 dp 即可。
- 设 $f(i,j)$ 表示 i 个盒子，已经有 j 个球了，没有盒子的球数达到 k，所有方案的权值之和。

解法

- 假设放完之后第 i 个盒子里有 a_i 个球，那么其对应的方案数有

$$\frac{(\sum_{i=1}^{m} a_i)!}{\prod_{i=1}^{m} a_i!}$$

- 对 $\prod_{i=1}^{m} \frac{1}{a_i!}$ 进行 dp 即可。
- 设 $f(i,j)$ 表示 i 个盒子，已经有 j 个球了，没有盒子的球数达到 k，所有方案的权值之和。
- 设 $g(i,j)$ 表示 i 个盒子，已经有 j 个球了，有个盒子的球数达到 k，所有方案的权值之和。

解法

- 假设放完之后第 i 个盒子里有 a_i 个球，那么其对应的方案数有

$$\frac{(\sum_{i=1}^{m} a_i)!}{\prod_{i=1}^{m} a_i!}$$

- 对 $\prod_{i=1}^{m} \frac{1}{a_i!}$ 进行 dp 即可。
- 设 $f(i,j)$ 表示 i 个盒子，已经有 j 个球了，没有盒子的球数达到 k，所有方案的权值之和。
- 设 $g(i,j)$ 表示 i 个盒子，已经有 j 个球了，有个盒子的球数达到 k，所有方案的权值之和。
- 转移时枚举下一个盒子里放几个球，$O(n^2k^2)$，用 ntt 优化一下，就能做到 $O(n^2klog(nk))$。
- 那么 m 个盒子的答案就是

$$\sum_{i=0}^{n(k-1)} g(m,i) \cdot i! \cdot (\frac{1}{m})^{i+1} \cdot E_m(i+1)$$

补充证明

- $E_m(k)$ 表示在前 m 个盒子中放入 k 个球的期望步数。显然每次放球都有 $p = \frac{m}{n}$ 的概率放入前 m 个盒子。

补充证明

- $E_m(k)$ 表示在前 m 个盒子中放入 k 个球的期望步数。显然每次放球都有 $p = \frac{m}{n}$ 的概率放入前 m 个盒子。
- 可以看出，实际上要求的就是这个式子。

$$\sum_i \binom{k+i-1}{i} (1-p)^i p^k (k+i)$$

补充证明

- $E_m(k)$ 表示在前 m 个盒子中放入 k 个球的期望步数。显然每次放球都有 $p = \frac{m}{n}$ 的概率放入前 m 个盒子。
- 可以看出，实际上要求的就是这个式子。

$$\sum_i \binom{k+i-1}{i} (1-p)^i p^k (k+i)$$

- 首先

$$\sum_i \binom{k+i-1}{i} (1-p)^i p^k (k+i)$$

$$= k p^k \sum_i \binom{k+i}{i} (1-p)^i$$

补充证明

- 根据二项式定理有:

$$(1-x)^{-r} = \sum_i \binom{r+i-1}{i} x^i$$

令 $x = 1-p, r = k+1$，得到:

$$p^{-k-1} = \sum_i \binom{k+i}{i} (1-p)^i$$

补充证明

- 根据二项式定理有:

$$(1-x)^{-r} = \sum_i \binom{r+i-1}{i} x^i$$

令 $x = 1-p, r = k+1$，得到:

$$p^{-k-1} = \sum_i \binom{k+i}{i} (1-p)^i$$

- 因此:

$$\begin{aligned} & kp^k \sum_i \binom{k+i}{i} (1-p)^i \\ =& kp^k p^{-k-1} \\ =& \frac{k}{p} \end{aligned}$$

高妙做法: 分析

- 称一粒玉米是有效玉米，当且仅当它被投喂给了一只没有饱的鸽子。那么有效玉米序列的长度是固定的 $n \cdot k$。

高妙做法: 分析

- 称一粒玉米是有效玉米，当且仅当它被投喂给了一只没有饱的鸽子。那么有效玉米序列的长度是固定的 $n \cdot k$。
- 现在考虑枚举所有的有效玉米序列，计算对答案的贡献。下面记 r_i 表示投喂第 i 粒有效玉米前已经有多少鸽子饱了。

高妙做法: 分析

- 称一粒玉米是有效玉米，当且仅当它被投喂给了一只没有饱的鸽子。那么有效玉米序列的长度是固定的 $n \cdot k$。
- 现在考虑枚举所有的有效玉米序列，计算对答案的贡献。下面记 r_i 表示投喂第 i 粒有效玉米前已经有多少鸽子饱了。
- 那么一个有效玉米序列的贡献就是

$$(\prod_{i=1}^{nk} P(r_i)) \cdot (\sum_{i=1}^{nk} E(r_i))$$

其中 $P(x) = \frac{1}{n-x}, E(x) = \frac{n}{n-x}$。前一项表示这个序列的概率，后一项表示期望。

高妙做法: 解法

- 直接 dp 似乎不太方便。因为无法确定下一粒玉米投喂后是否会是一只鸽子吃饱。

高妙做法: 解法

- 直接 dp 似乎不太方便。因为无法确定下一粒玉米投喂后是否会是一只鸽子吃饱。
- 注意到贡献只和 r_i 有关，而一只鸽子吃饱前是不会对 r_i 产生影响的。所以可以认为一只鸽子吃饱前其有效玉米都是“白玉米”。我们只在一只鸽子吃饱的时侯把白玉米染色。

高妙做法: 解法

- 直接 dp 似乎不太方便。因为无法确定下一粒玉米投喂后是否会是一只鸽子吃饱。
- 注意到贡献只和 r_i 有关，而一只鸽子吃饱前是不会对 r_i 产生影响的。所以可以认为一只鸽子吃饱前其有效玉米都是“白玉米”。我们只在一只鸽子吃饱的时侯把白玉米染色。
- 这样就可以 dp 了，先强制鸽子吃饱的顺序是 1 到 n，最后乘 $n!$。设 $f(m, c)$ 表示投喂了 m 粒有效玉米，前 c 只鸽子已经饱了的贡献之和。$g(m, c)$ 表示概率之和。

高妙做法: 解法

- 考虑如何转移至 $m+1$，转移后新的贡献之和为:

$$\begin{aligned}
&\sum \prod_{i\le m} P_{r_i} P_{r_{m+1}} (\sum_{i\le m} E_{r_i} + E_{r_{m+1}}) \\
=&P_{r_{m+1}} (\sum \prod_{i\le m} P_{r_i} \sum_{i\le m} E_{r_i}) + P_{r_{m+1}} E_{r_{m+1}} (\sum \prod_{i\le m} P_{r_i}) \\
=&P_{r_{m+1}} f(m,c) + P_{r_{m+1}} E_{r_{m+1}} g(m,c)
\end{aligned}$$

显然 $r_{m+1} = c$。而新的概率之和只要简单地乘个 $P_{r_{m+1}}$ 就行了。

高妙做法: 解法

- 考虑如何转移至 $m+1$，转移后新的贡献之和为:

$$\begin{aligned}
&\sum \prod_{i\le m} P_{r_i} P_{r_{m+1}} (\sum_{i\le m} E_{r_i} + E_{r_{m+1}}) \\
=&P_{r_{m+1}} (\sum \prod_{i\le m} P_{r_i} \sum_{i\le m} E_{r_i}) + P_{r_{m+1}} E_{r_{m+1}} (\sum \prod_{i\le m} P_{r_i}) \\
=&P_{r_{m+1}} f(m,c) + P_{r_{m+1}} E_{r_{m+1}} g(m,c)
\end{aligned}$$

  显然 $r_{m+1} = c$。而新的概率之和只要简单地乘个 $P_{r_{m+1}}$ 就行了。

- 接下来有两种转移，第一种是加入一粒白玉米，这种直接做。另一种是在 $m+1$ 处有一只鸽子吃饱了，这种转移需要乘上 $\binom{m-cn}{k-1}$ 表示给白玉米染色的方案数。最后 $f(nk,n)\cdot n!$ 就是答案。

介绍

- 若多项式 $A(x) = \sum_i a_i x^i$ 满足某个方程 $A(x) = g(A(x))$，且 g 函数中的运算都方便计算，一般都是乘法，积分求导，简单的函数复合等等。那么可以用分治 FFT 来求出 $A(x)$ 的前若干项系数。

介绍

- 若多项式 $A(x)=\sum_i a_i x^i$ 满足某个方程 $A(x)=g(A(x))$，且 g 函数中的运算都方便计算，一般都是乘法，积分求导，简单的函数复合等等。那么可以用分治 FFT 来求出 $A(x)$ 的前若干项系数。
- 主要过程是通过分治区间 $[L,R]$ 来求出 x^L 到 x^R 的系数。分治时，首先递归区间 $[L,mid]$，然后计算 $[L,mid]$ 对 $[mid+1,R]$ 的贡献，最后递归 $[mid+1,R]$。复杂度一般是两个 log。

介绍

- 若多项式 $A(x)=\sum_i a_i x^i$ 满足某个方程 $A(x)=g(A(x))$，且 g 函数中的运算都方便计算，一般都是乘法，积分求导，简单的函数复合等等。那么可以用分治 FFT 来求出 $A(x)$ 的前若干项系数。
- 主要过程是通过分治区间 $[L,R]$ 来求出 x^L 到 x^R 的系数。分治时，首先递归区间 $[L,mid]$，然后计算 $[L,mid]$ 对 $[mid+1,R]$ 的贡献，最后递归 $[mid+1,R]$。复杂度一般是两个 log。
- 与牛顿迭代相比，分治 FFT 显得简单灵活一些。很多情况下可以和牛顿迭代互相代替（虽然复杂度劣一些）。

例子: 多项式求 exp

- 现给出多项式 $A(x)$ 满足常数项为零，求其 exp 前若干项，这是一道牛顿迭代的经典题。

例子: 多项式求 exp

- 现给出多项式 $A(x)$ 满足常数项为零，求其 exp 前若干项，这是一道牛顿迭代的经典题。
- 要求的是 $B(x)$ 满足

$$B(x) = e^{A(x)}$$

$$B'(x) = e^{A(x)} A'(x)$$

$$B(x) = \int (B(x)A'(x))$$

例子: 多项式求 exp

- 现给出多项式 $A(x)$ 满足常数项为零，求其 exp 前若干项，这是一道牛顿迭代的经典题。
- 要求的是 $B(x)$ 满足

$$B(x) = e^{A(x)}$$

$$B'(x) = e^{A(x)} A'(x)$$

$$B(x) = \int (B(x)A'(x))$$

- 首先根据定义 $b_0 = 1$，根据上式可以得到 $n > 1$ 时 $b_n = \frac{1}{n} \sum_{i<n} b_i a'_{n-i-1}$。

例子: 多项式求 exp

- 现给出多项式 $A(x)$ 满足常数项为零，求其 exp 前若干项，这是一道牛顿迭代的经典题。
- 要求的是 $B(x)$ 满足

$$B(x) = e^{A(x)}$$

$$B'(x) = e^{A(x)} A'(x)$$

$$B(x) = \int (B(x)A'(x))$$

- 首先根据定义 $b_0 = 1$，根据上式可以得到 $n > 1$ 时 $b_n = \frac{1}{n} \sum_{i<n} b_i a'_{n-i-1}$。
- 可以先不管外面的 $\frac{1}{n}$，在递归到底层的时候除一下就可以了。

例子: 多项式求 exp

- 现给出多项式 $A(x)$ 满足常数项为零，求其 exp 前若干项，这是一道牛顿迭代的经典题。
- 要求的是 $B(x)$ 满足

$$B(x) = e^{A(x)}$$

$$B'(x) = e^{A(x)} A'(x)$$

$$B(x) = \int (B(x)A'(x))$$

- 首先根据定义 $b_0 = 1$，根据上式可以得到 $n > 1$ 时 $b_n = \frac{1}{n} \sum_{i<n} b_i a'_{n-i-1}$。
- 可以先不管外面的 $\frac{1}{n}$，在递归到底层的时候除一下就可以了。
- 在计算 $[L, mid]$ 对 $[mid+1, R]$ 的贡献时，只需将 $b_L x^L + b_{L+1} x^{L+1} + \cdots + b_{mid} x^{mid}$ 和 $a'_0 + a'_1 x^1 + a'_2 x^2 + \cdots + a'_{R-L} x^{R-L}$ 乘起来再对应地加上去就行了。

例子: 多项式求 exp

- 现给出多项式 $A(x)$ 满足常数项为零，求其 exp 前若干项，这是一道牛顿迭代的经典题。
- 要求的是 $B(x)$ 满足

$$B(x) = e^{A(x)}$$

$$B'(x) = e^{A(x)} A'(x)$$

$$B(x) = \int (B(x)A'(x))$$

- 首先根据定义 $b_0 = 1$，根据上式可以得到 $n > 1$ 时 $b_n = \frac{1}{n} \sum_{i<n} b_i a'_{n-i-1}$。
- 可以先不管外面的 $\frac{1}{n}$，在递归到底层的时候除一下就可以了。
- 在计算 $[L, mid]$ 对 $[mid+1, R]$ 的贡献时，只需将 $b_L x^L + b_{L+1} x^{L+1} + \cdots + b_{mid} x^{mid}$ 和 $a'_0 + a'_1 x^1 + a'_2 x^2 + \cdots + a'_{R-L} x^{R-L}$ 乘起来再对应地加上去就行了。
- 以上便是分治 FFT 的主要内容，更复杂的情况只需在此基础上推广即可。

北大集训: 矩形面积周长

你可以在平面上任意画出长宽都是正整数的矩形。

有 T 次询问，每次询问是否存在一种画矩形的方案满足周长之和为 C，面积之和为 S。

$T, C, S \leq 10^5$

做法

- 设 $f(i,j)$ 表示周长和为 i，面积和为 j 是否可行，总共有 n^2 种状态。

做法

- 设 $f(i,j)$ 表示周长和为 i，面积和为 j 是否可行，总共有 n^2 种状态。
- 每次枚举一个矩形判断，总共有 $nln(n)$ 种转移。

做法

- 设 $f(i,j)$ 表示周长和为 i，面积和为 j 是否可行，总共有 n^2 种状态。
- 每次枚举一个矩形判断，总共有 $nln(n)$ 种转移。
- 复杂度 $n^3log(n)$。

优化一

- 注意到若存在两个矩形 $a \times b$，$a \times c$。那么可以用 $a \times (b + c - 1)$ 和 $1 \times a$ 来等效代替。

优化一

- 注意到若存在两个矩形 $a \times b$，$a \times c$。那么可以用 $a \times (b + c - 1)$ 和 $1 \times a$ 来等效代替。
- 也就是说 $a > 1$ 的矩形最多有一个，$1 \times a$ 其中 $a > 1$ 的矩形也最多只有一个。不过 1×1 的矩形可能有很多。

优化一

- 注意到若存在两个矩形 $a \times b$，$a \times c$。那么可以用 $a \times (b + c - 1)$ 和 $1 \times a$ 来等效代替。
- 也就是说 $a > 1$ 的矩形最多有一个，$1 \times a$ 其中 $a > 1$ 的矩形也最多只有一个。不过 1×1 的矩形可能有很多。
- 那么枚举短边的长度 $w \leq \sqrt{n}$，加入 $a = w$ 的矩形。

优化一

- 注意到若存在两个矩形 $a \times b$，$a \times c$。那么可以用 $a \times (b + c - 1)$ 和 $1 \times a$ 来等效代替。
- 也就是说 $a > 1$ 的矩形最多有一个，$1 \times a$ 其中 $a > 1$ 的矩形也最多只有一个。不过 1×1 的矩形可能有很多。
- 那么枚举短边的长度 $w \leq \sqrt{n}$，加入 $a = w$ 的矩形。
- 加入时，先新定义一个函数 $g(i, j)$ 表示至少存在一个边长为 w 的矩形时是否可行。

优化一

- 注意到若存在两个矩形 $a \times b$，$a \times c$。那么可以用 $a \times (b + c - 1)$ 和 $1 \times a$ 来等效代替。
- 也就是说 $a > 1$ 的矩形最多有一个，$1 \times a$ 其中 $a > 1$ 的矩形也最多只有一个。不过 1×1 的矩形可能有很多。
- 那么枚举短边的长度 $w \leq \sqrt{n}$，加入 $a = w$ 的矩形。
- 加入时，先新定义一个函数 $g(i, j)$ 表示至少存在一个边长为 w 的矩形时是否可行。
- 一开始有转移 $f(i, j) \to g(i + 4w, j + w^2)$。之后有转移 $g(i, j) \to g(i + 2, j + w)$。

优化一

- 注意到若存在两个矩形 $a \times b$，$a \times c$。那么可以用 $a \times (b + c - 1)$ 和 $1 \times a$ 来等效代替。
- 也就是说 $a > 1$ 的矩形最多有一个，$1 \times a$ 其中 $a > 1$ 的矩形也最多只有一个。不过 1×1 的矩形可能有很多。
- 那么枚举短边的长度 $w \leq \sqrt{n}$，加入 $a = w$ 的矩形。
- 加入时，先新定义一个函数 $g(i, j)$ 表示至少存在一个边长为 w 的矩形时是否可行。
- 一开始有转移 $f(i, j) \to g(i + 4w, j + w^2)$。之后有转移 $g(i, j) \to g(i + 2, j + w)$。
- 于是转移复杂度被优化至根号，总复杂度 $n^{2.5}$。

优化二

- 考虑 1×1 的矩形，这意味着若 $f(i,j) = 1$，则 $f(i+4, j+1) = 1$。

优化二

- 考虑 1×1 的矩形，这意味着若 $f(i,j) = 1$，则 $f(i+4, j+1) = 1$。
- 于是可以设状态 $dp(i)$ 表示满足 $4S - C = i$ 的合法的方案中 S 最小是多少。

优化二

- 考虑 1×1 的矩形，这意味着若 $f(i,j) = 1$，则 $f(i+4, j+1) = 1$。
- 于是可以设状态 $dp(i)$ 表示满足 $4S - C = i$ 的合法的方案中 S 最小是多少。
- 这样状态变成了只有 $O(n)$ 种了，总复杂度 $n^2 log(n)$。

解法

上面两个优化，一个是关于转移的一个是关于状态的。

两者结合，复杂度 $n\sqrt{n}$。

介绍

两个状态 j 对状态 i 的转移表现为一些跟直线有关的式子时，可以把每个状态抽象成二维平面上的点。

如果根据转移的形式发现最优的状态必定在凸壳上的话，就可以斜率优化了。

斜率优化十分经典（套路）常见（现在好像不常见了）。

bzoj3437

有 n 个牧场，自西向东用 1 到 n 编号。

为了控制所有的牧场，需要在某些牧场上面建立控制站。

每个没有控制站的牧场被其右边第一个控制站控制，代价等于它到控制它的控制站之间距离乘上该牧场的放养量。

在第 i 个牧场建立控制站的花费是 a_i，牧场 i 的放养量是 b_i，求出最小总代价。

$n \leq 10^6$

解法

- 设 $f(i)$ 表示在第 i 个牧场设立控制站，前 i 个牧场的最小代价。

解法

- 设 $f(i)$ 表示在第 i 个牧场设立控制站，前 i 个牧场的最小代价。
- 转移显然 $f(i) = a_i + min_{j<i}\{f_j + \sum_{j<k\leq i}(i-k)b_k\}$

解法

- 设 $f(i)$ 表示在第 i 个牧场设立控制站，前 i 个牧场的最小代价。
- 转移显然 $f(i) = a_i + min_{j<i}\{f_j + \sum_{j<k\leq i}(i-k)b_k\}$
- 设 $C_n = \sum_{i\leq n} b_i, D_n = \sum_{i\leq n} ib_i$

解法

- 设 $f(i)$ 表示在第 i 个牧场设立控制站，前 i 个牧场的最小代价。
- 转移显然 $f(i) = a_i + min_{j<i}\{f_j + \sum_{j<k\leq i}(i-k)b_k\}$
- 设 $C_n = \sum_{i\leq n} b_i, D_n = \sum_{i\leq n} ib_i$
- 那么 $f(i) = a_i + min_{j<i}\{f_j + i(C_i - C_j) - (D_i - D_j)\}$

解法

- 设 $f(i)$ 表示在第 i 个牧场设立控制站，前 i 个牧场的最小代价。
- 转移显然 $f(i) = a_i + min_{j<i}\{f_j + \sum_{j<k\leq i}(i-k)b_k\}$
- 设 $C_n = \sum_{i\leq n} b_i, D_n = \sum_{i\leq n} ib_i$
- 那么 $f(i) = a_i + min_{j<i}\{f_j + i(C_i - C_j) - (D_i - D_j)\}$
- 那么用 j 来转移 i 时的贡献是 $(f_j + D_j) + i(C_i - C_j) + (a_i - D_i)$。

解法

- 设 $f(i)$ 表示在第 i 个牧场设立控制站，前 i 个牧场的最小代价。
- 转移显然 $f(i) = a_i + min_{j<i}\{f_j + \sum_{j<k\leq i}(i-k)b_k\}$
- 设 $C_n = \sum_{i\leq n} b_i, D_n = \sum_{i\leq n} ib_i$
- 那么 $f(i) = a_i + min_{j<i}\{f_j + i(C_i - C_j) - (D_i - D_j)\}$
- 那么用 j 来转移 i 时的贡献是 $(f_j + D_j) + i(C_i - C_j) + (a_i - D_i)$。
- 设点 $P_j = (C_j, f_j + D_j)$，这个贡献就是经过 P_j 的一条斜率为 i 的直线在 $x = C_i$ 上的截距加上一个常数。

解法

- 设 $f(i)$ 表示在第 i 个牧场设立控制站，前 i 个牧场的最小代价。
- 转移显然 $f(i) = a_i + min_{j<i}\{f_j + \sum_{j<k\leq i}(i-k)b_k\}$
- 设 $C_n = \sum_{i\leq n} b_i, D_n = \sum_{i\leq n} ib_i$
- 那么 $f(i) = a_i + min_{j<i}\{f_j + i(C_i - C_j) - (D_i - D_j)\}$
- 那么用 j 来转移 i 时的贡献是 $(f_j + D_j) + i(C_i - C_j) + (a_i - D_i)$。
- 设点 $P_j = (C_j, f_j + D_j)$，这个贡献就是经过 P_j 的一条斜率为 i 的直线在 $x = C_i$ 上的截距加上一个常数。
- 因此只有下凸壳上的点会产生贡献，又因为斜率是递增的，所以决策也是单调的。单调队列维护下凸壳即可。

介绍

现在有 dp:$f(i) = \max_{j<i} f(j) + v(i,j)$。

若函数 v 满足对于任意 $a < b < c < d$ 有 $v(a,d) + v(b,c) \leq v(a,c) + v(b,d)$ 那么这个 dp 满足决策单调性。

证明: 若

$$f(a) + v(a,c) \leq f(b) + v(b,c)$$

则两个不等式相加化简得到

$$f(a) + v(a,d) \leq f(b) + v(b,d)$$

于是可以得到 d 的决策必不在 c 的决策之前。

然而考试的时候一般通过打表来“证明”决策单调性。

介绍

这里只介绍分治法。

分治时带四个参数 (L, R, ql, qr)，表示当前求解 L 到 R，可用决策的区间是 ql 到 qr。

首先找到 $mid = \lfloor \frac{L+R}{2} \rfloor$ 的决策 qm。

然后递归 (L, mid, ql, qm) 和 $(mid + 1, R, qm, qr)$ 即可。递归层数只有 $log(n)$，每层复杂度是 n 乘上求 $v(i, j)$ 的复杂度。

北大集训: 树上乱走题

给出一棵 n 个点的树和一个排列。要求将排列分成连续非空的 k 段区间。

定义一段区间的价值为所有点并上 1 的虚树边长和的两倍。

求所有划分方案中最大的权值和。

$n \leq nk \leq 2 \times 10^5$

解法

- 一段区间的价值等于区间内的点按 dfs 序排序后相邻两个点的距离之和加上首尾两点到 1 的距离。

解法

- 一段区间的价值等于区间内的点按 dfs 序排序后相邻两个点的距离之和加上首尾两点到 1 的距离。
- 于是如果知道 $v(i,j)$ 可以在 $log(n)$ 的时间内将左右端点移动一位。

解法

- 一段区间的价值等于区间内的点按 dfs 序排序后相邻两个点的距离之和加上首尾两点到 1 的距离。
- 于是如果知道 $v(i,j)$ 可以在 $log(n)$ 的时间内将左右端点移动一位。
- 设 $f(i,j)$ 表示前 j 个元素，划分了 i 个区间的最大权值和。可以看出 $f(i,*)$ 转移到 $f(i+1,*)$ 时满足决策单调。

解法

- 一段区间的价值等于区间内的点按 dfs 序排序后相邻两个点的距离之和加上首尾两点到 1 的距离。
- 于是如果知道 $v(i,j)$ 可以在 $log(n)$ 的时间内将左右端点移动一位。
- 设 $f(i,j)$ 表示前 j 个元素，划分了 i 个区间的最大权值和。可以看出 $f(i,*)$ 转移到 $f(i+1,*)$ 时满足决策单调。
- 于是分治就好了。注意这里算 $v(i,j)$ 的时候要小心一点。

当转移是卷积的形式时，可以考虑列生成函数的方程来求解，之后利用多项式相关算法求解。

~~有时候甚至可以求出通项~~

51nod1947

将 $1-n$ 按顺序入栈，且可以在任意“栈非空的时刻”弹出栈顶元素，以此种方法得到的出栈序列（弹出的栈顶元素依次组成的序列）即为合法的出栈序列。

两个出栈序列是不同的，当且仅当两个序列中存在至少一位不相同的元素。

定义一个出栈序列的代价为 $cost$，则 $cost$ 为“每次有元素入栈后，将全部栈内元素的标号求和”的和。

求全部不同的合法出栈序列的 $cost$ 的和，答案对 1000000007 取模。

$n \leq 5 \times 10^7$

分析

- 尝试设 h_n 表示 n 个元素的出栈序列，每次入栈时对栈内元素标号求和的所有方案之和。

分析

- 尝试设 h_n 表示 n 个元素的出栈序列，每次入栈时对栈内元素标号求和的所有方案之和。
- 枚举 1 的出栈时间来转移。于是发现还需要记 f_n 表示 n 个元素的出栈序列方案数，g_n 表示 n 个元素的出栈序列，每次入栈时对栈内元素个数的所有方案之和。

分析

- 尝试设 h_n 表示 n 个元素的出栈序列，每次入栈时对栈内元素标号求和的所有方案之和。
- 枚举 1 的出栈时间来转移。于是发现还需要记 f_n 表示 n 个元素的出栈序列方案数，g_n 表示 n 个元素的出栈序列，每次入栈时对栈内元素个数的所有方案之和。
- 有初值 $f_0 = 1, g_0 = h_0 = 0$。可以列出转移:

$$f_n = \sum_{i=1}^{n} f_{i-1} f_{n-i}$$

$$g_n = \sum_{i=1}^{n} (g_{i-1} + i f_{i-1}) f_{n-i} + f_{i-1} g_{n-i}$$

$$h_n = \sum_{i=1}^{n} (h_{i-1} + g_{i-1} + i f_{i-1}) f_{n-i} + f_{i-1} (h_{n-i} + i g_{n-i})$$

分析

- 设 $F(x), G(x), H(x)$ 分别是三个数列的生成函数，方便起见设 $A(x) = (xF(x))'$，那么 $[x^n]A(x) = (n+1)f_n$。
- 根据转移可以列出方程:

$$F = F^2 + 1$$

$$G = x((G + A)F + GF)$$

$$H = x((H + G + A)F + HF + GA)$$

分析

- 设 $F(x), G(x), H(x)$ 分别是三个数列的生成函数，方便起见设 $A(x) = (xF(x))'$，那么 $[x^n]A(x) = (n+1)f_n$。
- 根据转移可以列出方程:

$$F = F^2 + 1$$

$$G = x((G + A)F + GF)$$

$$H = x((H + G + A)F + HF + GA)$$

- 我们知道 $F(x)$ 就是卡特兰数，其形式幂级数为 $\frac{1}{2x}(1 - u^{\frac{1}{2}})$，其中 $u = 1 - 4x$。

分析

- 设 $F(x), G(x), H(x)$ 分别是三个数列的生成函数，方便起见设 $A(x) = (xF(x))'$，那么 $[x^n]A(x) = (n+1)f_n$。
- 根据转移可以列出方程:

$$F = F^2 + 1$$
$$G = x((G + A)F + GF)$$
$$H = x((H + G + A)F + HF + GA)$$

- 我们知道 $F(x)$ 就是卡特兰数，其形式幂级数为 $\frac{1}{2x}(1 - u^{\frac{1}{2}})$，其中 $u = 1 - 4x$。
- 那么只要解方程，就可以算出

$$G(x) = \frac{1}{2}(u^{-1} - u^{-\frac{1}{2}})$$

和

$$H(x) = \frac{1}{4}(u^{-\frac{3}{2}} - u^{-\frac{1}{2}}) + \frac{x}{2}(u^{-2} - u^{-\frac{3}{2}})$$

分析

- 考虑求 $[x^n]u^{-\frac{3}{2}}$，由二项式定理有

$$\begin{aligned} u^{-\frac{3}{2}} &= (-4)^n \frac{(-\frac{3}{2})^{\underline{n}}}{n!} \\ &= 2^n \frac{3 \times 5 \times 7 \times \cdots \times (2n+1)}{n!} \\ &= \frac{(2n+1)!}{n!n!} \\ &= \binom{2n}{n}(2n+1) \end{aligned}$$

分析

- 考虑求 $[x^n]u^{-\frac{3}{2}}$，由二项式定理有

$$\begin{aligned} u^{-\frac{3}{2}} &= (-4)^n \frac{(-\frac{3}{2})^{\underline{n}}}{n!} \\ &= 2^n \frac{3 \times 5 \times 7 \times \cdots \times (2n+1)}{n!} \\ &= \frac{(2n+1)!}{n!n!} \\ &= \binom{2n}{n}(2n+1) \end{aligned}$$

- 类似地，有 $[x^n]u^{-\frac{1}{2}} = \binom{2n}{n}, [x^n]u^{-2} = 4^n(n+1)$。

分析

- 于是

$$
\begin{aligned}
[x^n]H(x) &= \frac{1}{4}(\binom{2n}{n}(2n+1)-\binom{2n}{n})+\frac{1}{2}(4^{n-1}n-(2n-1)\binom{2n-2}{n-1}) \\
&= \frac{1}{4}(\binom{2n}{n}2n-2(2n-1)\binom{2n-2}{n-1})+\frac{n}{2}4^{n-1} \\
&= \frac{1}{4}(\binom{2n-1}{n-1}2n+\binom{2n-1}{n}2n-2n\binom{2n-1}{n-1})+\frac{n}{2}4^{n-1} \\
&= \frac{n}{2}(\binom{2n-1}{n-1}+4^{n-1})
\end{aligned}
$$

???

谢谢大家。