题意

太长了。原题 https://qoj.ac/problem/8010

题解

首先这是个带标号问题，还是树的一个递归组合，需要用到指数生成函数。

但是，如果我们设所有的猫树构成的组合类为 $\mathcal{T}$，那么我们会发现没有办法进行递归组合。因为无法区分根节点。

这道题需要将猫树按照根节点是否为猫点进行分类，这样才能在递归式中满足"任何结点和它的子结点中，猫点数量小于一半"的要求。

设组合类 $\mathcal{C}$ 为根节点是猫点的猫树构成的组合类，$\mathcal{N}$ 是根节点不是猫树的猫树构成的组合类，则有 $\mathcal{T} = \mathcal{C} + \mathcal{N}$。

当根节点是猫点时，有构造：

$$\mathcal{C} = \{\circ\} \times \left(\bigcup_{i \ge 1} \bigcup_{j=0}^{i-1} SET_i(\mathcal{N}) \times SEQ_j(\mathcal{C}) \right)$$

即组合类 $\mathcal{C}$ 可以拆分为"猫点根节点、无序的 i 棵非猫点根节点的树、有序的 j 棵猫点根节点的树"，因为猫点是根节点，所以必须有 $j \le i-1$。

相似的，我们可以得到：

$$\mathcal{N} = \{\bullet\} \times \left(\bigcup_{i \ge 0} \bigcup_{j=0}^{i+1} SET_i(\mathcal{N}) \times SEQ_j(\mathcal{C}) \right)$$

接下来开始推导生成函数形式：

其实可以直接看符号化体系（

$SEQ_j(\mathcal{C})$ 的每个元素都由有序的 j 个 $\mathcal{C}$ 中元素构成，因此只是简单的笛卡尔积：

$$SEQ_j(\mathcal{C}) = \underbrace{\mathcal{C} \times \mathcal{C} \times \cdots \times \mathcal{C}}_{j\ times}$$

$$SEQ_j(\mathcal{C}) \Leftrightarrow C^j(x)$$

$SET_i(\mathcal{N})$ 的每个元素则都是无序的，但是由于这里是有标号体系，因此只需要除以 $i!$ 就能简单的去重。

$$SET_i(\mathcal{N}) = SEQ_i(N)/G_i$$

$$SET_i(\mathcal{N}) \Leftrightarrow \frac{N^i(x)}{i!}$$

其中 G_i 为大小为 i 的全排列构成的置换群。

带入之前得到的递归式中（从这里往后，为了简洁，不再写出(x)）：

$$C = x\sum_{i\geq 1}\sum_{j=0}^{i-1}\frac{N^i}{i!}C^j = x\frac{e^N - e^{CN}}{1-C}$$
$$N = x\sum_{i\geq 0}\sum_{j=0}^{i+1}\frac{N^i}{i!}C^j = x\frac{e^C - C^2e^{CN}}{1-C}$$

至此，你可以用半在线卷积或者叫分治 fft 什么的方法在 $O(n\log^2 n)$ 的时间内求出 C 和 N 的前 n 项。

时间复杂度 $T(n) = 2T(n/2) + O(n\log n) = O(n\log^2 n)$。

具体做法是因为 C 和 N 的某一项都是由这一项之前的值的某种卷积决定的，所以你只需要在卷积的时候维护：

$$C, N, 1-C, \frac{1}{1-C}, C^2, e^C, CN, e^{CN}, C^2e^{CN} \text{（或许还有？）}$$

嗯嘛啊，总能做到的。

另一种方法是使用牛顿迭代法，可以在 $O(n\log n)$ 的时间内解出上述方程组的前 n 项。

我们调整一下方程组，并设两个以多项式为参数的函数：

$$\begin{cases} P(C,N) = C - C^2 - xe^N + xe^{CN} = 0 \\ Q(C,N) = N - CN - xe^N + C^2e^{CN} = 0 \end{cases}$$

很显然 $C(x)$ 的第零项是 0，$N(x)$ 的第零项是 1。

假设我们已经求出了 $C(x) \pmod{x^n}$ 和 $N(x) \pmod{x^n}$，我们令它们分别为 C_0 和 N_0。

现在需要求出 $C(x) \pmod{x^{2n}}$ 和 $N(x) \pmod{x^{2n}}$，将 P 和 Q 在 C_0, N_0 处进行二元泰勒展开：

$P_0 = P(C_0, N_0), Q_0 = Q(C_0, N_0)$

$$\begin{cases} P = P_0 + \left.\frac{\partial P}{\partial C}\right|_{C_0}(C - C_0) + \left.\frac{\partial P}{\partial N}\right|_{N_0}(N - N_0) + o(x^{2n}) = 0 \\ Q = Q_0 + \left.\frac{\partial Q}{\partial C}\right|_{C_0}(C - C_0) + \left.\frac{\partial Q}{\partial N}\right|_{N_0}(N - N_0) + o(x^{2n}) = 0 \end{cases}$$

最后的部分是 $o(x^{2n})$，是因为 $C - C_0, N - N_0$ 的次数都大于等于 x^{n+1}，二次项后的结果只会大于等于 x^{2n+2}。

截取前 x^{2n} 项，并将 $\left.\frac{\partial P}{\partial C}\right|_{C_0}$ 简记为 P_{C_0}：

$$\begin{cases} P_0 + P_{C_0}(C - C_0) + P_{N_0}(N - N_0) = 0 \\ Q_0 + Q_{C_0}(C - C_0) + Q_{N_0}(N - N_0) = 0 \end{cases} \pmod{x^{2n}}$$

现在，这个式子中只有 C, N 是未知量了，也就是二元一次方程。不过你想用消元法吗?

想来也是，所以这里用到的是克拉默法则：

$$\begin{cases} C=\dfrac{\begin{vmatrix} C_0P_{C_0}+N_0P_{N_0}-P_0 & P_{N_0} \\ C_0Q_{C_0}+N_0Q_{N_0}-Q_0 & Q_{N_0} \end{vmatrix}}{\begin{vmatrix} P_{C_0} & P_{N_0} \\ Q_{C_0} & Q_{N_0} \end{vmatrix}} \\ Q=\dfrac{\begin{vmatrix} P_{C_0} & C_0P_{C_0}+N_0P_{N_0}-P_0 \\ Q_{C_0} & C_0Q_{C_0}+N_0Q_{N_0}-Q_0 \end{vmatrix}}{\begin{vmatrix} P_{C_0} & P_{N_0} \\ Q_{C_0} & Q_{N_0} \end{vmatrix}} \end{cases} \pmod{x^{2n}}$$

终于，漫长的旅程结束了。

总结一下，从 $C(x)$ 的第零项是 0，$N(x)$ 的第零项是 1 开始，每次迭代中计算：

$$\begin{cases} P_0=C_0-C_0^2-xe^{N_0}+xe^{C_0N_0} \\ Q_0=N_0-C_0N_0-xe^{N_0}+xC_0^2e^{C_0N_0} \\ P_{C_0}=1-2C_0+xN_0e^{C_0N_0} \\ P_{N_0}=-xe^{N_0}+xC_0e^{C_0N_0} \\ Q_{C_0}=-N_0+2xC_0e^{C_0N_0}+xC_0^2N_0e^{C_0N_0} \\ Q_{N_0}=1-C_0-xe^{N_0}+xC_0^3e^{C_0N_0} \\ C=\dfrac{\begin{vmatrix} C_0P_{C_0}+N_0P_{N_0}-P_0 & P_{N_0} \\ C_0Q_{C_0}+N_0Q_{N_0}-Q_0 & Q_{N_0} \end{vmatrix}}{\begin{vmatrix} P_{C_0} & P_{N_0} \\ Q_{C_0} & Q_{N_0} \end{vmatrix}} \\ Q=\dfrac{\begin{vmatrix} P_{C_0} & C_0P_{C_0}+N_0P_{N_0}-P_0 \\ Q_{C_0} & C_0Q_{C_0}+N_0Q_{N_0}-Q_0 \end{vmatrix}}{\begin{vmatrix} P_{C_0} & P_{N_0} \\ Q_{C_0} & Q_{N_0} \end{vmatrix}} \end{cases} \pmod{x^{2n}}$$

嗯嘛啊。

时间复杂度 $T(n)=T(n/2)+O(n\log n)=O(n\log n)$，常数比较大，有些共同的部分注意不要重复计算。

恭喜你看到最后。