题目大意

有一棵树，树初始只有一个根节点。

在时刻 0 到 $n-1$，每个时刻会发生一次以下的操作：

- 所有深度与当前时刻 t 模 2 同余的点长出一个叶子。（根节点深度为 0，儿子节点的深度为父亲节点的深度 $+1$。）

求 n 次操作后树上距离为 d 的无序点对的数量，对 998244353 取模。

数据范围

对于所有数据：$1 \leq n \leq 10^9$，$1 \leq d \leq 10^5$。

$Subtask\ 1\ (1\ pts)$：$1 \leq n \leq 25$，$1 \leq d \leq 400$。

$Subtask\ 2\ (7\ pts)$：$1 \leq n \leq 400$，$1 \leq d \leq 400$。

$Subtask\ 3\ (10\ pts)$：$1 \leq n \leq 2000$，$1 \leq d \leq 2000$。

$Subtask\ 4\ (14\ pts)$：$1 \leq n \leq 5000$，$1 \leq d \leq 5000$。

$Subtask\ 5\ (19\ pts)$：$1 \leq d \leq 100$。

$Subtask\ 6\ (36\ pts)$：$1 \leq d \leq 5000$。

$Subtask\ 7\ (13\ pts)$：无特殊限制。

时间限制：$1s$。

空间限制：$512MB$。

解题过程

算法 1

考虑暴力建出树，然后换根维护，复杂度 $O(1.618^n n^2)$，可以通过子任务 1。

算法 2

设 $dp_{i,j,k}$ 表示：

- j 为奇数时 $dp_{i,j,0}$ 与 $dp_{i,j,1}$ 均表示时刻 i 时长度为 j 的二元组数量。
- j 为偶数时 $dp_{i,j,0}$ 表示时刻 i 时长度为 j 且两个端点深度都为偶数的数量。
- j 为奇数时 $dp_{i,j,1}$ 表示时刻 i 时长度为 j 且两个端点深度都为奇数的数量。

我们可以讨论在下一时刻新增叶子时我们是否把端点移到新增的叶子上，分时刻奇偶性讨论容易发现转移是 $O(1)$ 的。

算法复杂度 $O(nd)$，可以通过子任务 $1,2,3,4$，期望得分 32。

算法 3

我们将算法 2 的转移用矩阵表示，容易发现转移矩阵只和时刻的奇偶性有关。

所以我们可以用矩阵快速幂优化转移的过程，算法复杂度 $O(d^3logn)$，可以通过子任务 $1,2,5$，期望得分 27。

算法 4

让我们考虑另一种转移方式：将最终的树关于第一次操作的边进行边分治。

这样我们就可以认为一棵 n 时刻的树是由一棵 $n-1$ 时刻的树和一棵 $n-2$ 时刻的树拼成的。

我们可以讨论链是否经过我们钦定的特殊边，容易发现瓶颈在计算 i 次操作后深度为 j 的点的数量的卷积上。

设 $f_{i,j}$ 表示 i 次操作后深度为 j 的点的数量，则：

- $f_{0,0}=1$
- 当 $i+j$ 为偶数的时候 $f_{i,j}=f_{i-1,j-1}+f_{i-1,j}$
- 当 $i+j$ 为奇数的时候 $f_{i,j}=f_{i-1,j}$

容易发现 f 的转移是 $O(nd)$ 的，求 f_i 和 f_{i+1} 卷积的复杂度为 $O(nd^2)$。

算法总复杂度 $O(nd^2)$，可以通过子任务 $1,2$，期望得分 8。

另外，$f_{i,j}$ 有优秀的表达形式：

让我们只考虑 $i+j$ 为偶数的 $f_{i,j}$，则 $f_{i,j}=f_{i-1,j-1}+f_{i-2,j}$

设 $k=\frac{i+j}{2}$，令 $g_{k,j}=f_{i,j}$，则有 $g_{k,j}=g_{k-1,j}+g_{k-1,j-1}$

容易发现 g 就是组合数，故 $f_{i,j}=\binom{\frac{i+j}{2}}{j}$

算法 5

使用 NTT 优化卷积，复杂度为 $O(ndlogd)$，可以通过子任务 $1,2,3$，期望得分 18。

算法 6

让我们结合算法 2，算法 3，算法 4：

注意到，算法 2 的转移可以被看作带入，更具体地，$dp_{i,j,k}$ 可以被写成 $dp_{i-1,j,k}+\sum dp_{i-1,j-t,l}$ 的形式，其中 t 和 l 都是常数，且 t 是正整数。

这意味着我们有了一种不损失复杂度的，使第一维减 1，且不会损失后两维的代换方式（这里不考虑 $\sum dp_{i-1,j-t,l}$，我们默认我们在求出 $dp_{n,d}$ 前已经求出了 $dp_{n,d-1}$）

我们对于算法 4 的式子使用这种代换，使得第一维全部为 $n-2$，然后进行换元。

这样我们就得到了 $O(1)$ 从 $dp_{i,j}$ 推到 $dp_{i,j+1}$ 的递推式，且只需要计算 f_n 与 f_{n+1} 的卷积。

值得注意的是前 $O(1)$ 项并不能直接套用这里的递推式，因为我们在带入的时候有越界的情况，未被定义的部分并不为 0，你需要用算法 3 算出前 $O(1)$ 项。

复杂度为 $O(logn+d^2)$，可以通过子任务 $1,2,3,4,5,6$，期望得分 87。

算法 7

使用 NTT 优化卷积，复杂度为 $O(logn + dlogd)$，可以通过子任务 $1, 2, 3, 4, 5, 6, 7$，期望得分 100 。

参考资料

无。