堆解丑数问题

↳

堆的方法本质上是一种 BFS（广度优先搜索）
为什么这么说，深度优先和广度优先有什么区别？

这是一个非常好的问题！**为什么堆的方法本质上是 BFS（广度优先搜索），而不是 DFS（深度优先搜索）？**

我们先回顾 BFS 和 DFS 的核心区别，然后具体分析 **堆方法为什么是 BFS**。

📌 深度优先搜索（DFS） vs. 广度优先搜索（BFS）

	DFS（深度优先）	BFS（广度优先）
搜索方式	先深入到某一条路径到底，再回溯	按层级扩展，每次探索当前层的所有可能
数据结构	**栈（递归或显式栈）**	**队列（FIFO，先进先出）**
特点	可能会先找到较大的值，但未必是最优的	总是按照从小到大的顺序依次找到目标
适用于	需要深入探索、回溯的情况（如迷宫、决策树）	需要逐层扩展、寻找最短路径的情况

🎯 为什么堆的方法本质是 BFS？

🔍 观察堆的行为

堆方法的核心是：

1. **维护一个小顶堆**，每次取出最小的丑数 `cur`。
2. **生成下一个丑数**：`cur * 2`、`cur * 3`、`cur * 5` 并加入堆（如果之前没出现过）。
3. **循环进行 `n` 次**，直到找到第 `n` 个丑数。

从这里可以看到：

- 每次都从堆中取出 **当前最小的丑数**，然后生成更大的丑数，**按照大小逐步扩展丑数集合**。
- 这意味着 **丑数是按照大小从小到大一层层扩展的，与 BFS 的特性完全一致！**

📌 BFS 的关键特征

1. **一层层扩展**：堆的最小值始终是当前能找到的最小丑数，每次扩展 `cur * 2, cur * 3, cur * 5`，类似于 BFS 扩展新一层。
2. **逐步寻找目标**：每次取出的 `cur` **一定是当前可达的最小丑数**，不会错过任何数，符合 BFS 的性质。
3. **避免重复（去重）**：通过 `set` 记录访问过的丑数，避免重复计算，和 BFS 记录访问过的节点类似。

🚀 为什么不是 DFS？

如果是 DFS，意味着：

- 我们会直接沿着一条路径深入，比如 `1 → 2 → 4 → 8 → 16...`，然后再回溯到 `1 → 3 → 9 → 27...`。
- 这会导致我们**可能找到一些很大的丑数，而错过了较小的丑数**，打乱了顺序。
- 但是这道题要求**严格递增**，DFS **无法保证优先找到较小的丑数**，不适合！

🔎 总结

为什么堆方法是 BFS？

- **堆每次取出的是当前最小的丑数**，保证丑数按顺序增长。
- **新扩展的丑数都是由当前最小丑数生成的**，像 BFS 层层扩展。
- **使用 `set` 去重，避免重复计算**，和 BFS 访问标记类似。
- **DFS 会先深入某个分支，导致找到的丑数顺序混乱**，而 BFS 总是按照从小到大的顺序生成丑数，更符合要求。

所以，这道题用堆的方法实际上就是**带优先级的 BFS**，而不是 DFS！

你觉得这个解释清楚了吗？或者你希望用更形象的方式来理解？😊