题目地址(401. 二进制手表)

https://leetcode-cn.com/problems/binary-watch/

入选理由

1. 搜索篇的题目为： 回溯 -> BFS/DFS（BFS 和 DFS 不做区分，因为基本上能用 DFS 的，也能用 BFS，互通的），先从简单的回溯开始。 大家注意画图哦

标签

- 回溯

难度

- 简单

题目描述

二进制手表顶部有 4 个 LED 代表 小时（0-11），底部的 6 个 LED 代表 分钟（0-59）。

每个 LED 代表一个 0 或 1，最低位在右侧。

例如，上面的二进制手表读取“3:25”。

给定一个非负整数 n 代表当前 LED 亮着的数量，返回所有可能的时间。

示例：

输入：n = 1
返回：["1:00", "2:00", "4:00", "8:00", "0:01", "0:02", "0:04", "0:08", "0:16", "0:32"]

提示：

输出的顺序没有要求。
小时不会以零开头，比如“01:00”是不允许的，应为“1:00”。
分钟必须由两位数组成，可能会以零开头，比如“10:2”是无效的，应为“10:02”。
超过表示范围（小时 0-11，分钟 0-59）的数据将会被舍弃，也就是说不会出现 "13:00", "0:61" 等时间。

前置知识

- 笛卡尔积
- 回溯

公司

- 阿里
- 腾讯
- 百度
- 字节

思路

一看题目就是一个笛卡尔积问题。

即给你一个数字 num， 我可以将其分成两部分。其中一部分（不妨设为 a）给小时，另一部分给分（就是 num - a）。 最终的结果就是 a 能表示的所有小时的集合和 num - a 所能表示的分的集合的笛卡尔积。

用代码表示就是：

```
# 枚举小时
for a in possible_number(i):
    # 小时确定了，分就是 num - i
    for b in possible_number(num - i, True):
        ans.add(str(a) + ":" + str(b).rjust(2, '0'))
```

枚举所有可能的 (a, num - a) 组合即可。

核心代码：

```
for i in range(min(4, num + 1)):
    for a in possible_number(i):
        for b in possible_number(num - i, True):
            ans.add(str(a) + ":" + str(b).rjust(2, '0'))
```

代码

```
class Solution:
    def readBinaryWatch(self, num: int) -> List[str]:
        def possible_number(count, minute=False):
            if count == 0: return [0]
            if minute:
                return filter(lambda a: a < 60, map(sum, combinations([1, 2, 4, 8, 16, 32], count)))
            return filter(lambda a: a < 12, map(sum, combinations([1, 2, 4, 8], count)))
        ans = set()
        for i in range(min(4, num + 1)):
            for a in possible_number(i):
                for b in possible_number(num - i, True):
                    ans.add(str(a) + ":" + str(b).rjust(2, '0'))
        return list(ans)
```

进一步思考，实际上，我们要找的就是 a 和 b 相加等于 num，并且 a 和 b 就是二进制表示中 1 的个数。 因此可以将逻辑简化为：

```python
class Solution:
    def readBinaryWatch(self, num: int) -> List[str]:
        return [str(a) + ":" + str(b).rjust(2, '0') for a in range(12) for b in range(60) if (bin(a)+bin(b)).count('1')
```

大家对此有何看法，欢迎给我留言，我有时间都会一一查看回答。更多算法套路可以访问我的 LeetCode 题解仓库：https://github.com/azl397985856/leetcode 。 目前已经 37K star 啦。 大家也可以关注我的公众号《力扣加加》带你啃下算法这块硬骨头。

上一页 下一页