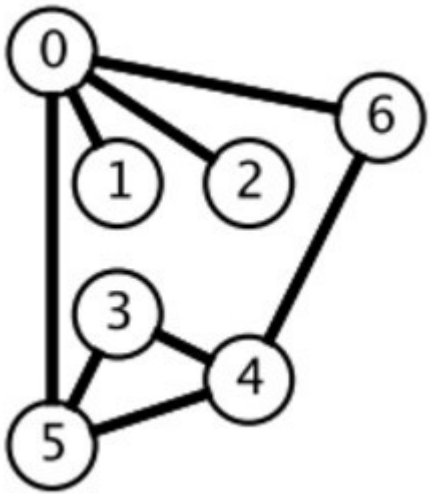
279. 完全平方数

题目：279. Perfect Squares 语言: python3 英文版链接：https://leetcode.com/problems/perfect-squares/description/
中文版链接：https://leetcode-cn.com/problems/perfect-squares/description/

题目分析



广度优先搜索一层一层地进行遍历，每层遍历都以上一层遍历的结果作为起点，遍历一个距离能访问到的所有节点。需要注意的是，遍历过的节点不能再次被遍历。

第一层：

- 0 -> {6,2,1,5}

第二层：

- 6 -> {4}
- 2 -> {}
- 1 -> {}
- 5 -> {3}

第三层：

- 4 -> {}
- 3 -> {}

每一层遍历的节点都与根节点距离相同。设 di 表示第 i 个节点与根节点的距离，推导出一个结论：对于先遍历的节点 i 与后遍历的节点 j，有 di <= dj。利用这个结论，可以求解最短路径等 **最优解** 问题：第一次遍历到目的节点，其所经过的路径为最短路径。应该注意的是，使用 BFS 只能求解无权图的最短路径。

在程序实现 BFS 时需要考虑以下问题：

- 队列：用来存储每一轮遍历得到的节点；
- 标记：对于遍历过的节点，应该将它标记，防止重复遍历。

可以将每个整数看成图中的一个节点，如果两个整数之差为一个平方数，那么这两个整数所在的节点就有一条边。

要求解最小的平方数数量，就是求解从节点 n 到节点 0 的最短路径。

答案

```python
class Solution:
    def numSquares(self, n: int) -> int:
        squares = []
        square, interval = 1, 3
        while square <= n:
            squares.append(square)
            square += interval
            interval += 2
        list.reverse(squares)
        q = [(0, n)]
        # 使用used判断从n到x这条路径是否走过，根据BFS原理，如果走过那一定是更近的或者一样的路
        used = [False for x in range(n+1)]
        while q:
            temp = q.pop(0)
            if temp[1] == 0:
                return temp[0]
            for x in squares:
                if temp[1] - x >= 0 and not used[temp[1] - x]:
                    q.append((temp[0] + 1, temp[1] - x))
                    used[temp[1] - x] = True
        return -1
```