773. Sliding Puzzle

On an `2 x 3` board, there are five tiles labeled from `1` to `5`, and an empty square represented by `0`. A move consists of choosing `0` and a 4-directionally adjacent number and swapping it.

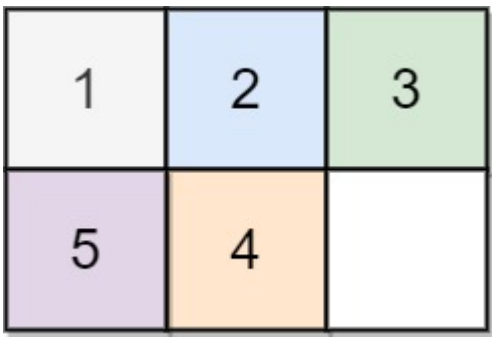
The state of the board is solved if and only if the board is `[[1,2,3],[4,5,0]]`.

Given the puzzle board `board`, return *the least number of moves required so that the state of the board is solved*. If it is impossible for the state of the board to be solved, return `-1`.

Example 1:

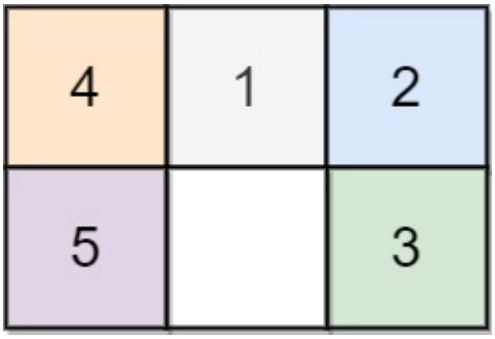

```
Input: board = [[1,2,3],[4,0,5]]
Output: 1
Explanation: Swap the 0 and the 5 in one move.
```

Example 2:



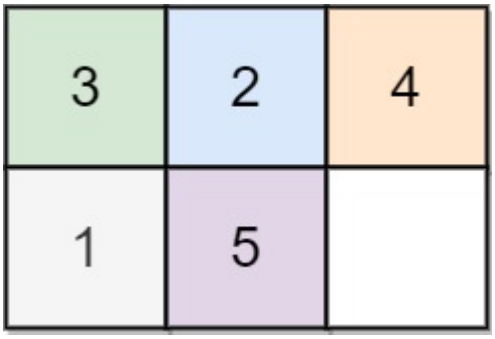

```
Input: board = [[1,2,3],[5,4,0]]
Output: -1
Explanation: No number of moves will make the board solved.
```

Example 3:




```
Input: board = [[4,1,2],[5,0,3]]
Output: 5
Explanation: 5 is the smallest number of moves that solves the board.
An example path:
After move 0: [[4,1,2],[5,0,3]]
After move 1: [[4,1,2],[0,5,3]]
After move 2: [[0,1,2],[4,5,3]]
After move 3: [[1,0,2],[4,5,3]]
After move 4: [[1,2,0],[4,5,3]]
After move 5: [[1,2,3],[4,5,0]]
```

Example 4:




```
Input: board = [[3,2,4],[1,5,0]]
Output: 14
``````Python
class Solution:
    def slidingPuzzle(self, board: List[List[int]]) -> int:
        queue = []
        initial = ''
        for i in range(len(board)):
            for j in range(len(board[0])):
                initial = initial + str(board[i][j])
        target = '123450'
        seen = set()
        moves = {0:[1,3],1:[0,2,4],2:[1,5],3:[0,4],4:[1,3,5],5:[2,4]}
        queue.append(initial)
```
```

```
        res = 0
        while len(queue):
            length = len(queue)
            while length>0:
                temp = queue.pop(0)
                if temp==target:
                    return res
                idx = temp.index('0')
                for ids in moves[idx]:
                    newStr = self.swappedStr(temp,idx,ids)
                    if newStr not in seen:
                        queue.append(newStr)
                        seen.add(newStr)
                length = length-1
            res = res+1
        return -1


    def swappedStr(self,string,idx,newidx):
        res = list(string)
        res[idx],res[newidx] = res[newidx],res[idx]
        return ''.join(res)
```