```
/*
在给定的网格中，每个单元格可以有以下三个值之一：

    值 0 代表空单元格；
    值 1 代表新鲜橘子；
    值 2 代表腐烂的橘子。

每分钟，任何与腐烂的橘子（在 4 个正方向上）相邻的新鲜橘子都会腐烂。

返回直到单元格中没有新鲜橘子为止所必须经过的最小分钟数。如果不可能，返回 -1。



示例 1：

输入：[[2,1,1],[1,1,0],[0,1,1]]
输出：4

示例 2：

输入：[[2,1,1],[0,1,1],[1,0,1]]
输出：-1
解释：左下角的橘子（第 2 行， 第 0 列）永远不会腐烂，因为腐烂只会发生在 4 个正向上。

示例 3：

输入：[[0,2]]
输出：0
解释：因为 0 分钟时已经没有新鲜橘子了，所以答案就是 0 。



提示：

    1 <= grid.length <= 10
    1 <= grid[0].length <= 10
    grid[i][j] 仅为 0、1 或 2

通过次数8,299
提交次数17,563

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/rotting-oranges
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析：

- 方法一：按分钟一个回合进行BFS.

方法一：C++_BFS

```
class Solution
{
    private:

        int minute_event(   vector<vector<int>>&        grid            ,
                            vector<vector<int>>&        vistied         ,
                            int                         r               ,
                            int                         c               ,
                            int                         cur_fresh_num
    )
    {

        int i = 0;
        int j = 0;

        for (i = 0; i < r; i++)
        {
            for (j = 0; j < c; j++)
            {
                if (grid[i][j] == 0)
                {
                    continue;
                }
                else if (grid[i][j] == 1)
                {
                    continue;
                }
                else
                {
                    if (vistied[i][j] == 0 || vistied[i][j] == 1)
                    {
                        //上
                        if (i - 1 >= 0 && grid[i - 1][j] == 1)
                        {
                            grid[i - 1][j] = 2;
                            vistied[i - 1][j] = 2;
                            cur_fresh_num--;
                        }

                        //下
                        if (i + 1 < r && grid[i + 1][j] == 1)
                        {
                            grid[i + 1][j] = 2;
                            vistied[i + 1][j] = 2;
                            cur_fresh_num--;
                        }

                        //左
                        if (j - 1 >= 0 && grid[i][j - 1] == 1)
                        {
```

```
                            grid[i][j - 1] = 2;
                            vistied[i][j - 1] = 2;
                            cur_fresh_num--;
                        }
                        //右

                        if (j + 1 < c && grid[i][j + 1] == 1)
                        {
                            grid[i][j + 1] = 2;
                            vistied[i][j + 1] = 2;
                            cur_fresh_num--;
                        }

                        vistied[i][j] = 1;

                    }
                    else if (vistied[i][j] == 2)
                    {
                        vistied[i][j] = 1;
                    }
                }
            }
        }

        for (i = 0; i < r; i++)
        {
            for (j = 0; j < c; j++)
            {
                if (vistied[i][j] == 2)
                {
                    vistied[i][j] = 1;
                }
            }

        }
        return cur_fresh_num;
    }

    public:
        int orangesRotting(vector<vector<int>>& grid)
        {
            int r               = grid.size()       ;
            int c               = grid[0].size()    ;
            int cur_rotten_num  = 0                 ;
            int cur_fresh_num   = 0                 ;
            int new_fresh_num   = 0                 ;
            int minutes         = 0                 ;
            int i               = 0                 ;
            int j               = 0                 ;

            vector<vector<int>>  vistied(r, vector<int>(c, 0));

            for (i = 0; i < r; i++)
            {
                for (j = 0; j < c; j++)
                {
                    if (grid[i][j] == 1)
                    {
```

```
                        cur_fresh_num++;
                    }
                    else if (grid[i][j] == 2)
                    {
                        cur_rotten_num++;
                    }
                }
            }

            if (cur_rotten_num != 0)
            {

                while (1)
                {
                    new_fresh_num = minute_event(grid, vistied, r, c,
cur_fresh_num);
                    if (new_fresh_num == cur_fresh_num)
                    {
                        break;
                    }
                    cur_fresh_num = new_fresh_num;
                    minutes++;
                }
            }

            if (cur_fresh_num == 0)
            {
                return minutes;
            }
            else
            {
                return -1;
            }
        }
};


/*
执行结果:
通过
显示详情
执行用时 :8 ms, 在所有 C++ 提交中击败了76.60% 的用户
内存消耗 :15.2 MB, 在所有 C++ 提交中击败了5.40%的用户
*/
```

AlimyBreak
2020.03.04(2020年第10周)