

```

1  /*
2  在给定的网格中，每个单元格可以有以下三个值之一：
3
4      值 0 代表空单元格；
5      值 1 代表新鲜橘子；
6      值 2 代表腐烂的橘子。
7
8  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。
9
10  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。
11
12
13
14  示例 1:
15
16  输入: [[2,1,1],[1,1,0],[0,1,1]]
17  输出: 4
18
19  示例 2:
20
21  输入: [[2,1,1],[0,1,1],[1,0,1]]
22  输出: -1
23  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。
24
25  示例 3:
26
27  输入: [[0,2]]
28  输出: 0
29  解释: 因为 0 分钟时已经没有新鲜橘子了，所以答案就是 0 。
30
31
32
33  提示:
34
35      1 <= grid.length <= 10
36      1 <= grid[0].length <= 10
37      grid[i][j] 仅为 0、1 或 2
38
39  通过次数8,299
40  提交次数17,563
41
42  来源：力扣（LeetCode）
43  链接：https://leetcode-cn.com/problems/rotting-oranges
44  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。
45  */

```

分析：

- 方法一：按分钟一个回合进行BFS.

方法一: C++_BFS

```
1  class Solution
2  {
3      private:
4
5          int minute_event(    vector<vector<int>>&    grid            ,
6                              vector<vector<int>>&    vistied        ,
7                              int                    r            ,
8                              int                    c            ,
9                              int                    cur_fresh_num
10         )
11     {
12
13         int i = 0;
14         int j = 0;
15
16         for (i = 0; i < r; i++)
17         {
18             for (j = 0; j < c; j++)
19             {
20                 if (grid[i][j] == 0)
21                 {
22                     continue;
23                 }
24                 else if (grid[i][j] == 1)
25                 {
26                     continue;
27                 }
28                 else
29                 {
30                     if (vistied[i][j] == 0 || vistied[i][j] == 1)
31                     {
32                         //上
33                         if (i - 1 >= 0 && grid[i - 1][j] == 1)
34                         {
35                             grid[i - 1][j] = 2;
36                             vistied[i - 1][j] = 2;
37                             cur_fresh_num--;
38                         }
39
40                         //下
41                         if (i + 1 < r && grid[i + 1][j] == 1)
42                         {
43                             grid[i + 1][j] = 2;
44                             vistied[i + 1][j] = 2;
45                             cur_fresh_num--;
46                         }
47
48                         //左
49                         if (j - 1 >= 0 && grid[i][j - 1] == 1)
50                         {
```

```

51         grid[i][j - 1] = 2;
52         vistied[i][j - 1] = 2;
53         cur_fresh_num--;
54     }
55     //右
56
57     if (j + 1 < c && grid[i][j + 1] == 1)
58     {
59         grid[i][j + 1] = 2;
60         vistied[i][j + 1] = 2;
61         cur_fresh_num--;
62     }
63
64     vistied[i][j] = 1;
65
66     }
67     else if (vistied[i][j] == 2)
68     {
69         vistied[i][j] = 1;
70     }
71 }
72 }
73 }
74
75 for (i = 0; i < r; i++)
76 {
77     for (j = 0; j < c; j++)
78     {
79         if (vistied[i][j] == 2)
80         {
81             vistied[i][j] = 1;
82         }
83     }
84 }
85
86 return cur_fresh_num;
87 }
88
89 public:
90 int orangesRotting(vector<vector<int>>& grid)
91 {
92     int r          = grid.size()          ;
93     int c          = grid[0].size()       ;
94     int cur_rotten_num = 0                  ;
95     int cur_fresh_num  = 0                  ;
96     int new_fresh_num  = 0                  ;
97     int minutes       = 0                  ;
98     int i             = 0                  ;
99     int j             = 0                  ;
100
101     vector<vector<int>> vistied(r, vector<int>(c, 0));
102
103     for (i = 0; i < r; i++)
104     {
105         for (j = 0; j < c; j++)
106         {
107             if (grid[i][j] == 1)
108             {

```

```

109         cur_fresh_num++;
110     }
111     else if (grid[i][j] == 2)
112     {
113         cur_rotten_num++;
114     }
115 }
116 }
117
118 if (cur_rotten_num != 0)
119 {
120
121     while (1)
122     {
123         new_fresh_num = minute_event(grid, vistied, r, c,
cur_fresh_num);
124         if (new_fresh_num == cur_fresh_num)
125         {
126             break;
127         }
128         cur_fresh_num = new_fresh_num;
129         minutes++;
130     }
131 }
132
133 if (cur_fresh_num == 0)
134 {
135     return minutes;
136 }
137 else
138 {
139     return -1;
140 }
141 }
142 };
143
144
145 /*
146 执行结果:
147 通过
148 显示详情
149 执行用时 :8 ms, 在所有 C++ 提交中击败了76.60% 的用户
150 内存消耗 :15.2 MB, 在所有 C++ 提交中击败了5.40%的用户
151 */

```