

🌊简单Java, 秒懂图的BFS～

[Sweetiee 🍬](https://leetcode-cn.com/u/sweetiee/)发布于 5 小时前3.5kJavaBFS

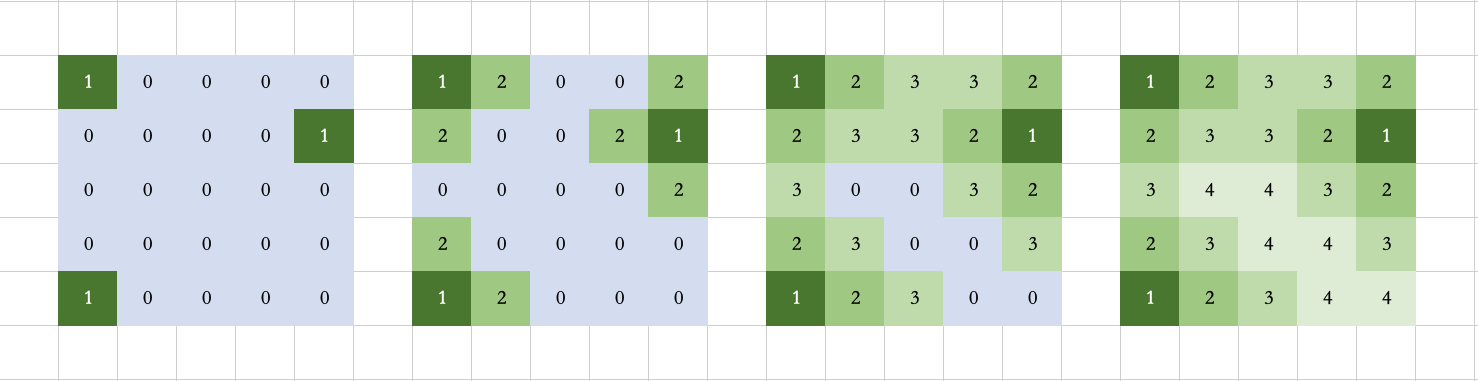
男たちは グランド ライン を 目指し（めざし）梦を（ゆめを）追い（おい）続ける（つづける）。世はまさに大海贼 时代（じだい）。  
于是男子汉们起程前往伟大的航路，追逐梦想，大海贼时代来临了。

🙋🙋 某不可描述的原因，我迟到了。。。但不会缺席！

**一、题目分析**

相信对于**Tree的BFS**大家都已经轻车熟路了：要把root节点先入队，然后再一层一层的无脑遍历就行了。

对于**图的BFS**也是一样滴～ 与Tree的BFS区别如下：  
**1、tree只有1个root，而图可以有多个源点，所以首先需要把多个源点都入队。**  
**2、tree是有向的因此不需要标志是否访问过，而对于无向图来说，必须得标志是否访问过！  
并且为了防止某个节点多次入队，需要在入队之前就将其设置成已访问！**

**这是一道典型的BFS基础应用，为什么这么说呢？**  
因为我们只要先把所有的陆地都入队，然后从各个陆地**同时开始**一层一层的向海洋扩散，那么最后扩散到的海洋就是最远的海洋！  
并且这个海洋肯定是被离他最近的陆地给扩散到的！  
下面是扩散的图示，1表示陆地，0表示海洋。每次扩散的时候会标记相邻的4个位置的海洋：  


你可以想象成你从每个陆地上派了很多支船去踏上伟大航道，踏遍所有的海洋。每当船到了新的海洋，就会分裂成4条新的船，向新的未知海洋前进（访问过的海洋就不去了）。如果船到达了某个未访问过的海洋，那他们是第一个到这片海洋的。很明显，这么多船最后访问到的海洋，肯定是离陆地最远的海洋。

**二、代码实现**

class Solution {

public int maxDistance(int[][] grid) {

int[] dx = {0, 0, 1, -1};

int[] dy = {1, -1, 0, 0};

Queue<int[]> queue = new ArrayDeque<>();

int m = grid.length, n = grid[0].length;

// 先把所有的陆地都入队。

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

if (grid[i][j] == 1) {

queue.offer(new int[] {i, j});

}

}

}

// 从各个陆地开始，一圈一圈的遍历海洋，最后遍历到的海洋就是离陆地最远的海洋。

boolean hasOcean = false;

int[] point = null;

while (!queue.isEmpty()) {

point = queue.poll();

int x = point[0], y = point[1];

// 取出队列的元素，将其四周的海洋入队。

for (int i = 0; i < 4; i++) {

int newX = x + dx[i];

int newY = y + dy[i];

if (newX < 0 || newX >= m || newY < 0 || newY >= n || grid[newX][newY] != 0) {

continue;

}

grid[newX][newY] = grid[x][y] + 1; // 这里我直接修改了原数组，因此就不需要额外的数组来标志是否访问

hasOcean = true;

queue.offer(new int[] {newX, newY});

}

}

// 没有陆地或者没有海洋，返回-1。

if (point == null || !hasOcean) {

return -1;

}

// 返回最后一次遍历到的海洋的距离。

return grid[point[0]][point[1]] - 1;

}

}

**三、再来一道感受一下图的BFS**

[**310. 最小高度树**](https://leetcode-cn.com/problems/minimum-height-trees/)

**题目描述**：对于一个具有树特征的无向图，我们可选择任何一个节点作为根。图因此可以成为树，在所有可能的树中，具有最小高度的树被称为最小高度树。给出这样的一个图，写出一个函数找到所有的最小高度树并返回他们的根节点。

本题的其中一种解法就是**图的BFS**：首先把各个叶子节点（入度为1的节点）全部入队，一层一层的剥掉最外层的叶子结点，那么最后剩下的1个节点（或2个节点）则就是最终的根节点。

以上谢谢大家，**求赞求赞求赞！**

❤️大佬们随手关注下我的wx公众号【[甜姨的奇妙冒险](https://pic.leetcode-cn.com/304599b006dd41bcf2042715f31a2dc4fbdc4cf9748a11a81d8978ea1e839956-wxgzh.jpeg)】和 知乎专栏【[甜姨的力扣题解](https://zhuanlan.zhihu.com/c_1224355183452614656)】  
更多题解干货等你来～～

下一篇：C++陆地不断长大，直到覆盖整个地图

© 著作权归作者所有