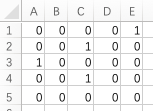
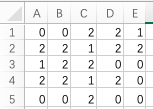
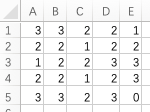
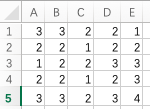


C++陆地不断长大，直到覆盖整个地图

[Lately](https://leetcode-cn.com/u/lately/)发布于 6 小时前1.2kC++广度优先搜索

**解题思路**

如标题所示，将陆地一轮又一轮地扩大，直到覆盖完全，计算扩大了多少轮，输出轮数即可，扩大的地方进行赋值，并根据具体轮数做区分，看我的例子就明白了  
  
（这是初始状况）  
  
（进行第1轮操作,对值为1的陆地判断它上下左右的四个点，如果值为0就进行“长大”，赋值为2）（同时还记录剩余的为0的点数，直到地图上没有0）  
  
（第2轮操作，对值为2的陆地做“长大”，同上，并赋值为3）  
  
（第3轮操作，地图填满了，结束，跳出）（输出轮数3，就是答案,坐标（4，4）值为4的那个点就是要找的最优海洋）

如果有帮到你，就点个赞吧^\_^

**代码**

class Solution {

public:

int maxDistance(vector<vector<int>>& grid) {

int N = grid.size();

int count\_land = 0;

int count\_turn = 0;

int count\_last\_space;

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

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

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

count\_land++;

}

}

}

if(count\_land==N\*N||count\_land==0){

return -1;

}

count\_last\_space = N\*N-count\_land;

while(count\_last\_space!=0){

count\_turn++;

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

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

if(grid[i][j]==count\_turn){

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

grid[i-1][j]=count\_turn+1;

count\_last\_space--;

}

if(i<N-1&&grid[i+1][j]==0){

grid[i+1][j]=count\_turn+1;

count\_last\_space--;

}

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

grid[i][j-1]=count\_turn+1;

count\_last\_space--;

}

if(j<N-1&&grid[i][j+1]==0){

grid[i][j+1]=count\_turn+1;

count\_last\_space--;

}

}

}

}

}

return count\_turn;

}

};//题意是对每个海洋区域，计算到每个陆地的距离并取最短的，在这些最短的区域里找最大的那个；只有陆地或海洋的情况单独拿出来看

下一篇：广度优先遍历（Java）

© 著作权归作者所有