Bidirectional Breadth First Search

双向广度优先搜索

问题：

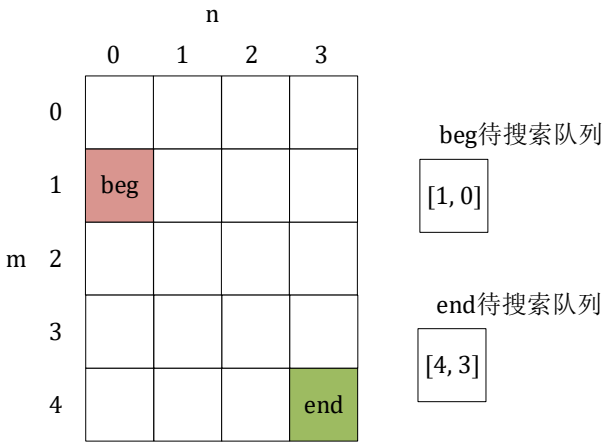
在$m \times n$的二维方格图 s 中从 beg 点移动到 end 点。

解法：

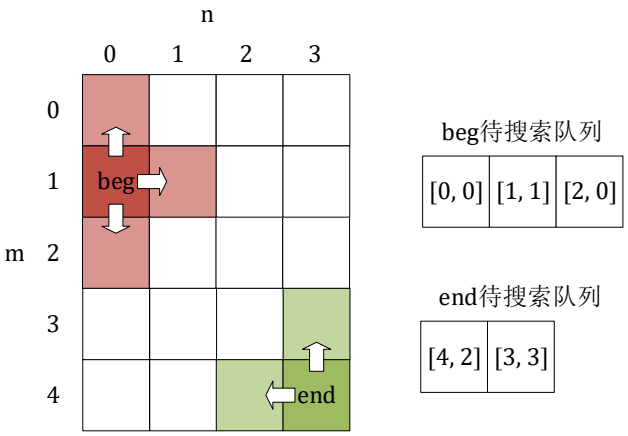
双向广度优先搜索是在广度优先搜索基础上的一个变种，搜索速度更快，内存占用量更大。该算法从 beg 和 end 两个点开始，同时进行广度优先搜索，两边的点在某一处相遇，即可得到一条从 beg 到 end 的路径。

每个广度优先搜索会维护一个已经搜索过的二维方格 visit，$visit[i,j]$为 0 表示该点未被访问过，为 1 表示该点已经被访问过。将 beg 和 end 的两个广度优先搜索的 visit 表分别称为 beg_visit 和 end_visit。当 beg 的队列进行扩张时，不仅检查 beg_visit 表，也检查 end_visit 表，对于某个被扩张的点 x，若其已经在 end_visit 中被访问过，则说明 end 的队列已经到达这里了，则 beg 与 end 在此处相遇，算法结束。

在下面这个$m = 5$，$n = 4$的5×4二维方格 s 中，从$beg = [1,0]$移动到$end = [4,3]$的过程如下：

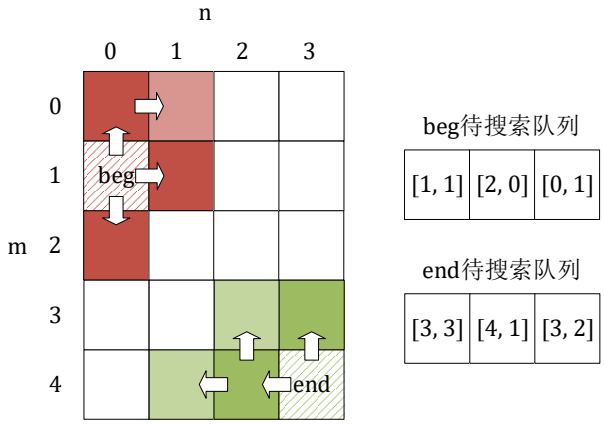


(1) 从 beg 和 end 开始，将 beg 和 end 加入各自的等待队列，beg 染红 end 染绿；

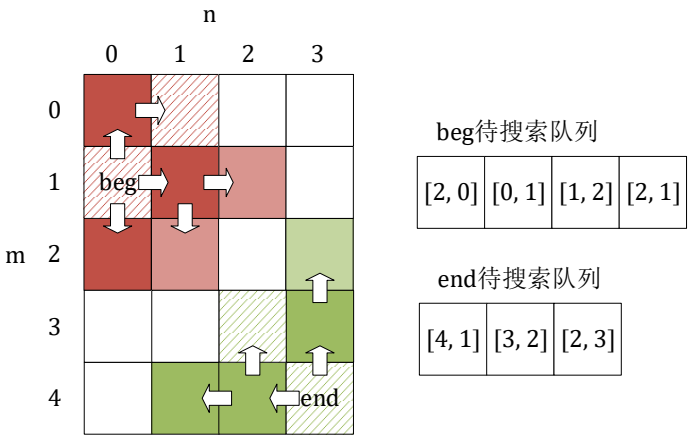


(2) 从 beg 等待队列中取出并检查$[1,0]$不是绿色，将它周围的$[0,0]$、$[1,1]$、$[2,0]$加入 beg 等待队列并染红；从 end 等待队列中取出并检查$[4,3]$不是红色，将它周围的

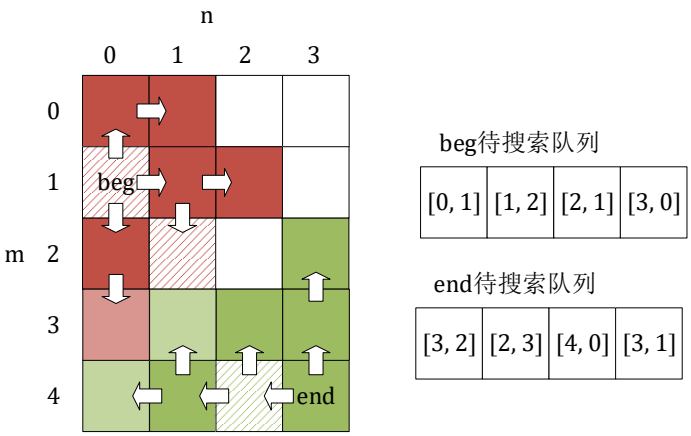
[4, 2]、[3, 3]加入 end 等待队列并染绿；



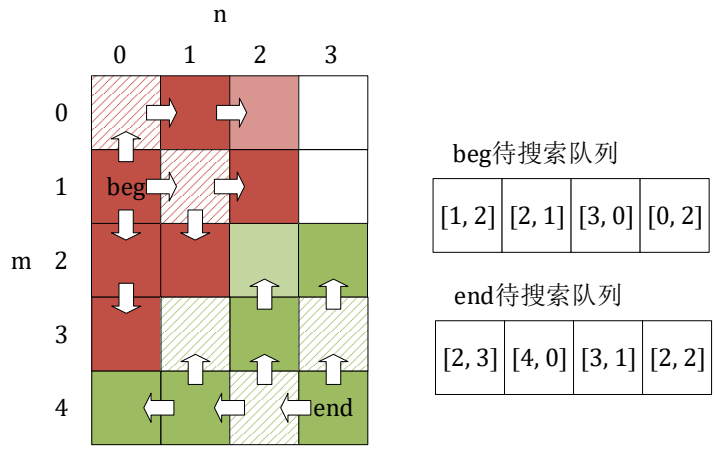
(3) 从 beg 等待队列中取出并检查[0, 0]不是绿色，将它周围的[0, 1]加入 beg 等待队列并染红；从 end 等待队列中取出并检查[4, 2]不是红色，将它周围的[4, 1]、[3, 2]加入 end 等待队列并染绿；



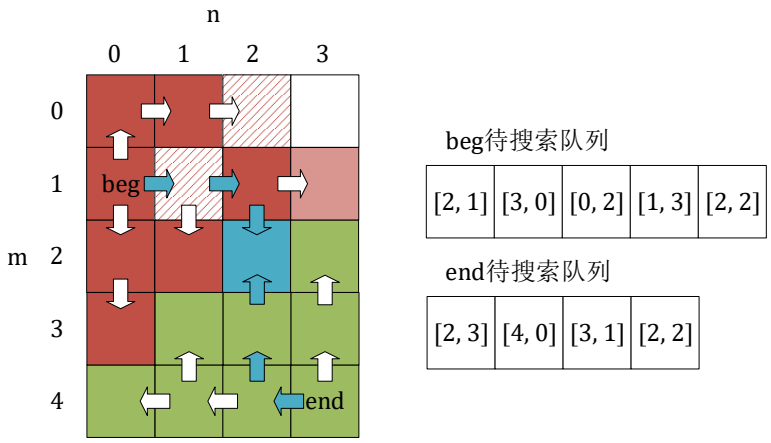
(4) 从 beg 等待队列中取出并检查[1, 1]不是绿色，将它周围的[1, 2]、[2, 1]加入 beg 等待队列并染红；从 end 等待队列中取出并检查[3, 3]不是红色，将它周围的[2, 3]加入 end 等待队列并染绿；



(5) 从 beg 等待队列中取出并检查[2, 0]不是绿色，将它周围的[3, 0]加入 beg 等待队列并染红；从 end 等待队列中取出并检查[4, 1]不是红色，将它周围的[4, 0]、[3, 1]加入 end 等待队列并染绿；

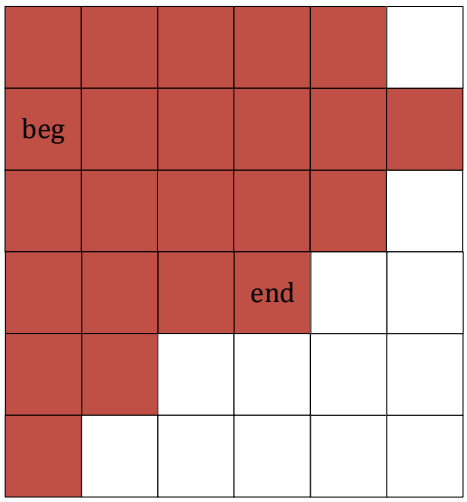


(6) 从 beg 等待队列中取出并检查$[0,1]$不是绿色，将它周围的$[0,2]$加入 beg 等待队列并染红；从 end 等待队列中取出并检查$[3,2]$不是红色，将它周围的$[2,2]$加入 end 等待队列并染绿；

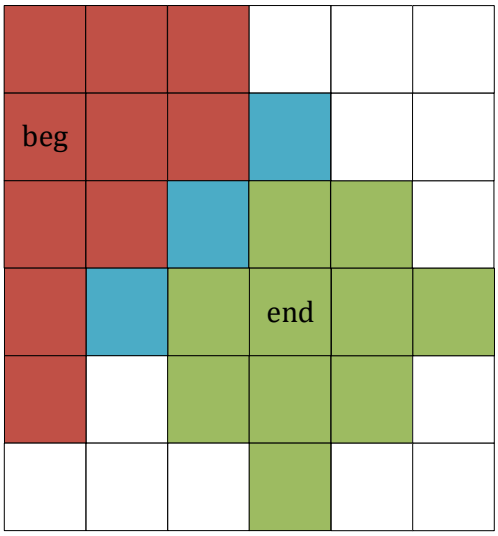


(7) 从 beg 等待队列中取出并检查$[1,2]$不是绿色，将它周围的$[1,3]$、$[2,2]$加入 beg 等待队列并染红，但是$[2,2]$点已经是绿色的了，因此从 beg 和 end 出发的两个广度优先搜索相遇，算法结束；

对于$m \times n$的二维方格 s，广度优先搜索从 beg 点遍历到 end 点的过程一般是从 beg 向四周发散开，一直到达 end 点：



而双向广度优先搜索则是从 beg 和 end 两个点分别发散开，在中间相遇：



假设 beg 和 end 点距离为 k，节点的邻居数量为 b，在本问题中，每个节点有上下左右 4 个邻居，即$b = 4$。则广度优先搜索会考虑的节点数量为$1 + b + b^2 + \cdots + b^k$，而双向广度优先搜索会考虑的节点数量是$2 + 2b + 2b^2 + \cdots + 2b^{\frac{k}{2}}$，最坏情况下时间复杂度与广度优先搜索一样也为$O(m \times n)$。