Breadth First Search

广度优先搜索

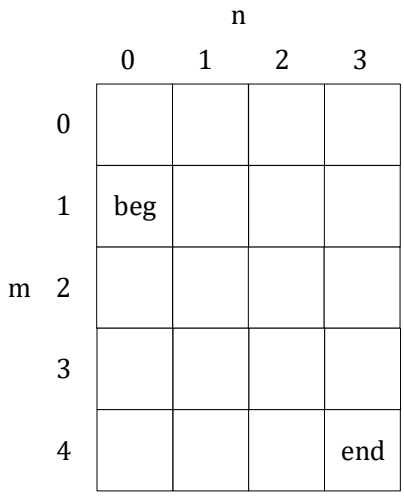
问题：

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

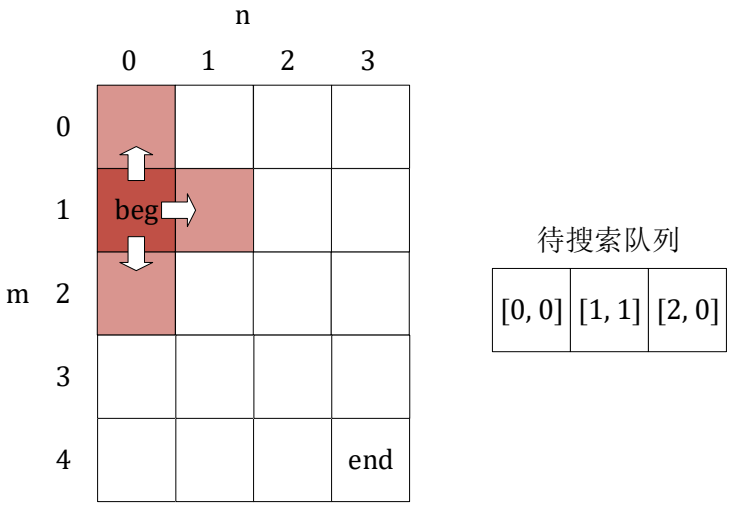
解法：

广度优先搜索是优先搜索二维方格图 s 中每个节点的相邻节点，与之相对的深度优先搜索则会沿着节点的一个相邻节点试图走到最远。

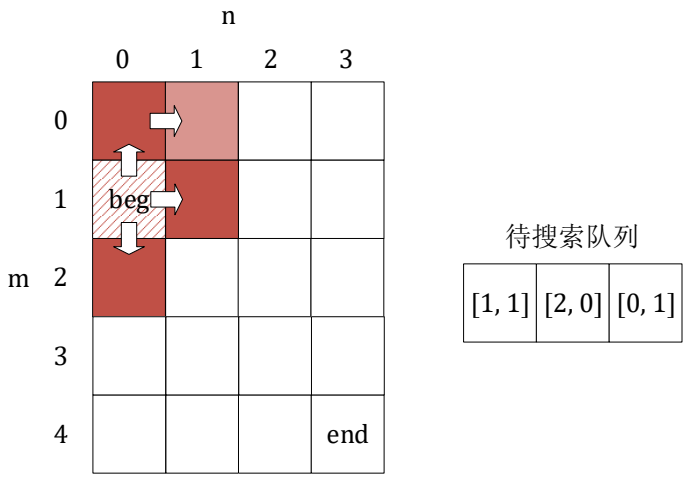
在下面这个$m = 5$，$n = 4$的5×4二维方格 s 中，从$beg = [1,0]$移动到$end = [4,3]$：



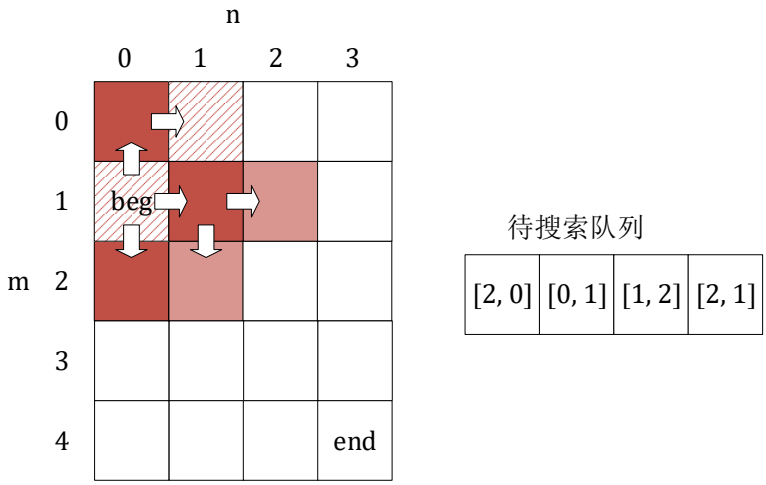
(1) $beg \neq end$，将 beg 染红并继续搜索 beg 的上下左右四个相邻点，其中超出边界的、不存在的或已经染红的点不考虑，将$[0,0]$、$[1,1]$和$[2,0]$加入等待搜索的队列中，并全部染红；



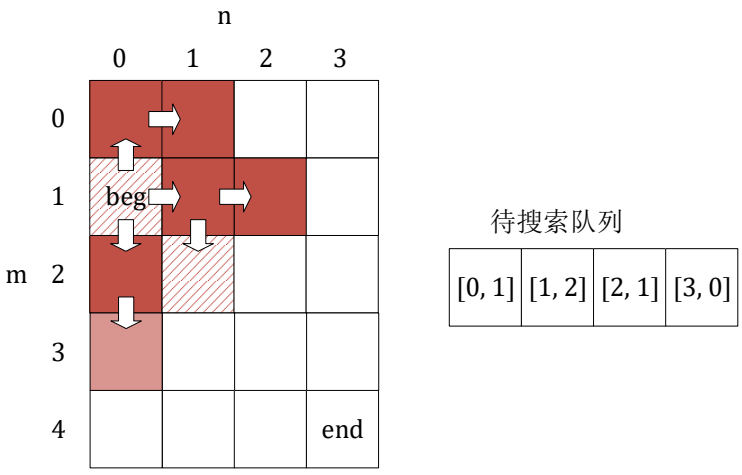
(2) 从等待队列中取出并比较$[0,0] \neq end$，继续搜索$[0,0]$相邻的点，将$[0,1]$加入等待队列中并染红；



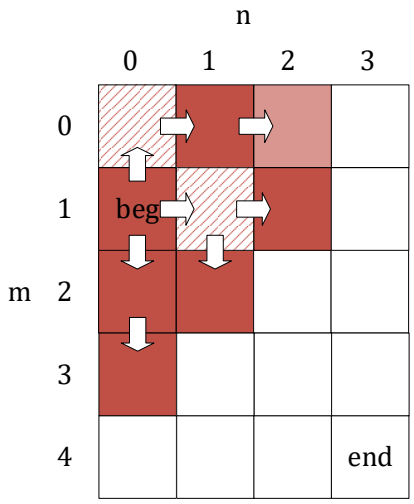
(3) 从等待队列中取出并比较$[1,1] \neq end$，继续搜索$[1,1]$相邻的点，因为 beg 和$[0,1]$是红色的，因此不加入等待队列，将$[1,2]$、$[2,1]$加入等待队列中并染红；



(4) 从等待队列中取出并比较$[2,0] \neq end$，继续搜索$[2,0]$相邻的点，因为 beg 和$[2,1]$是红色的，因此不加入等待队列，将$[3,0]$加入等待队列中并染红；



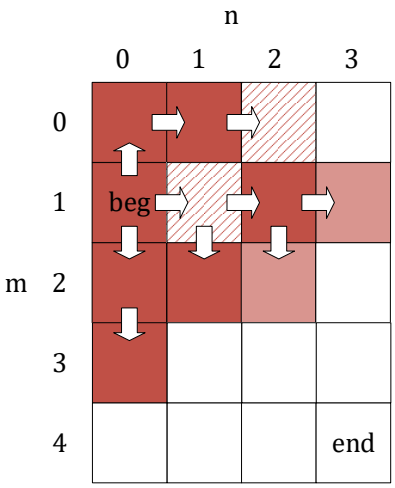
(5) 从等待队列中取出并比较$[0,1] \neq end$，继续搜索$[0,1]$相邻的点，因为$[0,0]$和$[1,1]$是红色的，因此不加入等待队列，将$[0,2]$加入等待队列中并染红；



待搜索队列

[1, 2]	[2, 1]	[3, 0]	[0, 2]

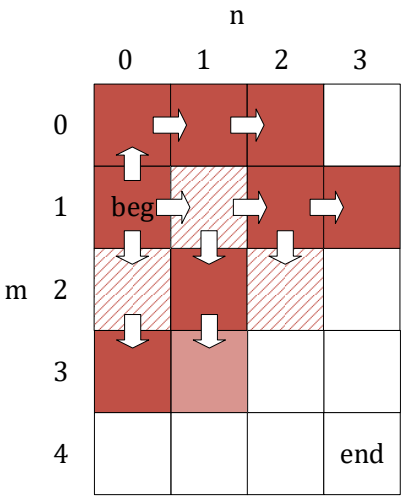
(6) 从等待队列中取出并比较$[1,2] \neq end$，继续搜索$[1,2]$相邻的点，因为$[1,1]$和$[0,2]$是红色的，因此不加入等待队列，将$[1,3]$和$[2,2]$加入等待队列中并染红；



待搜索队列

[2, 1]	[3, 0]	[0, 2]	[1, 3]	[2, 2]

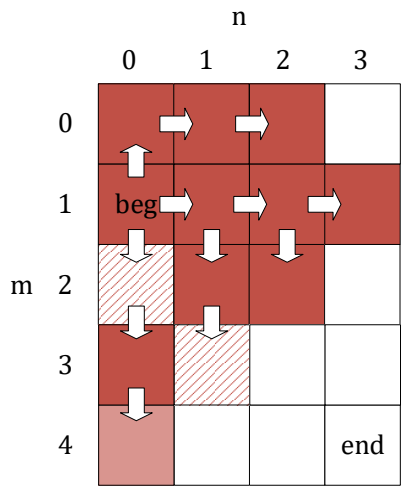
(7) 从等待队列中取出并比较$[2,1] \neq end$，继续搜索$[2,1]$相邻的点，因为$[2,0]$和$[1,1]$是红色的，因此不加入等待队列，将$[3,1]$加入等待队列中并染红；



待搜索队列

[3, 0]	[0, 2]	[1, 3]	[2, 2]	[3, 1]

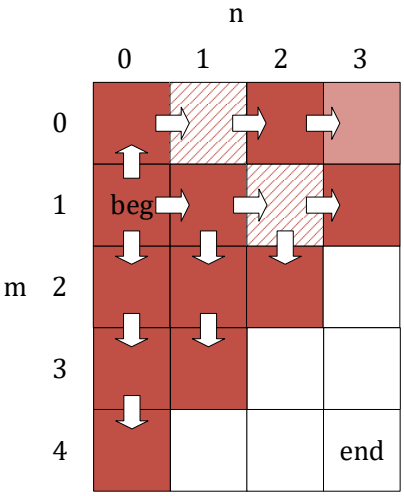
(8) 从等待队列中取出并比较$[3,0] \neq end$，继续搜索$[3,0]$相邻的点，因为$[2,0]$和$[3,1]$是红色的，因此不加入等待队列，将$[4,0]$加入等待队列中并染红；



待搜索队列

[0, 2]	[1, 3]	[2, 2]	[3, 1]	[4, 0]

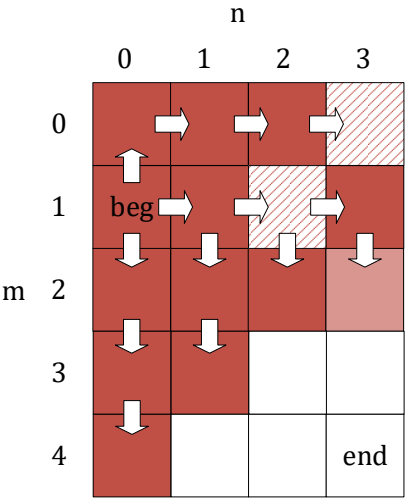
(9) 从等待队列中取出并比较$[0,2] \neq end$，继续搜索$[0,2]$相邻的点，因为$[0,1]$和$[1,2]$是红色的，因此不加入等待队列，将$[0,3]$加入等待队列中并染红；



待搜索队列

[1, 3]	[2, 2]	[3, 1]	[4, 0]	[0, 3]

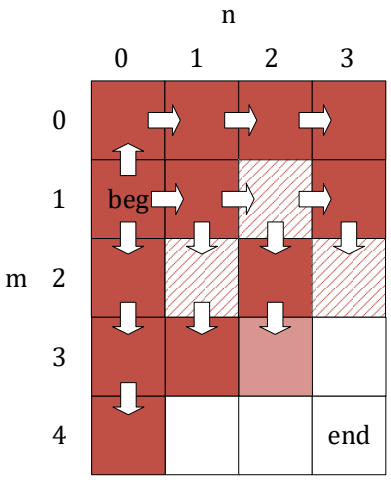
(10)从等待队列中取出并比较$[1,3] \neq end$，继续搜索$[1,3]$相邻的点，因为$[1,2]$和$[0,3]$是红色的，因此不加入等待队列，将$[2,3]$加入等待队列中并染红；



待搜索队列

[2, 2]	[3, 1]	[4, 0]	[0, 3]	[2, 3]

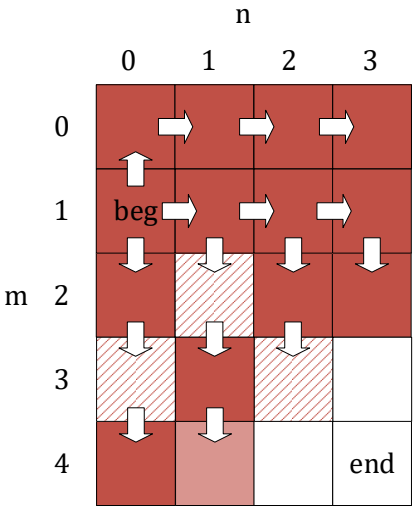
(11)从等待队列中取出并比较$[2,2] \neq end$，继续搜索$[2,2]$相邻的点，因为$[2,1]$、$[1,2]$和$[2,3]$是红色的，因此不加入等待队列，将$[3,2]$加入等待队列中并染红；



待搜索队列

[3, 1]	[4, 0]	[0, 3]	[2, 3]	[3, 2]

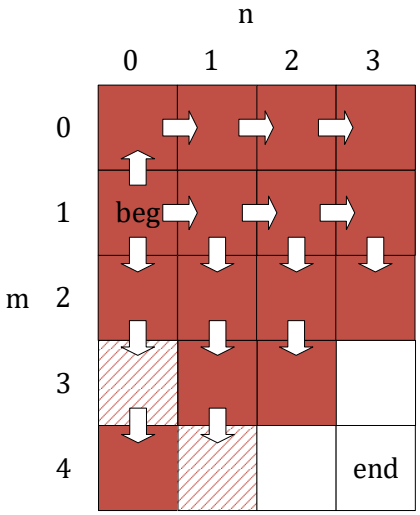
(12)从等待队列中取出并比较$[3,1] \neq end$，继续搜索[3,1]相邻的点，因为[3,0]、[2,1]和[3,2]是红色的，因此不加入等待队列，将[4,1]加入等待队列中并染红；



待搜索队列

[4, 0]	[0, 3]	[2, 3]	[3, 2]	[4, 1]

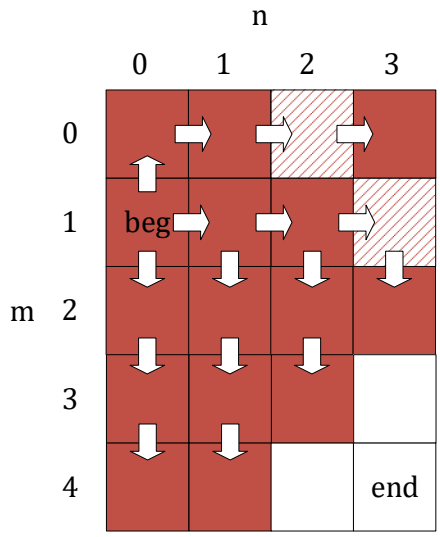
(13)从等待队列中取出并比较$[4,0] \neq end$，[4,0]相邻的所有点都是红色，因此没有新的点加入等待队列；



待搜索队列

[0, 3]	[2, 3]	[3, 2]	[4, 1]

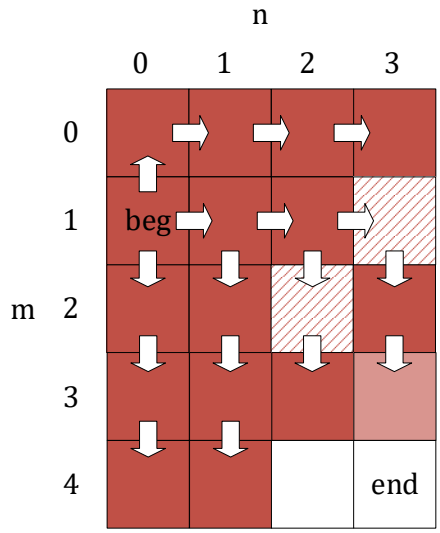
(14)从等待队列中取出并比较$[0,3] \neq end$，[0,3]相邻的所有点都是红色，因此没有新的点加入等待队列；



待搜索队列

[2, 3]	[3, 2]	[4, 1]

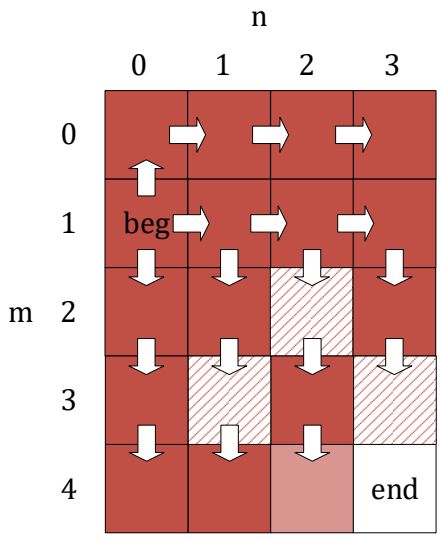
(15)从等待队列中取出并比较$[2,3] \neq end$，继续搜索[2,3]相邻的点，因为[2,2]、[1,3]是红色的，因此不加入等待队列，将[3,3]加入等待队列中并染红；



待搜索队列

[3, 2]	[4, 1]	[3, 3]

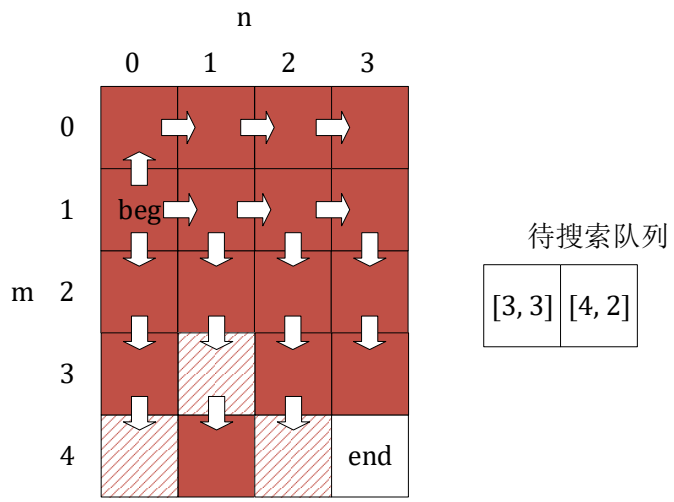
(16)从等待队列中取出并比较$[3,2] \neq end$，继续搜索[3,2]相邻的点，因为[3,1]、[2,2]和[3,3]是红色的，因此不加入等待队列，将[4,2]加入等待队列中并染红；



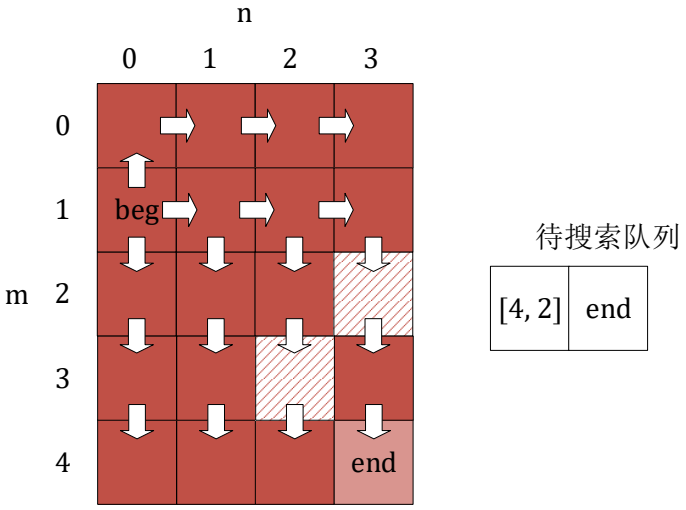
待搜索队列

[4, 1]	[3, 3]	[4, 2]

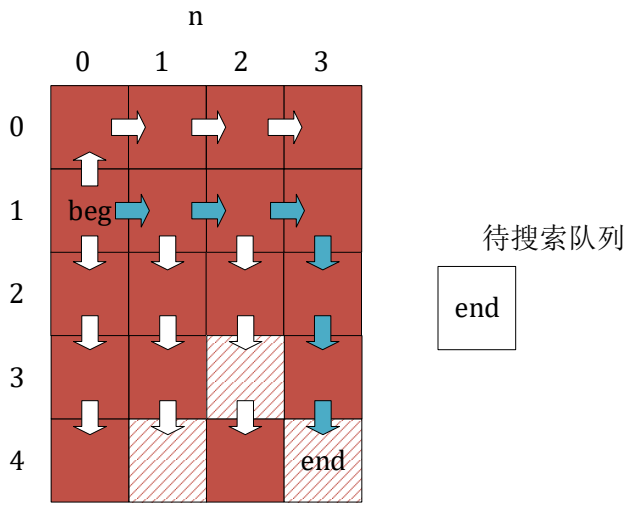
(17)从等待队列中取出并比较$[4,1] \neq end$，$[4,1]$相邻的所有点都是红色，因此没有新的点加入等待队列；



(18)从等待队列中取出并比较$[3,3] \neq end$，继续搜索$[3,3]$相邻的点，因为$[2,3]$、$[3,2]$是红色的，因此不加入等待队列，将 end 加入等待队列中并染红；



(19)从等待队列中取出并比较$[4,2] \neq end$，$[4,2]$相邻的所有点都是红色，因此没有新的点加入等待队列；



(20)从等待队列中取出并比较$[4,3] = end$即为所求，算法结束。如果需要额外的获取从 beg 点到 end 点的路径，则需要在遍历时标记经过的点的上一个点，即其父节点，最终可以通过父节点指针逆向找到一条回到 beg 点的路径，如果不需要获取路径，则不需要记录父节点指针；

对于$m \times n$的二维方格 s，从 beg 点遍历到 end 点，最坏情况下时间复杂度为$O(m \times n)$。