

```

1  /*
2  给出 R 行 C 列的矩阵，其中的单元格的整数坐标为 (r, c)，满足 0 <= r < R 且 0 <= c < C。
3  另外，我们在该矩阵中给出了一个坐标为 (r0, c0) 的单元格。
4  返回矩阵中的所有单元格的坐标，并按到 (r0, c0) 的距离从最小到最大的顺序排，其中，两单元格 (r1, c1) 和 (r2, c2) 之间的距离是曼哈顿距离，|r1 - r2| + |c1 - c2|。（你可以按任何满足此条件的顺序返回答案。）
5  示例 1:
6  输入: R = 1, C = 2, r0 = 0, c0 = 0
7  输出: [[0,0],[0,1]]
8  解释: 从 (r0, c0) 到其他单元格的距离为: [0,1]
9  示例 2:
10 输入: R = 2, C = 2, r0 = 0, c0 = 1
11 输出: [[0,1],[0,0],[1,1],[1,0]]
12 解释: 从 (r0, c0) 到其他单元格的距离为: [0,1,1,2]
13 [[0,1],[1,1],[0,0],[1,0]] 也会被视作正确答案。
14 示例 3:
15 输入: R = 2, C = 3, r0 = 1, c0 = 2
16 输出: [[1,2],[0,2],[1,1],[0,1],[1,0],[0,0]]
17 解释: 从 (r0, c0) 到其他单元格的距离为: [0,1,1,2,2,3]
18 其他满足题目要求的答案也会被视为正确，例如 [[1,2],[1,1],[0,2],[1,0],[0,1],[0,0]]。
19 提示:
20     1 <= R <= 100
21     1 <= C <= 100
22     0 <= r0 < R
23     0 <= c0 < C
24 来源：力扣（LeetCode）
25 链接：https://leetcode-cn.com/problems/matrix-cells-in-distance-order
26 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。
27 */

```

分析:

- 通过画图可知道,与 (r_0, c_0) 点距离为 d 的点是4条直线围成的正方形,我们只要依次枚举这个4条直线上的每个点,判断每个点的坐标是否在允许范围内即可.

方法一:C++_直线方程遍历法

```

1  class Solution
2  {
3      bool check_point_valid(int R, int C, int x, int y)
4      {
5          return ( (x>=0) && (x<R) && (y>=0) && ( y < C) );
6      }
7  }
8
9
10 public:
11     vector<vector<int>> allCellsDistOrder(int R, int C, int r0, int
c0)
12     {
13         vector<vector<int>> vvi_ret ;
14         vector<int> vi ;

```

```

15         int                edge_distance[4]          =  {0,}      ;    //
16         int                max_distance              =  0          ;
17         int                i                        =  0          ;
18         int                j                        =  0          ;
19         int                x                        =  0          ;
20         int                y                        =  0          ;
21
22         /*计算最长距离*/
23         edge_distance[0]    =  abs(0-r0)  +  abs(0-c0);
24         edge_distance[1]    =  abs(0-r0)  +  abs(C-1-c0);
25         edge_distance[2]    =  abs(R-1-r0) +  abs(0-c0);
26         edge_distance[3]    =  abs(R-1-r0) +  abs(C-1-c0);
27
28         max_distance = edge_distance[0];
29         for(i = 1; i < 4; i++)
30         {
31             if(max_distance < edge_distance[i])
32             {
33                 max_distance = edge_distance[i];
34             }
35         }
36
37
38         /*按距离从小到大遍历(菱形)*/
39         vi.push_back(r0);
40         vi.push_back(c0);
41         vvi_ret.push_back(vi);
42         vi.clear();
43         for(i=1; i<=max_distance; i++)
44         {
45             /*
46             Left-->Bottom:(r0,c0-i) ---> (r0+i,c0),不到bottom点
47             */
48             for(j=0; j<i; j++)
49             {
50                 x = r0 + j;    //这里的x表示行下标 , y表示列下表
51                 y = c0 - i + j;
52                 if(check_point_valid(R,C,x,y))
53                 {
54                     vi.push_back(x);
55                     vi.push_back(y);
56                     vvi_ret.push_back(vi);
57                     vi.clear();
58                 }
59             }
60
61             /*
62             Bottom-->Right:(r0+i,c0) --->(r0,c0+i),不到right点
63             */
64             for(j=0; j<i; j++)
65             {
66                 x = r0 + i - j;
67                 y = c0 + j;
68                 if(check_point_valid(R,C,x,y))
69                 {
70                     vi.push_back(x);
71                     vi.push_back(y);

```

```

72         vvi_ret.push_back(vi);
73         vi.clear();
74     }
75 }
76
77 /*
78 Right-->Top: (r0,c0+i) ---->(r0-i,c0), 不到top点
79 */
80 for(j=0;j<i;j++)
81 {
82     x = r0 - j;
83     y = c0 + i - j ;
84     if(check_point_valid(R,C,x,y))
85     {
86         vi.push_back(x);
87         vi.push_back(y);
88         vvi_ret.push_back(vi);
89         vi.clear();
90     }
91 }
92
93 /*
94 Top-->Left: (r0-i,c0) ---->(r0,c0-i) , 不到top点
95 */
96 for(j=0;j<i;j++)
97 {
98     x = r0 - i + j;
99     y = c0 - j ;
100    if(check_point_valid(R,C,x,y))
101    {
102        vi.push_back(x);
103        vi.push_back(y);
104        vvi_ret.push_back(vi);
105        vi.clear();
106    }
107 }
108 }
109 return vvi_ret;
110 }
111 };
112
113 /*
114 执行结果:
115 通过
116 显示详情
117 执行用时 :124 ms, 在所有 cpp 提交中击败了98.05% 的用户
118 内存消耗 :17.3 MB, 在所有 cpp 提交中击败了100.00%的用户
119 */

```