```
            int[][] map, Queue<Integer> rQ, Queue<Integer> cQ) {
        if (toR < 0 || toR == m.length || toC < 0 || toC == m[0].length
                || m[toR][toC] != 1 || map[toR][toC] != 0) {
            return;
        }
        map[toR][toC] = pre + 1;
        rQ.add(toR);
        cQ.add(toC);
    }
```

数组中未出现的最小正整数

【题目】

给定一个无序整型数组 arr，找到数组中未出现的最小正整数。

【举例】

arr=[-1,2,3,4]。返回 1。

arr=[1,2,3,4]。返回 5。

【难度】

尉 ★★☆☆

【解答】

原问题。如果 arr 长度为 N，本题的最优解可以做到时间复杂度为 $O(N)$，额外空间复杂度为 $O(1)$。具体过程如下：

1．在遍历 arr 之前先生成两个变量。变量 l 表示遍历到目前为止，数组 arr 已经包含的正整数范围是[1,l]，所以没有开始遍历之前令 l=0，表示 arr 目前没有包含任何正整数。变量 r 表示遍历到目前为止，在后续出现最优状况的情况下，arr 可能包含的正整数范围是[1,r]，所以没有开始遍历之前，令 r=N，因为还没有开始遍历，所以后续出现的最优状况是 arr 包含 1~N 所有的整数。r 同时表示 arr 当前的结束位置。

2．从左到右遍历 arr，遍历到位置 l，位置 l 的数为 arr[l]。

3．如果 arr[l]==l+1。没有遍历 arr[l]之前，arr 已经包含的正整数范围是[1,l]，此时出现了 arr[l]==l+1 的情况，所以 arr 包含的正整数范围可以扩到[1,l+1]，即令 l++。然后重复步骤 2。

4．如果 arr[l]<=l。没有遍历 arr[l]之前，arr 在后续最优的情况下可能包含的正整数范围是[1,*r*]，已经包含的正整数范围是[1,*l*]，所以需要[*l*+1,*r*]上的数。而此时出现了 arr[l]<=l，说明[*l*+1,*r*]范围上的数少了一个，所以 arr 在后续最优的情况下，可能包含的正整数范围缩小了，变为[1,*r*-1]，此时把 arr 最后位置的数(arr[r-1])放在位置 *l* 上，下一步检查这个数，然后令 r--。重复步骤 2。

5．如果 arr[l]>r，与步骤 4 同理，把 arr 最后位置的数(arr[r-1])放在位置 *l* 上，下一步检查这个数，然后令 r--。重复步骤 2。

6．如果 arr[arr[l]-1]==arr[l]。如果步骤 4 和步骤 5 没中，说明 arr[l]是在[*l*+1,*r*]范围上的数，而且这个数应该放在 arr[l]-1 位置上。可是此时发现 arr[l]-1 位置上的数已经是 arr[l]，说明出现了两个 arr[l]，既然在[*l*+1,*r*]上出现了重复值，那么[*l*+1,*r*]范围上的数又少了一个，所以与步骤 4 和步骤 5 一样，把 arr 最后位置的数(arr[r-1])放在位置 *l* 上，下一步检查这个数，然后令 r--。重复步骤 2。

7．如果步骤 4、步骤 5 和步骤 6 都没中，说明发现了[*l*+1,r]范围上的数，并且此时并未发现重复。那么 arr[l]应该放到 arr[l]-1 位置上，所以把 *l* 位置上的数和 arr[l]-1 位置上的数交换，下一步继续遍历 *l* 位置上的数。重复步骤 2。

8．最终 *l* 位置和 *r* 位置会碰在一起（l==r），arr 已经包含的正整数范围是[1,*l*]，返回 *l*+1 即可。

具体过程请参看如下代码中的 missNum 方法。

```
public int missNum(int[] arr) {
    int l = 0;
    int r = arr.length;
    while (l < r) {
        if (arr[l] == l + 1) {
                l++;
        } else if (arr[l] <= l || arr[l] > r || arr[arr[l] - 1] == arr[l]) {
                arr[l] = arr[--r];
        } else {
                swap(arr, l, arr[l] - 1);
        }
    }
    return l + 1;
}
```