1.移动零

「数组分两块」是非常常见的一种题型，主要就是根据一种划分方式，将数组的内容分成左右两部分。这种类型的题，一般就是使用「双指针」来解决

题目链接： 283.移动零

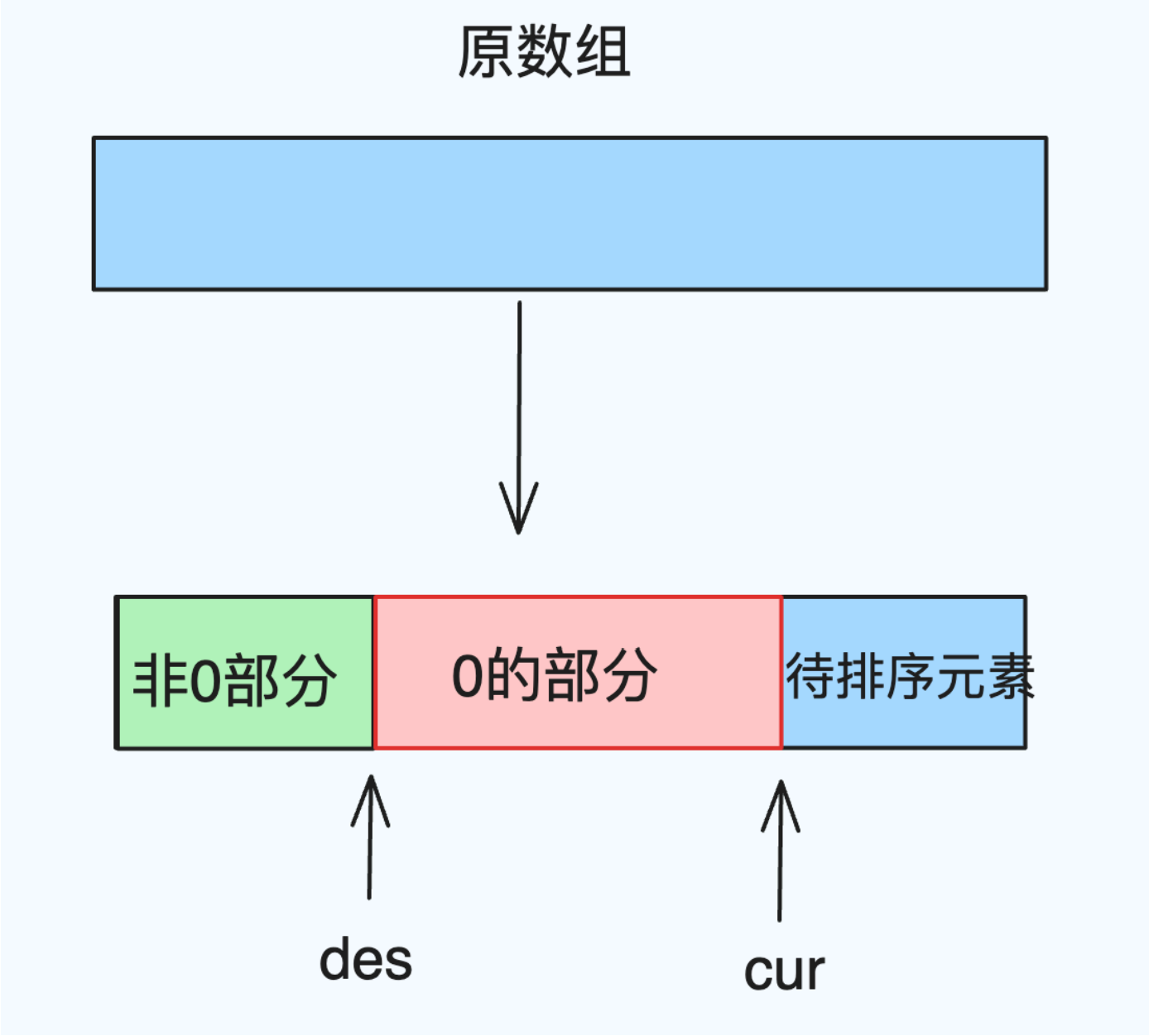
给定一个数组 nums，编写一个函数将所有 0 移动到数组的末尾，同时保持非零元素的相对顺序

请注意 ，必须在不复制数组的情况下原地对数组进行操作。

解法：（快排的思想：数组划分区间-数组分两块）：

算法思路：

在本题中，我们可以用一个 cur 指针来扫描整个数组，另一个 dest 指针用来记录非零数序列的最后一个位置。根据cur在扫描的过程中，遇到不同的情况，分类处理，实现数组的划分。以下是我们的目标



我们如何实现这种呢?

1.cur 指针的目的是遍历数组，那么cur指针一定是在数组首元素位置

2.既然cur指针在首元素，为了实现数组被划分三个阶段，那么des只能在cur之前的位置也就是 -1 处。

3.cur指针开始向后移动，为了实现【des+1 ， cur-1】中间是0，那么控制cur的条件就是，cur遇到0就跳过，也就是继续往前走。如果遇到了不是0，那么就将des+1，进行交换，交换后cur当前位置就变成了0，继续加加，直到遍历完数组。

因此可以简化思想：

cur 从前往后遍历过程中：

1.遇到0元素：cur++

2.遇到非0元素：1️⃣swap(++des;cur) 2️⃣ cur++

代码编写：

```
void moveZeroes(vector<int>& nums) {
    int cur = 0;
    int des = -1;
    while(cur < nums.size()){
        //cur是0就跳过
        if(nums[cur] == 0)
            cur++;
        else{
            //不是0就交换，在++
            swap(nums[++des],nums[cur]);
            cur++;
        }
    }
}
```

也可以用for循环

```
void moveZeroes(vector<int>& nums) {
        for(int cur = 0,des = -1;cur < nums.size();cur++){
            if(nums[cur])//处理非0元素
            {
                swap(nums[++des],nums[cur]);
            }
        }
    }
```
