BITTIGER

CLASS_2
LINEAR STRUCTURE

Content of Class_2

75. Sort Colors	438. Find All Anagrams in a String	189. Rotate Array
11. Container With Most Water	155. Min Stack	48. Rotate Image

75. Sort Colors

Given an array with *n* objects colored red, white or blue, sort them so that objects of the same color are adjacent, with the colors in the order red, white and blue.

Here, we will use the integers 0, 1, and 2 to represent the color red, white, and blue respectively.

0—->放置在队首 2——->放置在队尾

0 —> 1 —> 2 排序！ O(nlogn)

0交换到队首，2交换到队尾 —> 双指针 O(n)

10 min

```
public void sortColors(int[] nums)

}
```

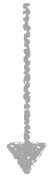
```java
public void sortColors(int[] nums) {
    if(nums == null || nums.length == 0){
        return;
    }

    int len = nums.length;
    int pre = 0;
    int end = len - 1;

    for(int i = 0; i <= end; i++){
        if(nums[i] == 0){
            swap(nums, i , pre);
            pre++;
        }else if(nums[i] == 2){
            swap(nums, i, end);
            end--;
            i--;
        }
    }

    return;
}

public void swap(int[] nums, int i , int j){
    int temp = nums[i];
    nums[i] = nums[j];
    nums[j] = temp;
}
```

双指针

0 向队首交换

2 向队尾交换 为什么 i-- ?

四色，五色，六色 或者 不规则排序 （0， 2， 1， 3）？



无论颜色如何排列，无论有多少种颜色 ——> 颜色数量固定

计数器分别记录不同颜色的个数，再重写array O(n)

当已知数字分别情况时，排序时间复杂度 O(n)

```java
public void sortColors(int[] nums) {
    if(nums == null || nums.length == 0){
        return;
    }

    int len = nums.length;
    int zero = 0;
    int one = 0;
    int two = 0;

    for(int i = 0; i < len; i++){
        if(nums[i] == 0){
            zero++;
        }else if(nums[i] == 1){
            one++;
        }else if(nums[i] == 2){
            two++;
        }
    }

    for(int i = 0; i < zero; i++){
        nums[i] = 0;
    }

    for(int i = zero; i < zero + one; i++){
        nums[i] = 1;
    }

    for(int i = zero + one; i < len; i++){
        nums[i] = 2;
    }

    return;
}
```

计数器

统计数字频率

分别覆盖，注意起始和结束位置

438. Find All Anagrams in a String

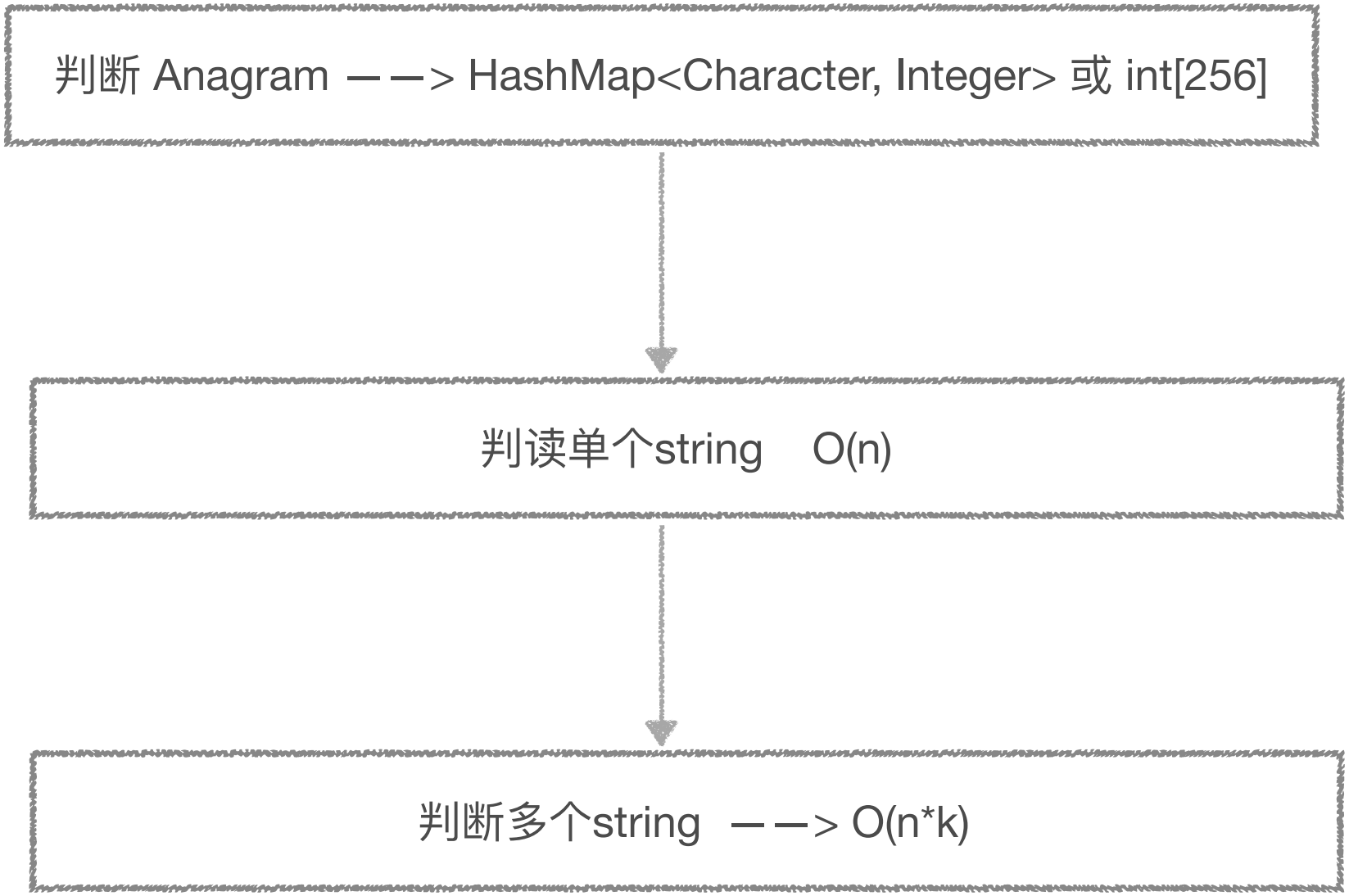
Given a string **s** and a **non-empty** string **p**, find all the start indices of **p**'s anagrams in **s**.

Strings consists of lowercase English letters only and the length of both strings **s** and **p** will not be larger than 20,100.

The order of output does not matter.

Example 1:

```
Input:
s: "cbaebabacd" p: "abc"

Output:
[0, 6]

Explanation:
The substring with start index = 0 is "cba", which is an anagram of "abc".
The substring with start index = 6 is "bac", which is an anagram of "abc".
```


判断 Anagram ——> HashMap<Character, Integer> 或 int[256]
判读单个string O(n)
判断多个string ——> O(n*k)

多个string 判断时会发生重复计数

滑动窗口（sliding window）定长扫描——> 无重复计数

10 min

```
public List<Integer> findAnagrams(String s, String p) {

}
```

```java
public List<Integer> findAnagrams(String s, String p) {
    List<Integer> res = new ArrayList<>();
    if(s == null || s.length() == 0 || p == null || p.length() == 0){
        return res;
    }

    int[] map = new int[256];

    for(char cur: p.toCharArray()){
        map[cur]++;
    }

    int left = 0;
    int right = 0;
    int counter = p.length();

    while(right < s.length()){

        if(map[s.charAt(right)] > 0){
            counter--;
        }

        map[s.charAt(right)]--;
        right++;

        if(counter == 0){
            res.add(left);
        }

        if(right - left == p.length()){
            if(map[s.charAt(left)] >= 0){
                counter++;
            }
            map[s.charAt(left)]++;
            left++;
        }
    }

    return res;
}
```

经典统计词频的方法

right 指针 右移计数

left 指针 左移计数

189. Rotate Array

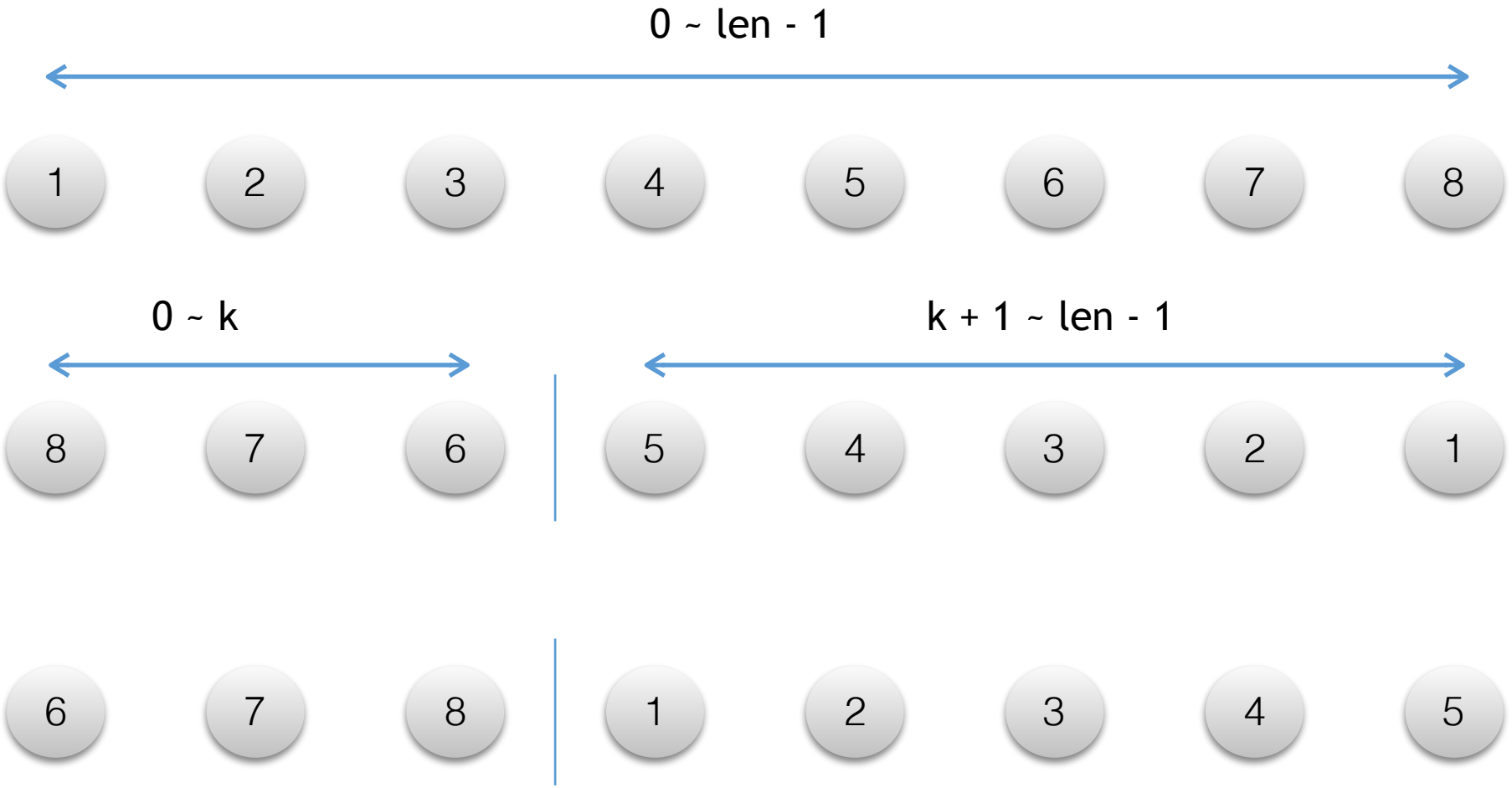
Rotate an array of *n* elements to the right by *k* steps.

For example, with *n* = 7 and *k* = 3, the array `[1,2,3,4,5,6,7]` is rotated to `[5,6,7,1,2,3,4]`.

Note:

Try to come up as many solutions as you can, there are at least 3 different ways to solve this problem.

经典的 三步反转



10 min

```
public void rotate(int[] nums, int k) {
}
```

```java
public void rotate(int[] nums, int k) {
    if(nums == null || nums.length <= 1){
        return;
    }
    k = k % nums.length;
    swap(nums, 0, nums.length - 1);
    swap(nums, 0, k - 1);
    swap(nums, k, nums.length - 1);
}

public void swap(int[] nums, int beg, int end){
    while(beg < end){
        int temp = nums[beg];
        nums[beg] = nums[end];
        nums[end] = temp;
        beg++;
        end--;
    }
}
```

三步反转 注意起始和结束位置

48. Rotate Image

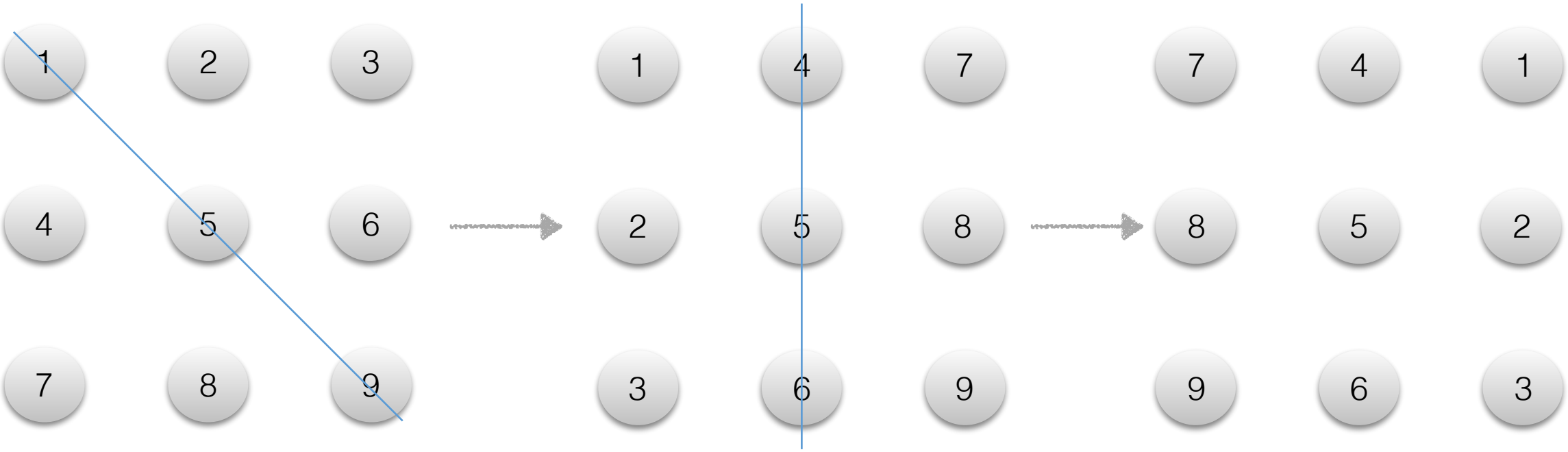
You are given an *n* x *n* 2D matrix representing an image.

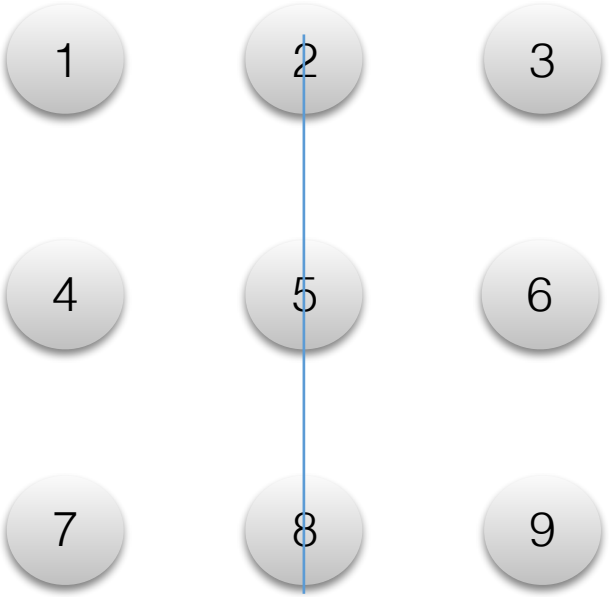
Rotate the image by 90 degrees (clockwise).

Follow up:

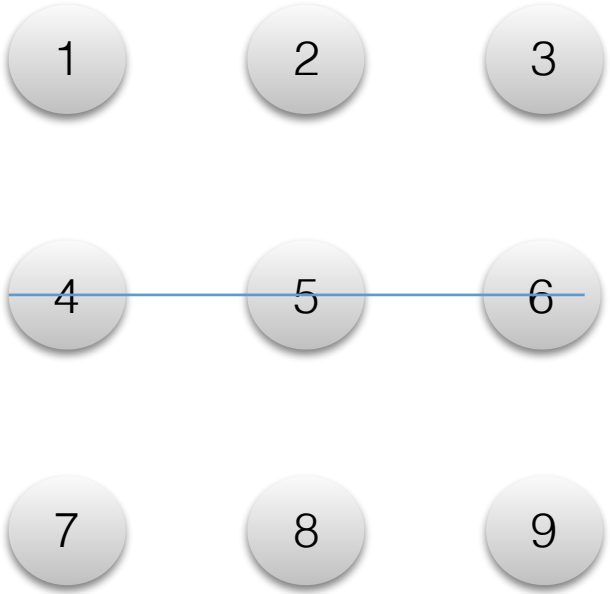
Could you do this in-place?

使用翻折操作组合来实现旋转操作

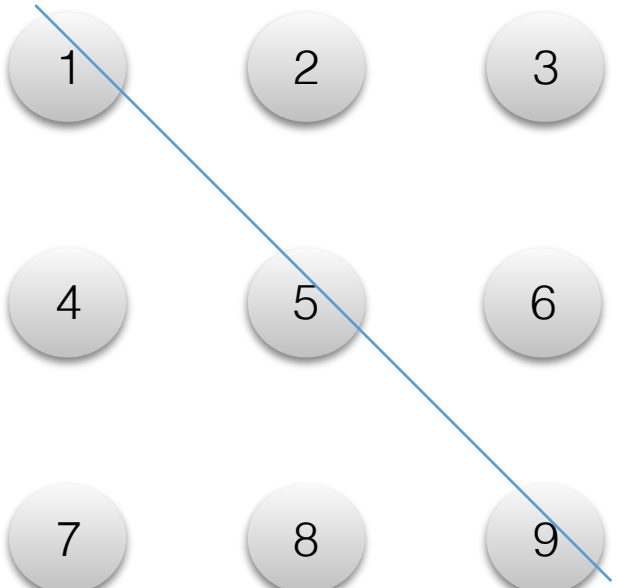




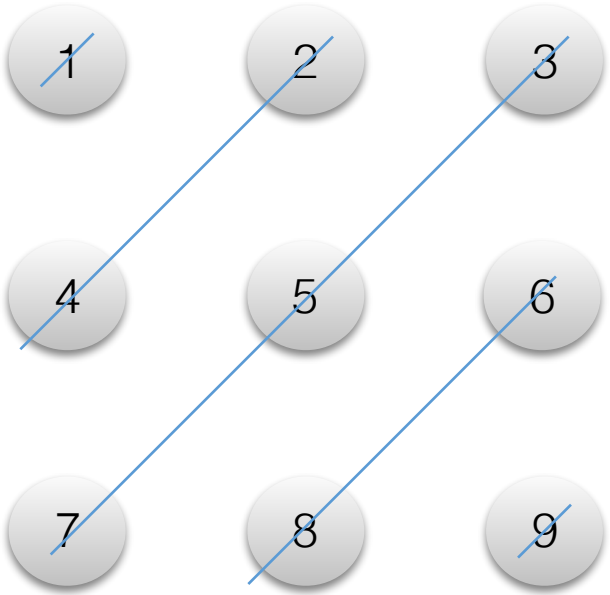
matrix[i][j] = matrix[i][col - j - 1]



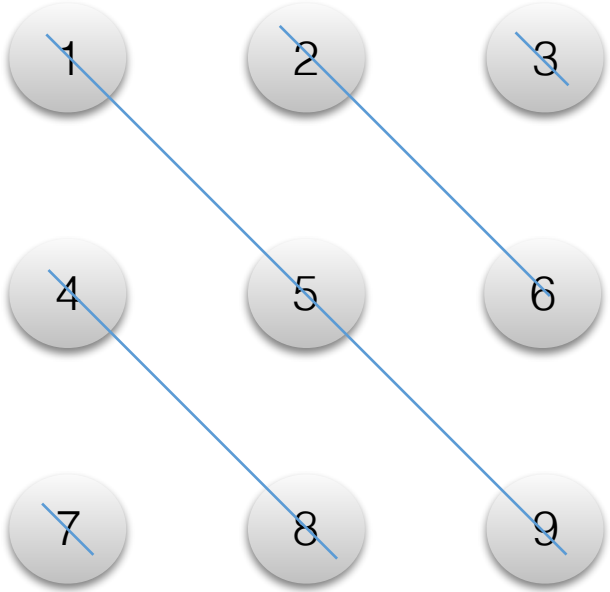
matrix[i][j] = matrix[row - i - 1][j]



matrix[i][j] = matrix[j][i]



同轴 i + j = k (k = 0, 1, 2..row + col - 2)



同轴 i - j = k (k = - col +1..0… row - 1)

10 min

```
public void rotate(int[][] matrix) {
}
```

```java
public void rotate(int[][] matrix) {
    if(matrix == null || matrix.length == 0 || matrix[0].length == 0){
        return;
    }

    int row = matrix.length;
    int col = matrix[0].length;

    for(int i = 0; i < row; i++){
        for(int j = 0; j <= i; j++){
            swap(matrix, i, j, j, i);
        }
    }

    for(int i = 0; i < row; i++){
        for(int j = 0; j < col / 2; j++){
            swap(matrix, i, j, i, col - j - 1);
        }
    }

    return;
}

public void swap(int[][] matrix, int i, int j, int x, int y){
    int temp = matrix[i][j];
    matrix[i][j] = matrix[x][y];
    matrix[x][y] = temp;
}
```

对角线 反转

中轴 反转 注意起始和结束位置

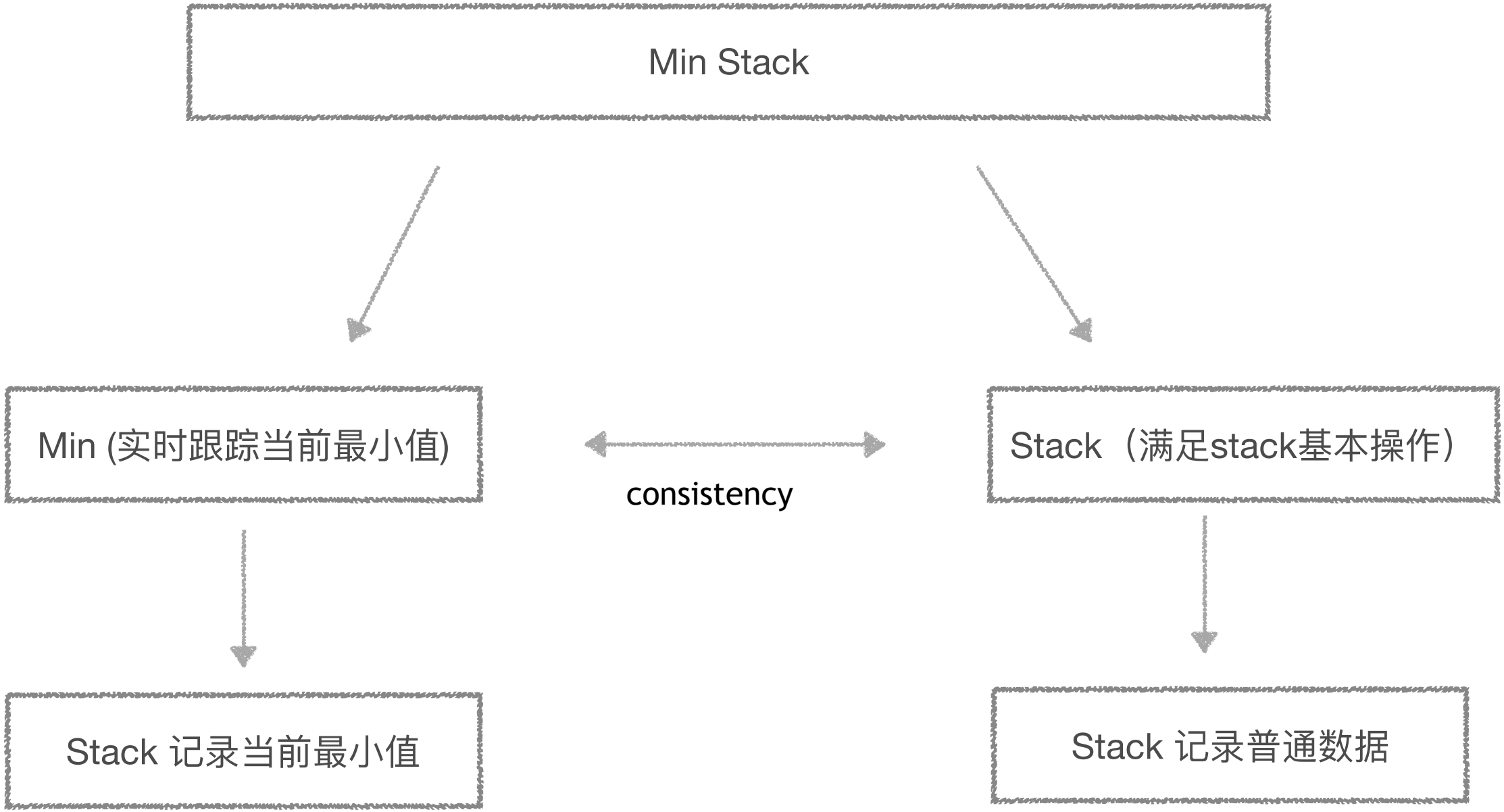
155. Min Stack

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- getMin() -- Retrieve the minimum element in the stack.

Example:

```
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin();   --> Returns -3.
minStack.pop();
minStack.top();      --> Returns 0.
minStack.getMin();   --> Returns -2.
```


Min Stack
Min (实时跟踪当前最小值)
consistency
Stack（满足stack基本操作）
Stack 记录当前最小值
Stack 记录普通数据

```
2  public class MinStack {
3
4      /** initialize your data structure here. */
5      public MinStack() {
6
7      }
8
9      public void push(int x) {
10
11     }
12
13     public void pop() {
14
15     }
16
17     public int top() {
18
19     }
20
21     public int getMin() {
22
23     }
24 }
```

10 min

```java
public class MinStack {
    Deque<Integer> pro;
    Deque<Integer> min;

    /** initialize your data structure here. */
    public MinStack() {
        pro = new ArrayDeque<>();
        min = new ArrayDeque<>();
    }

    public void push(int x) {
        pro.addLast(x);
        if(min.isEmpty()){
            min.addLast(x);
        }else{
            min.addLast(min.peekLast() < x ? min.peekLast() : x);
        }
    }

    public void pop() {
        pro.removeLast();
        min.removeLast();
    }

    public int top() {
        return pro.peekLast();
    }

    public int getMin() {
        return min.peekLast();
    }
}
```

初始化deque

push 操作

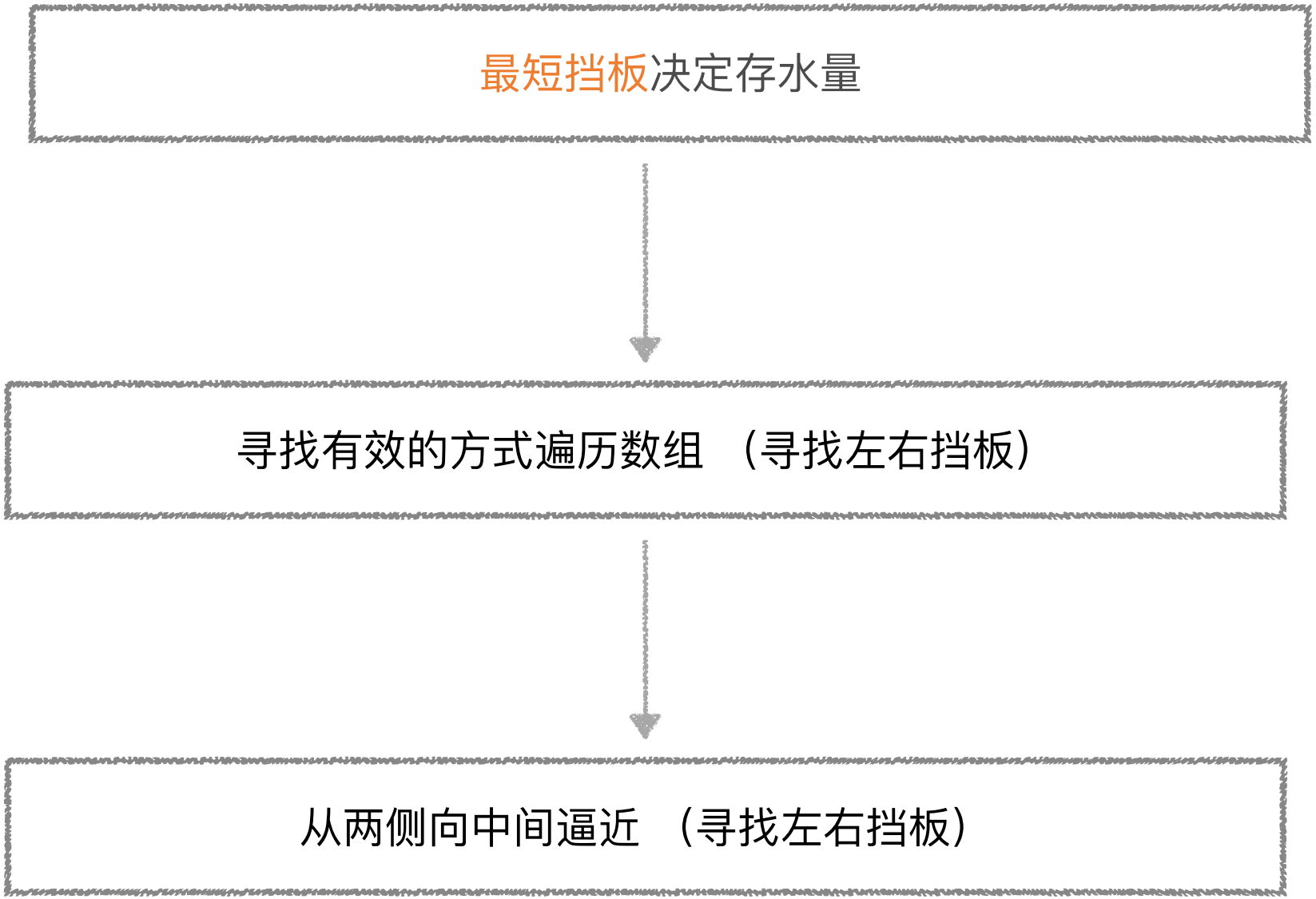
11. Container With Most Water

Given n non-negative integers $a_1, a_2, ..., a_n$, where each represents a point at coordinate (i, a_i). n vertical lines are drawn such that the two endpoints of line i is at (i, a_i) and $(i, 0)$. Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Note: You may not slant the container and n is at least 2.

10 min

```
public int maxArea(int[] height) {
}
```


最短挡板决定存水量
寻找有效的方式遍历数组 （寻找左右挡板）
从两侧向中间逼近 （寻找左右挡板）

```java
public int maxArea(int[] height) {
    if(height == null || height.length == 0){
        return 0;
    }
    int max = 0;
    int beg = 0;
    int end = height.length - 1;
    while(beg < end){
        if(height[beg] < height[end]){
            max = Math.max(max, (end - beg) * height[beg]);
            beg++;
        }else{
            max = Math.max(max, (end - beg) * height[end]);
            end--;
        }
    }
    return max;
}
```

依次向中间逼近

Homework

	225. Implement Stack using Queues	232. Implement Queue using Stacks
21. Merge Two Sorted Lists		
88. Merge Sorted Array		

Q & A

Thank you