

<https://leetcode.com/problems/remove-element/description/>

class Solution {

public int removeElement(int[] nums, int val) {

//easy

int j = 0;

for(int i = 0; i < nums.length; i++){

if(nums[i] != val){

nums[j] = nums[i];

j++;

}

}

return j;

}

}

<https://leetcode.com/problems/remove-duplicates-from-sorted-array/description/>

class Solution {

public:

int removeDuplicates(vector<int>& nums) {

return distance(nums.begin(), unique(nums.begin(), nums.end()));

}

};

[**11. Container With Most Water**](https://leetcode.com/problems/container-with-most-water/)

class Solution {

static {

for(int i=0;i<500;i++){

maxArea(new int[]{0,0});

}

}

public static int maxArea(int[] heights) {

int leftIdx = 0;

int rightIdx = heights.length - 1;

int maxArea = 0;

while (leftIdx < rightIdx) {

int currentMinHeight = heights[leftIdx] <= heights[rightIdx] ? heights[leftIdx] : heights[rightIdx];

int currentArea = currentMinHeight \* (rightIdx - leftIdx);

if (currentArea > maxArea) {

maxArea = currentArea;

}

while (heights[leftIdx] <= currentMinHeight && leftIdx < rightIdx) {

++leftIdx;

}

while (heights[rightIdx] <= currentMinHeight && leftIdx < rightIdx) {

--rightIdx;

}

}

return maxArea;

}

}

[**151. Reverse Words in a String**](https://leetcode.com/problems/reverse-words-in-a-string/)

class Solution {

    public String reverseWords(String s) {

        s = s.replaceAll("\\s+", " ").trim();

        List<String> words = Arrays.asList(s.split(" "));

        Collections.reverse(words);

        return String.join(" ", words);

    }

}

[**54. Spiral Matrix**](https://leetcode.com/problems/spiral-matrix/)

class Solution {

    public List<Integer> spiralOrder(int[][] matrix) {

        // m \* n matrix

        List<Integer> ans = new ArrayList<>();

        int m = matrix.length, n = matrix[0].length;

        int[][] dir = new int[][]{{0, 1}, {1, 0}, {0, -1}, {-1, 0}};

        int currDir = 0;

        // current row and current column

        int currR = 0, currC = 0;

        // loop until collect all elements, and when we reach the end - STOP

        while (ans.size() < m \* n) {

            ans.add(matrix[currR][currC]);

            matrix[currR][currC] = 1001;  // mark visited

            int nextR = currR + dir[currDir][0];

            int nextC = currC + dir[currDir][1];

            // change direction when encounter invalid cases

            if (nextR < 0 || nextR >= m || nextC < 0 || nextC >= n || matrix[nextR][nextC] == 1001)

                currDir = (currDir + 1) % 4;

            currR += dir[currDir][0];

            currC += dir[currDir][1];

        }

        return ans;

    }

}

[**48. Rotate Image**](https://leetcode.com/problems/rotate-image/)

class Solution {

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

        int n = matrix.length;

        for (int i = 0; i < n; i++) {

            for (int j = i; j < n; j++) {

                int temp = matrix[i][j];

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

                matrix[j][i] = temp;

            }

        }

        for (int i = 0; i < n; i++) {

            for (int j = 0; j < n / 2; j++) {

                int temp = matrix[i][j];

                matrix[i][j] = matrix[i][matrix.length - 1 - j];

                matrix[i][matrix.length - 1 - j] = temp;

            }

        }

    }

}