**[Two Sum](https://leetcode.com/problems/two-sum/)**

class Solution {

public int[] twoSum(int[] nums, int target) {

// HashMap

HashMap<Integer, Integer> seen = new HashMap<>();

int[] arr = new int[2];

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

int b = nums[i], a = target - b;

if (seen.containsKey(a)) {

arr[0] = seen.get(a);

arr[1] = i;

return arr;

}

seen.put(b, i);

}

return new int[] {};

}

}

[**167. Two Sum II - Input Array Is Sorted**](https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/)

class Solution {

public int[] twoSum(int[] numbers, int target) {

int l = 0, r = numbers.length - 1;

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

if (numbers[l] + numbers[r] == target) {

return new int[]{l + 1, r + 1};

} else if (numbers[l] + numbers[r] > target) {

--r;

} else {

++l;

}

}

return new int[]{};

}

}

[**88. Merge Sorted Array**](https://leetcode.com/problems/merge-sorted-array/)

class Solution {

public void merge(int[] nums1, int m, int[] nums2, int n) {

int i = m - 1, j = n - 1, k = m + n - 1;

while (i >= 0 && j >= 0) {

if (nums1[i] < nums2[j]) {

nums1[k] = nums2[j];

--k;

--j;

} else {

nums1[k] = nums1[i];

--k;

--i;

}

}

while (j >= 0) {

nums1[k] = nums2[j];

--k;

--j;

}

}

}

[**118. Pascal's Triangle**](https://leetcode.com/problems/pascals-triangle/)

class Solution {

public List<List<Integer>> generate(int numRows) {

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

List<Integer> row, pre = null;

// number of iterations = numRows

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

row = new ArrayList<Integer>();

// from j to i

for (int j = 0; j <= i; ++j)

if (j == 0 || j == i)

row.add(1);

else {

assert pre != null;

row.add(pre.get(j - 1) + pre.get(j));

}

pre = row;

ans.add(row);

}

return ans;

}

}

[**119. Pascal's Triangle II**](https://leetcode.com/problems/pascals-triangle-ii/)

class Solution {

public List<Integer> getRow(int rowIndex) {

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

List<Integer> row, pre = null;

// number of iterations = numRows

for (int i = 0; i < rowIndex + 1; ++i) {

row = new ArrayList<>();

// from j to i

for (int j = 0; j <= i; ++j)

if (j == 0 || j == i)

row.add(1);

else {

// assert pre != null;

row.add(pre.get(j - 1) + pre.get(j));

}

pre = row;

ans.add(row);

}

return ans.get(rowIndex);

}

}

[**121. Best Time to Buy and Sell Stock**](https://leetcode.com/problems/best-time-to-buy-and-sell-stock/)

class Solution {

public int maxProfit(int[] prices) {

// mn = minimum value of prices

int mn = Integer.MAX\_VALUE;

for (int price : prices) {

if (price < mn) {

mn = price;

}

}

int maxProfit = 0, currProfit = 0, currMin = Integer.MAX\_VALUE;

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

if (prices[i] < currMin) {

currMin = prices[i];

}

currProfit = prices[i] - currMin;

if (maxProfit < currProfit) {

maxProfit = currProfit;

}

}

return maxProfit;

}

}

[**122. Best Time to Buy and Sell Stock II**](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/)

class Solution {

public int maxProfit(int[] prices) {

int profit = 0;

for (int i = 1; i < prices.length; i++) {

int diff = prices[i] - prices[i - 1];

if (diff > 0) {

profit += diff;

}

}

return profit;

}

}

[**169. Majority Element**](https://leetcode.com/problems/majority-element/)

class Solution {

public int majorityElement(int[] nums) {

// this is not O(1) space complexity

// Creating a HashMap object with elements of inputArray as keys and their count as values

HashMap<Integer, Integer> elementCountMap = new HashMap<>();

// checking every element of the inputArray

for (int i : nums) {

if (elementCountMap.containsKey(i)) {

// If element is present in elementCountMap, incrementing it's count by 1

elementCountMap.put(i, elementCountMap.get(i) + 1);

} else {

// If element is not present in elementCountMap,

// adding this element to elementCountMap with 1 as it's value

elementCountMap.put(i, 1);

}

}

// Using for-each loop

int key = 0, value = Integer.MIN\_VALUE;

for (Map.Entry<Integer, Integer> mapElement : elementCountMap.entrySet()) {

if (mapElement.getValue() > value) {

value = mapElement.getValue();

key = mapElement.getKey();

}

}

return key;

}

}

[**229. Majority Element II**](https://leetcode.com/problems/majority-element-ii/)

class Solution {

public List<Integer> majorityElement(int[] nums) {

HashMap<Integer, Integer> elementCountMap = new HashMap<>();

// checking every element of the inputArray

for (int i : nums) {

if (elementCountMap.containsKey(i)) {

// If element is present in elementCountMap, incrementing it's count by 1

elementCountMap.put(i, elementCountMap.get(i) + 1);

} else {

// If element is not present in elementCountMap,

// adding this element to elementCountMap with 1 as it's value

elementCountMap.put(i, 1);

}

}

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

// Using for-each loop

for (Map.Entry<Integer, Integer> mapElement : elementCountMap.entrySet()) {

if (mapElement.getValue() > (nums.length / 3)) {

majorityElements.add(mapElement.getKey());

}

}

return majorityElements;

}

}

[**15. 3Sum**](https://leetcode.com/problems/3sum/)

class Solution {

public:

vector<vector<int>> threeSum(vector<int>& nums) {

vector<vector<int>> res;

int n = nums.size();

sort(nums.begin(), nums.end());

for (int i = 0; i < n - 2; i++) {

if(i > 0 && (nums[i] == nums[i-1])) continue;

int j = i + 1;

int k = n - 1;

// 2 pointer approach

while (j < k) {

int curr\_sum = nums[i] + nums[j] + nums[k];

if (curr\_sum == 0) {

res.push\_back({nums[i], nums[j], nums[k]});

while (j + 1 < k && nums[j] == nums[j + 1]) j++;

while (j < k - 1 && nums[k] == nums[k - 1]) k--;

j++;

k--;

} else if (curr\_sum > 0) {

k--;

} else {

j++;

}

}

}

return res;

}

};

[**16. 3Sum Closest**](https://leetcode.com/problems/3sum-closest/)

class Solution {

    public int threeSumClosest(int[] nums, int target) {

        Arrays.sort(nums);

        int closestSum = nums[0] + nums[1] + nums[2];

        for (int i = 0; i < nums.length - 2; i++) {

            int j = i + 1;

            // 2 pointer approach now

            int k = nums.length - 1;

            while (j < k) {

                int sum = nums[i] + nums[j] + nums[k];

                if (Math.abs(target - sum) < Math.abs(target - closestSum)) {

                    closestSum = sum;

                }

                if (sum < target) {

                    ++j;

                } else {

                    --k;

                }

            }

        }

        return closestSum;

    }

}

[**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;

}

}

}

}

[**18. 4Sum**](https://leetcode.com/problems/4sum/)

class Solution {

public List<List<Integer>> fourSum(int[] nums, int target) {

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

if (nums.length < 4)

return ans;

Arrays.sort(nums);

for (int i = 0; i < nums.length - 3; i++) {

if (i > 0 && nums[i] == nums[i - 1])

continue; // prevents duplicate result in ans list

for (int j = i + 1; j < nums.length - 2; j++) {

if (j > i + 1 && nums[j] == nums[j - 1])

continue; // prevents duplicate results in ans list

int low = j + 1;

int high = nums.length - 1;

while (low < high) {

int sum = nums[i] + nums[j] + nums[low] + nums[high];

if (sum == -294967296 || sum == -294967297) {

++low;

--high;

break;

}

if (sum == target) {

ans.add(Arrays.asList(nums[i], nums[j], nums[low], nums[high]));

while (low < high && nums[low] == nums[low + 1])

low++; // skipping over duplicate on low

while (low < high && nums[high] == nums[high - 1])

high--; // skipping over duplicate on high

++low;

--high;

} else if (sum < target) {

++low;

} else {

--high;

}

}

}

}

return ans;

}

}