MISSING NUMBER

var missingNumber = function(nums) {

    nums.sort()

     for (let i = 0; i < nums.length + 1; i++) {

         if (nums.includes(i) == false) {

             return i

         }

     }

 };

var missingNumber = function(nums) {

    let sum = 0, expected = 0;

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

        sum += nums[i];

        expected += i+1;

    }

    return expected - sum;

};

var findDisappearedNumbers = function (nums) {

    const set = new Set(nums) // Создаем множество из чисел в массиве

    const result = [] // Создаем пустой массив для хранения отсутствующих чисел

    // Проходим по числам в диапазоне [1, n]

    for (let i = 1; i <= nums.length; i++) {

        // Если текущее число отсутствует в множестве, добавляем его в массив result

        if (!set.has(i)) {

            result.push(i)

        }

    }

    return result

}

console.log(findDisappearedNumbers([4, 3, 2, 7, 8, 2, 3, 1]))

**Single Number**

Companies

Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

var singleNumber = function (nums) {

    let result = 0

    for (let num of nums) {

        result ^= num

    }

    return result

}

console.log(singleNumber([2, 2, 1, 1, 3, 0, 3]))

[**70. Climbing Stairs**](https://leetcode.com/problems/climbing-stairs/)

var climbStairs = function (n) {

    if (n <= 2) {

        return n

    }

    // Инициализация базовых случаев

    let dp = []

    dp[1] = 1

    dp[2] = 2

    // Заполнение массива dp для всех ступеней от 3 до n

    for (let i = 3; i <= n; i++) {

        dp[i] = dp[i - 1] + dp[i - 2]

    }

    // Количество способов добраться до n-й ступени

    return dp[n]

}

console.log(climbStairs(5))

var climbStairs = function (n) {

    if (n == 0 || n == 1) return 1

    let curr = 1

    let prev = 1

    for (let i = 2; i <= n; i++) {

        let temp = curr

        curr = curr + prev

        prev = temp

        console.log(temp, curr, prev)

    }

    return curr

}

console.log(climbStairs(7))

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

var maxProfit = function (prices) {

    let minPrice = Infinity

    let maxProfit = 0

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

        console.log(prices[i], minPrice)

        if (prices[i] < minPrice) {

            minPrice = prices[i]

        } else if (prices[i] - minPrice > maxProfit) {

            maxProfit = prices[i] - minPrice

            console.log(maxProfit)

        }

    }

    return maxProfit

}

console.log(maxProfit([7, 1, 5, 3, 6, 4, 1, 2, 1, 2])) // 2

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

var twoSum = function (nums, target) {

    let hashTable = {}

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

        let complement = target - nums[i]

        if (hashTable.hasOwnProperty(complement)) {

            return [hashTable[complement], i]

        }

        console.log(hashTable[nums[i]], nums[i], i)

        hashTable[nums[i]] = i

        console.log(hashTable)

    }

    return []

}

console.log(twoSum([2, 3, 11, 15, 7], 9))