**3. 205-Isomorphic Strings**

Given two strings ***s*** and ***t***, determine if they are isomorphic.

Two strings are isomorphic if the characters in ***s*** can be replaced to get ***t***.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character but a character may map to itself.

**Example 1:**

**Input: *s*** = "egg", ***t =*** "add"

**Output:** true

**Example 2:**

**Input: *s*** = "foo", ***t =*** "bar"

**Output:** false

**Example 3:**

**Input: *s*** = "paper", ***t =*** "title"

**Output:** true

**Note:** You may assume both ***s*** and ***t*** have the same length.

**5. 219-Contains Duplicate II**

Given an array of integers and an integer *k*, find out whether there are two distinct indices *i* and *j* in the array such that **nums[i] = nums[j]** and the **absolute** difference between *i* and *j* is at most *k*.

**Example 1:**

**Input:** nums = [1,2,3,1], k = 3

**Output:** true

**Example 2:**

**Input:** nums = [1,0,1,1], k = 1

**Output:** true

**Example 3:**

**Input:** nums = [1,2,3,1,2,3], k = 2

**Output:** false



**19. 350-Intersection of Two Arrays II**

Given two arrays, write a function to compute their intersection.

**Example 1:**

**Input:** nums1 = [1,2,2,1], nums2 = [2,2]

**Output:** [2,2]

**Example 2:**

**Input:** nums1 = [4,9,5], nums2 = [9,4,9,8,4]

**Output:** [4,9]

**Note:**

 Each element in the result should appear as many times as it shows in both arrays.

 The result can be in any order.

**Follow up:**

 What if the given array is already sorted? How would you optimize your algorithm?

 What if *nums1*'s size is small compared to *nums2*'s size? Which algorithm is better?

 What if elements of *nums2* are stored on disk, and the memory is limited such that you cannot load all elements into the memory at once?

**24. 400-Nth Digit**

Find the *n*th digit of the infinite integer sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...

**Note:** *n* is positive and will fit within the range of a 32-bit signed integer (*n* < 231).

**Example 1:**

**Input:**

3

**Output:**

3

**Example 2:**

**Input:**

11

**Output:**

0

**Explanation:**

The 11th digit of the sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ... is a 0, which is part of the number 10.

**28. 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".

**Example 2:**

**Input:**

s: "abab" p: "ab"

**Output:**

[0, 1, 2]

**Explanation:**

The substring with start index = 0 is "ab", which is an anagram of "ab".

The substring with start index = 1 is "ba", which is an anagram of "ab".

The substring with start index = 2 is "ab", which is an anagram of "ab".

**30. 443-String Compression**

Given an array of characters, compress it **in-place**.

The length after compression must always be smaller than or equal to the original array.

Every element of the array should be a **character** (not int) of length 1.

After you are done **modifying the Input array in-place**, return the new length of the array.

**Follow up:** Could you solve it using only O(1) extra space?

**Example 1:**

**Input:**

["a","a","b","b","c","c","c"]

**Output:**

Return 6, and the first 6 characters of the **Input** array should be: ["a","2","b","2","c","3"]

**Explanation:**

"aa" is replaced by "a2". "bb" is replaced by "b2". "ccc" is replaced by "c3".

**Example 2:**

**Input:**

["a"]

**Output:**

Return 1, and the first 1 characters of the **Input** array should be: ["a"]

**Explanation:**

Nothing is replaced.

**Example 3:**

**Input:**

["a","b","b","b","b","b","b","b","b","b","b","b","b"]

**Output:**

Return 4, and the first 4 characters of the **Input** array should be: ["a","b","1","2"].

**Explanation:**

Since the character "a" does not repeat, it is not compressed. "bbbbbbbbbbbb" is replaced by "b12".

Notice each digit has it's own entry in the array.

**Note:**

1. All characters have an ASCII value in [35, 126].

2. 1 <= len(chars) <= 1000.

**31. 448-Find All Numbers Disappeared in an Array**

Given an array of integers where 1 ≤ a[i] ≤ *n* (*n* = size of array), some elements appear twice and others appear once.

Find all the elements of [1, *n*] inclusive that do not appear in this array.

Could you do it without extra space and in O(*n*) runtime? You may assume the returned list does not count as extra space.

**Example:**

**Input:**

[4,3,2,7,8,2,3,1]

**Output:**

[5,6]

**32. 453-Minimum Moves to Equal Array Elements**

Given a **non-empty** integer array of size *n*, find the minimum number of moves required to make all array elements equal, where a move is incrementing *n* - 1 elements by 1.

**Example:**

**Input:**

[1,2,3]

**Output:**

3

**Explanation:**

Only three moves are needed (remember each move increments two elements):

[1,2,3] => [2,3,3] => [3,4,3] => [4,4,4]

**33. 455-Assign Cookies**

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie. Each child i has a greed factor gi, which is the minimum size of a cookie that the child will be content with; and each cookie j has a size sj. If sj >= gi, we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and **Output** the maximum number.

**Note:** You may assume the greed factor is always positive. You cannot assign more than one cookie to one child.

**Example 1:**

**Input:** [1,2,3], [1,1]

**Output:** 1

**Explanation:** You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

You need to **Output** 1.

**Example 2:**

**Input:** [1,2], [1,2,3]

**Output:** 2

**Explanation:** You have 2 children and 3 cookies. The greed factors of 2 children are 1, 2.

You have 3 cookies and their sizes are big enough to gratify all of the children,

You need to **Output** 2.

**34. 459-Repeated Substring Pattern**

Given a non-empty string check if it can be constructed by taking a substring of it and appending multiple copies of the substring together. You may assume the given string consists of lowercase English letters only and its length will not exceed 10000.

**Example 1:**

**Input:** "abab"

**Output:** True

**Explanation:** It's the substring "ab" twice.

**Example 2:**

**Input:** "aba"

**Output:** False

**Example 3:**

**Input:** "abcabcabcabc"

**Output:** True

**Explanation:** It's the substring "abc" four times. (And the substring "abcabc" twice.)

**35. 461-Hamming Distance**

The Hamming distance between two integers is the number of positions at which the corresponding bits are different.

Given two integers x and y, calculate the Hamming distance.

**Note:** 0 ≤ x, y < 231.

**Example:**

**Input:** x = 1, y = 4

**Output:** 2

**Explanation:**

1 (0 0 0 1)

4 (0 1 0 0)

↑ ↑

The above arrows point to positions where the corresponding bits are different.

**36. 475-Heaters**

Winter is coming! Your first job during the contest is to design a standard heater with fixed warm radius to warm all the houses.

Now, you are given positions of houses and heaters on a horizontal line, find out minimum radius of heaters so that all houses could be covered by those heaters.

So, your **Input** will be the positions of houses and heaters seperately, and your expected **Output** will be the minimum radius standard of heaters.

**Note:**

1. Numbers of houses and heaters you are given are non-negative and will not exceed 25000.

2. Positions of houses and heaters you are given are non-negative and will not exceed 10^9.

3. As long as a house is in the heaters' warm radius range, it can be warmed.

4. All the heaters follow your radius standard and the warm radius will the same.

**Example 1:**

**Input:** [1,2,3],[2]

**Output:** 1

**Explanation:** The only heater was placed in the position 2, and if we use the radius 1 standard, then all the houses can be warmed.

**Example 2:**

**Input:** [1,2,3,4],[1,4]

**Output:** 1

**Explanation:** The two heater was placed in the position 1 and 4. We need to use radius 1 standard, then all the houses can be warmed.

**37. 476-Number Complement**

Given a positive integer, **Output** its complement number. The complement strategy is to flip the bits of its binary representation.

**Note:**

1. The given integer is guaranteed to fit within the range of a 32-bit signed integer.

2. You could assume no leading zero bit in the integer’s binary representation.

**Example 1:**

**Input:** 5

**Output:** 2

**Explanation:** The binary representation of 5 is 101 (no leading zero bits), and its complement is 010. So you need to **Output** 2.

**Example 2:**

**Input:** 1

**Output:** 0

**Explanation:** The binary representation of 1 is 1 (no leading zero bits), and its complement is 0. So you need to **Output** 0.

**38. 479-Largest Palindrome Product**

Find the largest palindrome made from the product of two n-digit numbers.

Since the result could be very large, you should return the largest palindrome mod 1337.

**Example:**

**Input**: 2

**Output**: 987

**Explanation**: 99 x 91 = 9009, 9009 % 1337 = 987

**Note:**

The range of n is [1,8].

**39. 482-License Key Formatting**

You are given a license key represented as a string S which consists only alphanumeric character and dashes. The string is separated into N+1 groups by N dashes.

Given a number K, we would want to reformat the strings such that each group contains *exactly* K characters, except for the first group which could be shorter than K, but still must contain at least one character. Furthermore, there must be a dash inserted between two groups and all lowercase letters should be converted to uppercase.

Given a non-empty string S and a number K, format the string according to the rules described above.

**Example 1:**

**Input:** S = "5F3Z-2e-9-w", K = 4

**Output:** "5F3Z-2E9W"

**Explanation:** The string S has been split into two parts, each part has 4 characters.

**Note** that the two extra dashes are not needed and can be removed.

**Example 2:**

**Input:** S = "2-5g-3-J", K = 2

**Output:** "2-5G-3J"

**Explanation:** The string S has been split into three parts, each part has 2 characters except the first part as it could be shorter as mentioned above.

**Note:**

1. The length of string S will not exceed 12,000, and K is a positive integer.

2. String S consists only of alphanumerical characters (a-z and/or A-Z and/or 0-9) and dashes(-).

3. String S is non-empty.

**40. 485-Max Consecutive Ones**

Given a binary array, find the maximum number of consecutive 1s in this array.

**Example 1:**

**Input:** [1,1,0,1,1,1]

**Output:** 3

**Explanation:** The first two digits or the last three digits are consecutive 1s.

The maximum number of consecutive 1s is 3.

**Note:**

 The **Input** array will only contain 0 and 1.

 The length of **Input** array is a positive integer and will not exceed 10,000

**41. 496-Next Greater Element I**

You are given two arrays **(without duplicates)** nums1 and nums2 where nums1’s elements are subset of nums2. Find all the next greater numbers for nums1's elements in the corresponding places of nums2.

The Next Greater Number of a number **x** in nums1 is the first greater number to its right in nums2. If it does not exist, **Output** -1 for this number.

**Example 1:**

**Input: nums1** = [4,1,2], **nums2** = [1,3,4,2].

**Output:** [-1,3,-1]

**Explanation:**

For number 4 in the first array, you cannot find the next greater number for it in the second array, so **Output** -1.

For number 1 in the first array, the next greater number for it in the second array is 3.

For number 2 in the first array, there is no next greater number for it in the second array, so **Output** -1.

**Example 2:**

**Input: nums1** = [2,4], **nums2** = [1,2,3,4].

**Output:** [3,-1]

**Explanation:**

For number 2 in the first array, the next greater number for it in the second array is 3.

For number 4 in the first array, there is no next greater number for it in the second array, so **Output** -1.

**Note:**

1. All elements in nums1 and nums2 are unique.

2. The length of both nums1 and nums2 would not exceed 1000.

**42. 507-Perfect Number**

We define the Perfect Number is a **positive** integer that is equal to the sum of all its **positive** divisors except itself.

Now, given an **integer** n, write a function that returns true when it is a perfect number and false when it is not.

**Example:**

**Input:** 28

**Output:** True

**Explanation:** 28 = 1 + 2 + 4 + 7 + 14

**Note:** The **Input** number **n** will not exceed 100,000,000. (1e8)

**43. 628-Maximum Product of Three Numbers**

Given an integer array, find three numbers whose product is maximum and **Output** the maximum product.

**Example 1:**

**Input:** [1,2,3]

**Output:** 6

**Example 2:**

**Input:** [1,2,3,4]

**Output:** 24

**Note:**

1. The length of the given array will be in range [3,104] and all elements are in the range [-1000, 1000].

2. Multiplication of any three numbers in the **Input** won't exceed the range of 32-bit signed integer.

**44. 633-Sum of Square Numbers**

Given a non-negative integer c, your task is to decide whether there're two integers a and b such that a2 + b2 = c.

**Example 1:**

**Input:** 5

**Output:** True

**Explanation:** 1 \* 1 + 2 \* 2 = 5

**Example 2:**

**Input:** 3

**Output:** False

**45. 672-Bulb Switcher II**

There is a room with n lights which are turned on initially and 4 buttons on the wall. After performing exactly m unknown operations towards buttons, you need to return how many different kinds of status of the n lights could be.

Suppose n lights are labeled as number [1, 2, 3 ..., n], function of these 4 buttons are given below:

1. Flip all the lights.

2. Flip lights with even numbers.

3. Flip lights with odd numbers.

4. Flip lights with (3k + 1) numbers, k = 0, 1, 2, ...

**Example 1:**

**Input:** n = 1, m = 1.

**Output:** 2

**Explanation:** Status can be: [on], [off]

**Example 2:**

**Input:** n = 2, m = 1.

**Output:** 3

**Explanation:** Status can be: [on, off], [off, on], [off, off]

**Example 3:**

**Input:** n = 3, m = 1.

**Output:** 4

**Explanation:** Status can be: [off, on, off], [on, off, on], [off, off, off], [off, on, on].

**Note:** n and m both fit in range [0, 1000].

**46. 728-Self Dividing Numbers**

A *self-dividing number* is a number that is divisible by every digit it contains.

For **Example**, 128 is a self-dividing number because 128 % 1 == 0, 128 % 2 == 0, and 128 % 8 == 0.

Also, a self-dividing number is not allowed to contain the digit zero.

Given a lower and upper number bound, **Output** a list of every possible self dividing number, including the bounds if possible.

**Example 1:**

**Input:**

left = 1, right = 22

**Output:** [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 22]

**Note:**

The boundaries of each **Input** argument are 1 <= left <= right <= 10000.

**47. 754-Reach a Number**

You are standing at position 0 on an infinite number line. There is a goal at position target.

On each move, you can either go left or right. During the *n*-th move (starting from 1), you take *n* steps.

Return the minimum number of steps required to reach the destination.

**Example 1:**

**Input:** target = 3

**Output:** 2

**Explanation:**

On the first move we step from 0 to 1.

On the second step we step from 1 to 3.

**Example 2:**

**Input:** target = 2

**Output:** 3

**Explanation:**

On the first move we step from 0 to 1.

On the second move we step from 1 to -1.

On the third move we step from -1 to 2.

**Note:**

target will be a non-zero integer in the range [-10^9, 10^9].

**48. 877-Stone Game**

Alex and Lee play a game with piles of stones. There are an even number of piles **arranged in a row**, and each pile has a positive integer number of stones piles[i].

The objective of the game is to end with the most stones. The total number of stones is odd, so there are no ties.

Alex and Lee take turns, with Alex starting first. Each turn, a player takes the entire pile of stones from either the beginning or the end of the row. This continues until there are no more piles left, at which point the person with the most stones wins.

Assuming Alex and Lee play optimally, return True if and only if Alex wins the game.

**Example 1:**

**Input:** [5,3,4,5]

**Output:** true

**Explanation:**

Alex starts first, and can only take the first 5 or the last 5.

Say he takes the first 5, so that the row becomes [3, 4, 5].

If Lee takes 3, then the board is [4, 5], and Alex takes 5 to win with 10 points.

If Lee takes the last 5, then the board is [3, 4], and Alex takes 4 to win with 9 points.

This demonstrated that taking the first 5 was a winning move for Alex, so we return true.

**Note:**

1. 2 <= piles.length <= 500

2. piles.length is even.

3. 1 <= piles[i] <= 500

4. sum(piles) is odd.

**49. 914-X of a Kind in a Deck of Cards**

In a deck of cards, each card has an integer written on it.

Return true if and only if you can choose X >= 2 such that it is possible to split the entire deck into 1 or more groups of cards, where:

 Each group has exactly X cards.

 All the cards in each group have the same integer.

**Example 1:**

**Input:** [1,2,3,4,4,3,2,1]

**Output:** true

**Explanation**: Possible partition [1,1],[2,2],[3,3],[4,4]

**Example 2:**

**Input:** [1,1,1,2,2,2,3,3]

**Output:** false

**Explanation**: No possible partition.

**Example 3:**

**Input:** [1]

**Output:** false

**Explanation**: No possible partition.

**Example 4:**

**Input:** [1,1]

**Output:** true

**Explanation**: Possible partition [1,1]

**Example 5:**

**Input:** [1,1,2,2,2,2]

**Output:** true

**Explanation**: Possible partition [1,1],[2,2],[2,2]

**Note**:

1. 1 <= deck.length <= 10000

2. 0 <= deck[i] < 10000

**50. 942-DI String Match**

Given a string S that **only** contains "I" (increase) or "D" (decrease), let N = S.length.

Return **any** permutation A of [0, 1, ..., N] such that for all i = 0, ..., N-1:

 If S[i] == "I", then A[i] < A[i+1]

 If S[i] == "D", then A[i] > A[i+1]

**Example 1:**

**Input:** "IDID"

**Output:** [0,4,1,3,2]

**Example 2:**

**Input:** "III"

**Output:** [0,1,2,3]

**Example 3:**

**Input:** "DDI"

**Output:** [3,2,0,1]

**Note:**

1. 1 <= S.length <= 10000

2. S only contains characters "I" or "D".

**51. 7-Reverse Integer**

**Given a 32-bit signed integer, reverse digits of an integer.**

**Example 1:**

**Input: 123**

**Output: 321**

**Example 2:**

**Input: -123**

**Output: -321**

**Example 3:**

**Input: 120**

**Output: 21**

**Note:**

**Assume we are dealing with an environment which could only store integers within the 32-bit signed integer range: [−2^31, 2^31 − 1]. For the purpose of this problem, assume that your function returns 0 when the reversed integer overflows.**

**52. 9-Palindrome Number**

Determine whether an integer is a palindrome. An integer is a palindrome when it reads the same backward as forward.

**Example 1:**

**Input: 121**

**Output:** true

**Example 2:**

**Input: -121**

**Output:** false

**Explanation:** From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.

**Example 3:**

**Input: 10**

**Output:** false

**Explanation:** Reads 01 from right to left. Therefore it is not a palindrome.

**Follow up:**

Coud you solve it without converting the integer to a string?

**53. 11-Container With Most Water**

Given n non-negative integers *a1*, *a2*, ..., *an* , where each represents a point at coordinate (*i*, *ai*). *n* vertical lines are drawn such that the two endpoints of line *i* is at (*i*, *ai*) 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.

The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

**Example:**

**Input:** [1,8,6,2,5,4,8,3,7]

**Output:** 49

**54. 12-Integer to Roman**

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

**Symbol Value**

**I 1**

**V 5**

**X 10**

**L 50**

**C 100**

**D 500**

**M 1000**

For **Example**, two is written as II in Roman numeral, just two one's added together. Twelve is written as, XII, which is simply X + II. The number twenty seven is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

•I can be placed before V (5) and X (10) to make 4 and 9.

•X can be placed before L (50) and C (100) to make 40 and 90.

•C can be placed before D (500) and M (1000) to make 400 and 900.

Given an integer, convert it to a roman numeral. **Input** is guaranteed to be within the range from 1 to 3999.

**Example 1:**

**Input: 3**

**Output:** "III"

**Example 2:**

**Input: 4**

**Output:** "IV"

**Example 3:**

**Input: 9**

**Output:** "IX"

**Example 4:**

**Input: 58**

**Output:** "LVIII"

**Explanation:** L = 50, V = 5, III = 3.

**Example 5:**

**Input: 1994**

**Output:** "MCMXCIV"

**Explanation:** M = 1000, CM = 900, XC = 90 and IV = 4.

**55. 14-Longest Common Prefix**

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

**Example 1:**

**Input:** ["flower","flow","flight"]

**Output:** "fl"

**Example 2:**

**Input:** ["dog","racecar","car"]

**Output: ""**

**Explanation:** There is no common prefix among the **Input** strings.

**Note:**

All given **Input**s are in lowercase letters a-z.

**56. 26-Remove Duplicates from Sorted Array**

Given a sorted array nums, remove the duplicates in-place such that each element appear only once and return the new length.

Do not allocate extra space for another array, you must do this by modifying the **Input** array in-place with O(1) extra memory.

**Example 1:**

Given nums = [1,1,2],

Your function should return length = 2, with the first two elements of nums being 1 and 2 respectively.

It doesn't matter what you leave beyond the returned length.

**Example 2:**

Given nums = [0,0,1,1,1,2,2,3,3,4],

Your function should return length = 5, with the first five elements of nums being modified to 0, 1, 2, 3, and 4 respectively.

It doesn't matter what values are set beyond the returned length.

**57. 27-Remove Element**

Given an array nums and a value val, remove all instances of that value in-place and return the new length. Do not allocate extra space for another array, you must do this by modifying the **Input** array in-place with O(1) extra memory. The order of elements can be changed. It doesn't matter what you leave beyond the new length.

**Example 1:**

Given nums = [3,2,2,3], val = 3,

Your function should return length = 2, with the first two elements of nums being 2.

It doesn't matter what you leave beyond the returned length.

**Example 2:**

Given nums = [0,1,2,2,3,0,4,2], val = 2,

Your function should return length = 5, with the first five elements of nums containing 0, 1, 3, 0, and 4.

**Note** that the order of those five elements can be arbitrary.

It doesn't matter what values are set beyond the returned length.

**58. 28-Implement strStr()**

Implement strStr().

Return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

**Example 1:**

**Input:** haystack = "hello", needle = "ll"

**Output: 2**

**Example 2:**

**Input:** haystack = "aaaaa", needle = "bba"

**Output: -1**

**59. 34-Find First and Last Position of Element in Sorted Array**

Given an array of integers nums sorted in ascending order, find the starting and ending position of a given target value. Your algorithm's runtime complexity must be in the order of O(log n). If the target is not found in the array, return [-1, -1].

**Example 1:**

**Input:** nums = [5,7,7,8,8,10], target = 8

**Output:** [3,4]

**Example 2:**

**Input:** nums = [5,7,7,8,8,10], target = 6

**Output: [-1,-1]**

**60. 35-Search Insert Position**

Given a sorted array and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You may assume no duplicates in the array.

**Example 1:**

**Input:** [1,3,5,6], 5

**Output: 2**

**Example 2:**

**Input:** [1,3,5,6], 2

**Output: 1**

**Example 3:**

**Input:** [1,3,5,6], 7

**Output: 4**

**Example 4:**

**Input:** [1,3,5,6], 0

**Output: 0**

**61. 50-Pow(x, n)**

Implement pow(x, n), which calculates x raised to the power n (xn).

**Example 1:**

**Input:** 2.00000, 10

**Output:** 1024.00000

**Example 2:**

**Input:** 2.10000, 3

**Output:** 9.26100

**Example 3:**

**Input:** 2.00000, -2

**Output:** 0.25000

**Explanation:** 2-2 = 1/22 = 1/4 = 0.25

**Note:**

•-100.0 < x < 100.0

•n is a 32-bit signed integer, within the range [−231, 231 − 1]

**62. 58-Length of Last Word**

Given a string s consists of upper/lower-case alphabets and empty space characters ' ', return the length of last word in the string.

If the last word does not exist, return 0.

**Note:** A word is defined as a character sequence consists of non-space characters only.

**Example:**

**Input:** "Hello World"

**Output: 5**

**63. 63-Unique Paths II**

A robot is located at the top-left corner of a *m* x *n* grid (marked 'Start' in the diagram below).

The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below).

Now consider if some obstacles are added to the grids. How many unique paths would there be?

An obstacle and empty space is marked as 1 and 0 respectively in the grid.

**Note:** *m* and *n* will be at most 100.

**Example 1:**

**Input:**

[

[0,0,0],

[0,1,0],

[0,0,0]

]

**Output:** 2

**Explanation:**

There is one obstacle in the middle of the 3x3 grid above.

There are two ways to reach the bottom-right corner:

1. Right -> Right -> Down -> Down

2. Down -> Down -> Right -> Right

**64. 64-Minimum Path Sum**

Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right which minimizes the sum of all numbers along its path.

**Note:** You can only move either down or right at any point in time.

**Example:**

**Input:**

[

[1,3,1],

[1,5,1],

[4,2,1]

]

**Output: 7**

**Explanation:** Because the path 1→3→1→1→1 minimizes the sum.

**65. 66-Plus One**

Given a non-empty array of digits representing a non-negative integer, plus one to the integer.

The digits are stored such that the most significant digit is at the head of the list, and each element in the array contain a single digit.

You may assume the integer does not contain any leading zero, except the number 0 itself.

**Example 1:**

**Input:** [1,2,3]

**Output:** [1,2,4]

**Explanation:** The array represents the integer 123.

**Example 2:**

**Input:** [4,3,2,1]

**Output:** [4,3,2,2]

**Explanation:** The array represents the integer 4321.

**66. 67-Add Binary**

Given two binary strings, return their sum (also a binary string).

The **Input** strings are both non-empty and contains only characters 1 or 0.

**Example 1:**

**Input:** a = "11", b = "1"

**Output:** "100"

**Example 2:**

**Input:** a = "1010", b = "1011"

**Output:** "10101"

**67. 69-Sqrt(x)**

Implement int sqrt(int x).

Compute and return the square root of x, where x is guaranteed to be a non-negative integer.

Since the return type is an integer, the decimal digits are truncated and only the integer part of the result is returned.

**Example 1:**

**Input: 4**

**Output: 2**

**Example 2:**

**Input: 8**

**Output: 2**

**Explanation:** The square root of 8 is 2.82842..., and since the decimal part is truncated, 2 is returned.

**68. 70-Climbing Stairs**

You are climbing a stair case. It takes n steps to reach to the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

**Note:** Given n will be a positive integer.

**Example 1:**

**Input: 2**

**Output: 2**

**Explanation:** There are two ways to climb to the top.

1. 1 step + 1 step

2. 2 steps

**Example 2:**

**Input: 3**

**Output: 3**

**Explanation:** There are three ways to climb to the top.

1. 1 step + 1 step + 1 step

2. 1 step + 2 steps

3. 2 steps + 1 step

**69. 88-Merge Sorted Array**

Given two sorted integer arrays nums1 and nums2, merge nums2 into nums1 as one sorted array.

**Note:**

**•The number of elements initialized in nums1 and nums2are m and n respectively.**

**•You may assume that nums1 has enough space (size that is greater or equal to m + n) to hold additional elements from nums2.**

**Example:**

**Input:**

**nums1 = [1,2,3,0,0,0], m = 3**

**nums2 = [2,5,6], n = 3**

**Output: [1,2,2,3,5,6]**

**70. 121-Best Time to Buy and Sell Stock**

Say you have an array for which the ith element is the price of a given stock on day i.

If you were only permitted to complete at most one transaction (i.e., buy one and sell one share of the stock), design an algorithm to find the maximum profit.

**Note** that you cannot sell a stock before you buy one.

**Example 1:**

**Input:** [7,1,5,3,6,4]

**Output: 5**

**Explanation:** Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Not 7-1 = 6, as selling price needs to be larger than buying price.

**Example 2:**

**Input:** [7,6,4,3,1]

**Output: 0**

**Explanation:** In this case, no transaction is done, i.e. max profit = 0.

**71. 125-Valid Palindrome**

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

**Note:** For the purpose of this problem, we define empty string as valid palindrome.

**Example 1:**

**Input:** "A man, a plan, a canal: Panama"

**Output:** true

**Example 2:**

**Input:** "race a car"

**Output:** false

**72. 136-Single Number**

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

**Note:**

Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

**Example 1:**

**Input:** [2,2,1]

**Output: 1**

**Example 2:**

**Input:** [4,1,2,1,2]

**Output: 4**

**73. 137-Single Number II**

Given a non-empty array of integers, every element appears three times except for one, which appears exactly once. Find that single one.

**Note:**

Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

**Example 1:**

**Input:** [2,2,3,2]

**Output: 3**

**Example 2:**

**Input:** [0,1,0,1,0,1,99]

**Output: 99**

**74. 167-Two Sum II - Input array is sorted**

Given an array of integers that is already sorted in ascending order, find two numbers such that they add up to a specific target number.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2.

**Note:**

•Your returned answers (both index1 and index2) are not zero-based.

•You may assume that each **Input** would have exactly one solution and you may not use the same element twice.

**Example:**

**Input:** numbers = [2,7,11,15], target = 9

**Output:** [1,2]

**Explanation:** The sum of 2 and 7 is 9. Therefore index1 = 1, index2 = 2.

**75. 169-Majority Element**

Given an array of size n, find the majority element. The majority element is the element that appears more than ⌊ n/2 ⌋ times.

You may assume that the array is non-empty and the majority element always exist in the array.

**Example 1:**

**Input:** [3,2,3]

**Output: 3**

**Example 2:**

**Input:** [2,2,1,1,1,2,2]

**Output: 2**

**76. 172-Factorial Trailing Zeroes**

Given an integer n, return the number of trailing zeroes in n!.

**Example 1:**

**Input: 3**

**Output: 0**

**Explanation:** 3! = 6, no trailing zero.

**Example 2:**

**Input: 5**

**Output: 1**

**Explanation:** 5! = 120, one trailing zero.

**77. 189-Rotate Array**

Given an array, rotate the array to the right by k steps, where k is non-negative.

**Example 1:**

**Input:** [1,2,3,4,5,6,7] and k = 3

**Output:** [5,6,7,1,2,3,4]

**Explanation:**

rotate 1 steps to the right: [7,1,2,3,4,5,6]

rotate 2 steps to the right: [6,7,1,2,3,4,5]

rotate 3 steps to the right: [5,6,7,1,2,3,4]

**Example 2:**

**Input:** [-1,-100,3,99] and k = 2

**Output:** [3,99,-1,-100]

**Explanation:**

rotate 1 steps to the right: [99,-1,-100,3]

rotate 2 steps to the right: [3,99,-1,-100]

**Note:**

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

•Could you do it in-place with O(1) extra space?

**78. 190-Reverse Bits**

Reverse bits of a given 32 bits unsigned integer.

**Example:**

**Input:** 43261596

**Output:** 964176192

**Explanation:** 43261596 represented in binary as 00000010100101000001111010011100,

return 964176192 represented in binary as 00111001011110000010100101000000.

**79. 191-Number of 1 Bits**

Write a function that takes an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight).

**Example 1:**

**Input: 11**

**Output: 3**

**Explanation:** Integer 11 has binary representation 00000000000000000000000000001011

**Example 2:**

**Input: 128**

**Output: 1**

**Explanation:** Integer 128 has binary representation 00000000000000000000000010000000

**80. 746-Min Cost Climbing Stairs**

On a staircase, the i-th step has some non-negative cost cost[i] assigned (0 indexed).

Once you pay the cost, you can either climb one or two steps. You need to find minimum cost to reach the top of the floor, and you can either start from the step with index 0, or the step with index 1.

**Example 1:**

**Input:** cost = [10, 15, 20]

**Output: 15**

**Explanation:** Cheapest is start on cost[1], pay that cost and go to the top.

**Example 2:**

**Input:** cost = [1, 100, 1, 1, 1, 100, 1, 1, 100, 1]

**Output: 6**

**Explanation:** Cheapest is start on cost[0], and only step on 1s, skipping cost[3].

**Note:**

1. cost will have a length in the range [2, 1000].

2. Every cost[i] will be an integer in the range [0, 999].

**81. 867-Transpose Matrix**

Given a matrix A, return the transpose of A.

The transpose of a matrix is the matrix flipped over it's main diagonal, switching the row and column indices of the matrix.

**Example 1:**

**Input:** [[1,2,3],[4,5,6],[7,8,9]]

**Output:** [[1,4,7],[2,5,8],[3,6,9]]

**Example 2:**

**Input:** [[1,2,3],[4,5,6]]

**Output:** [[1,4],[2,5],[3,6]]

**Note:**

1. 1 <= A.length <= 1000

2. 1 <= A[0].length <= 1000

**82. 868-Binary Gap**

Given a positive integer N, find and return the longest distance between two consecutive 1's in the binary representation of N.

If there aren't two consecutive 1's, return 0.

**Example 1:**

**Input: 22**

**Output:** 2

**Explanation:**

22 in binary is 0b10110.

In the binary representation of 22, there are three ones, and two consecutive pairs of 1's.

The first consecutive pair of 1's have distance 2.

The second consecutive pair of 1's have distance 1.

The answer is the largest of these two distances, which is 2.

**Example 2:**

**Input: 5**

**Output:** 2

**Explanation:**

5 in binary is 0b101.

**Example 3:**

**Input: 6**

**Output:** 1

**Explanation:**

6 in binary is 0b110.

**Example 4:**

**Input: 8**

**Output:** 0

**Explanation:**

8 in binary is 0b1000.

There aren't any consecutive pairs of 1's in the binary representation of 8, so we return 0.

**Note:**

• 1 <= N <= 10^9

**83. 884-Uncommon Words from Two Sentences**

We are given two sentences A and B. (A *sentence* is a string of space separated words. Each *word* consists only of lowercase letters.)

A word is *uncommon* if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Return a list of all uncommon words.

You may return the list in any order.

**Example 1:**

**Input:** A = "this apple is sweet", B = "this apple is sour"

**Output:** ["sweet","sour"]

**Example 2:**

**Input:** A = "apple apple", B = "banana"

**Output:** ["banana"]

**Note:**

1. 0 <= A.length <= 200

2. 0 <= B.length <= 200

3. A and B both contain only spaces and lowercase letters.

**84. 888. Fair Candy Swap**

Alice and Bob have candy bars of different sizes: A[i] is the size of the i-th bar of candy that Alice has, and B[j] is the size of the j-th bar of candy that Bob has.

Since they are friends, they would like to exchange one candy bar each so that after the exchange, they both have the same total amount of candy. (*The total amount of candy a person has is the sum of the sizes of candy bars they have.*)

Return an integer array ans where ans[0] is the size of the candy bar that Alice must exchange, and ans[1] is the size of the candy bar that Bob must exchange.

If there are multiple answers, you may return any one of them. It is guaranteed an answer exists.

**Example 1:**

**Input:** A = [1,1], B = [2,2]

**Output:** [1,2]

**Example 2:**

**Input:** A = [1,2], B = [2,3]

**Output:** [1,2]

**Example 3:**

**Input:** A = [2], B = [1,3]

**Output:** [2,3]

**Example 4:**

**Input:** A = [1,2,5], B = [2,4]

**Output:** [5,4]

**Note:**

• 1 <= A.length <= 10000

• 1 <= B.length <= 10000

• 1 <= A[i] <= 100000

• 1 <= B[i] <= 100000

• It is guaranteed that Alice and Bob have different total amounts of candy.

• It is guaranteed there exists an answer.

**85. 893-Groups of Special-Equivalent Strings**

You are given an array A of strings.

Two strings S and T are *special-equivalent* if after any number of *moves*, S == T.

A *move* consists of choosing two indices i and j with i % 2 == j % 2, and swapping S[i] with S[j].

Now, a *group of special-equivalent strings from A* is a non-empty subset S of A such that any string not in S is not special-equivalent with any string in S.

Return the number of groups of special-equivalent strings from A.

**Example 1:**

**Input:** ["a","b","c","a","c","c"]

**Output:** 3

**Explanation**: 3 groups ["a","a"], ["b"], ["c","c","c"]

**Example 2:**

**Input:** ["aa","bb","ab","ba"]

**Output:** 4

**Explanation**: 4 groups ["aa"], ["bb"], ["ab"], ["ba"]

**Example 3:**

**Input:** ["abc","acb","bac","bca","cab","cba"]

**Output:** 3

**Explanation**: 3 groups ["abc","cba"], ["acb","bca"], ["bac","cab"]

**Example 4:**

**Input:** ["abcd","cdab","adcb","cbad"]

**Output:** 1

**Explanation**: 1 group ["abcd","cdab","adcb","cbad"]

**Note:**

• 1 <= A.length <= 1000

• 1 <= A[i].length <= 20

• All A[i] have the same length.

• All A[i] consist of only lowercase letters.

**86. 896-Monotonic Array**

An array is *monotonic* if it is either monotone increasing or monotone decreasing.

An array A is monotone increasing if for all i <= j, A[i] <= A[j]. An array A is monotone decreasing if for all i <= j, A[i] >= A[j].

Return true if and only if the given array A is monotonic.

**Example 1:**

**Input:** [1,2,2,3]

**Output:** true

**Example 2:**

**Input:** [6,5,4,4]

**Output:** true

**Example 3:**

**Input:** [1,3,2]

**Output:** false

**Example 4:**

**Input:** [1,2,4,5]

**Output:** true

**Example 5:**

**Input:** [1,1,1]

**Output:** true

**Note:**

1. 1 <= A.length <= 50000

2. -100000 <= A[i] <= 100000

**87. 905-Sort Array By Parity**

Given an array A of non-negative integers, return an array consisting of all the even elements of A, followed by all the odd elements of A.

You may return any answer array that satisfies this condition.

**Example 1:**

**Input:** [3,1,2,4]

**Output:** [2,4,3,1]

The **Output**s [4,2,3,1], [2,4,1,3], and [4,2,1,3] would also be accepted.

**Note:**

1. 1 <= A.length <= 5000

2. 0 <= A[i] <= 5000

**88. 908-Smallest Range I**

Given an array A of integers, for each integer A[i] we may choose any x with -K <= x <= K, and add x to A[i].

After this process, we have some array B.

Return the smallest possible difference between the maximum value of B and the minimum value of B.

**Example 1:**

**Input:** A = [1], K = 0

**Output:** 0

**Explanation**: B = [1]

**Example 2:**

**Input:** A = [0,10], K = 2

**Output:** 6

**Explanation**: B = [2,8]

**Example 3:**

**Input:** A = [1,3,6], K = 3

**Output:** 0

**Explanation**: B = [3,3,3] or B = [4,4,4]

**Note:**

1. 1 <= A.length <= 10000

2. 0 <= A[i] <= 10000

3. 0 <= K <= 10000

**89. 914-X of a Kind in a Deck of Cards**

In a deck of cards, each card has an integer written on it.

Return true if and only if you can choose X >= 2 such that it is possible to split the entire deck into 1 or more groups of cards, where:

• Each group has exactly X cards.

• All the cards in each group have the same integer.

**Example 1:**

**Input:** [1,2,3,4,4,3,2,1]

**Output:** true

**Explanation**: Possible partition [1,1],[2,2],[3,3],[4,4]

**Example 2:**

**Input:** [1,1,1,2,2,2,3,3]

**Output:** false

**Explanation**: No possible partition.

**Example 3:**

**Input:** [1]

**Output:** false

**Explanation**: No possible partition.

**Example 4:**

**Input:** [1,1]

**Output:** true

**Explanation**: Possible partition [1,1]

**Example 5:**

**Input:** [1,1,2,2,2,2]

**Output:** true

**Explanation**: Possible partition [1,1],[2,2],[2,2] **Note:**

1. 1 <= deck.length <= 10000

2. 0 <= deck[i] < 10000

**90. 917-Reverse Only Letters**

Given a string S, return the "reversed" string where all characters that are not a letter stay in the same place, and all letters reverse their positions.

**Example 1:**

**Input:** "ab-cd"

**Output:** "dc-ba"

**Example 2:**

**Input:** "a-bC-dEf-ghIj"

**Output:** "j-Ih-gfE-dCba"

**Example 3:**

**Input:** "Test1ng-Leet=code-Q!"

**Output:** "Qedo1ct-eeLg=ntse-T!"

**Note:**

1. S.length <= 100

2. 33 <= S[i].ASCIIcode <= 122

3. S doesn't contain \ or "

4.

**91. 921-Minimum Add to Make Parentheses Valid**

Given a string S of '(' and ')' parentheses, we add the minimum number of parentheses ( '(' or ')', and in any positions ) so that the resulting parentheses string is valid.

Formally, a parentheses string is valid if and only if:

• It is the empty string, or

• It can be written as AB (A concatenated with B), where A and B are valid strings, or

• It can be written as (A), where A is a valid string.

Given a parentheses string, return the minimum number of parentheses we must add to make the resulting string valid.

**Example** 1:

**Input:** "())"

**Output:** 1

**Example 2:**

**Input:** "((("

**Output:** 3

**Example 3:**

**Input:** "()"

**Output:** 0

**Example 4:**

**Input:** "()))(("

**Output:** 4

**Note:**

1. S.length <= 1000

2. S only consists of '(' and ')' characters.

**92. 922-Sort Array By Parity II**

Given an array A of non-negative integers, half of the integers in A are odd, and half of the integers are even.

Sort the array so that whenever A[i] is odd, i is odd; and whenever A[i] is even, i is even.

You may return any answer array that satisfies this condition.

**Example 1:**

**Input:** [4,2,5,7]

**Output:** [4,5,2,7]

**Explanation:** [4,7,2,5], [2,5,4,7], [2,7,4,5] would also have been accepted.

**Note:**

1. 2 <= A.length <= 20000

2. A.length % 2 == 0

3. 0 <= A[i] <= 1000

**93. 925-Long Pressed Name**

Your friend is typing his name into a keyboard. Sometimes, when typing a character c, the key might get *long pressed*, and the character will be typed 1 or more times.

You examine the typed characters of the keyboard. Return True if it is possible that it was your friends name, with some characters (possibly none) being long pressed.

**Example** 1:

**Input:** name = "alex", typed = "aaleex"

**Output:** true

**Explanation:** 'a' and 'e' in 'alex' were long pressed.

**Example 2:**

**Input:** name = "saeed", typed = "ssaaedd"

**Output:** false

**Explanation:** 'e' must have been pressed twice, but it wasn't in the typed **Output**.

**Example 3:**

**Input:** name = "leelee", typed = "lleeelee"

**Output:** true

**Example 4:**

**Input:** name = "laiden", typed = "laiden"

**Output:** true

**Explanation:** It's not necessary to long press any character.

**Note:**

1. name.length <= 1000

2. typed.length <= 1000

3. The characters of name and typed are lowercase letters.

**94. 932-Beautiful Array**

For some fixed N, an array A is *beautiful* if it is a permutation of the integers 1, 2, ..., N, such that:

For every i < j, there is **no** k with i < k < j such that A[k] \* 2 = A[i] + A[j].

Given N, return **any** beautiful array A. (It is guaranteed that one exists.)

**Example** 1:

**Input: 4**

**Output:** [2,1,4,3]

**Example 2:**

**Input: 5**

**Output:** [3,1,2,5,4]

**Note:**

• 1 <= N <= 1000

**95. 933-Number of Recent Calls**

Write a class RecentCounter to count recent requests.

It has only one method: ping(int t), where t represents some time in milliseconds.

Return the number of pings that have been made from 3000 milliseconds ago until now.

Any ping with time in [t - 3000, t] will count, including the current ping.

It is guaranteed that every call to ping uses a strictly larger value of t than before.

**Example 1:**

**Input: Input**s = ["RecentCounter","ping","ping","ping","ping"], **Input**s = [[],[1],[100],[3001],[3002]]

**Output:** [null,1,2,3,3]

**Note:**

1. Each test case will have at most 10000 calls to ping.

2. Each test case will call ping with strictly increasing values of t.

3. Each call to ping will have 1 <= t <= 10^9.

**96. 937-Reorder Log Files**

You have an array of logs. Each log is a space delimited string of words.

For each log, the first word in each log is an alphanumeric *identifier*. Then, either:

• Each word after the identifier will consist only of lowercase letters, or;

• Each word after the identifier will consist only of digits.

We will call these two varieties of logs *letter-logs* and *digit-logs*. It is guaranteed that each log has at least one word after its identifier.

Reorder the logs so that all of the letter-logs come before any digit-log. The letter-logs are ordered lexicographically ignoring identifier, with the identifier used in case of ties. The digit-logs should be put in their original order.

Return the final order of the logs.

**Example 1:**

**Input:** ["a1 9 2 3 1","g1 act car","zo4 4 7","ab1 off key dog","a8 act zoo"]

**Output:** ["g1 act car","a8 act zoo","ab1 off key dog","a1 9 2 3 1","zo4 4 7"]

**97. 939-Minimum Area Rectangle**

Given a set of points in the xy-plane, determine the minimum area of a rectangle formed from these points, with sides parallel to the x and y axes.

If there isn't any rectangle, return 0.

**Example 1:**

**Input:** [[1,1],[1,3],[3,1],[3,3],[2,2]]

**Output:** 4

**Example 2:**

**Input:** [[1,1],[1,3],[3,1],[3,3],[4,1],[4,3]]

**Output:** 2

**98. 941-Valid Mountain Array**

Given an array A of integers, return true if and only if it is a *valid mountain array*.

Recall that A is a mountain array if and only if:

• A.length >= 3

• There exists some i with 0 < i < A.length - 1 such that:

• A[0] < A[1] < ... A[i-1] < A[i]

• A[i] > A[i+1] > ... > A[B.length - 1]

**Example 1:**

**Input:** [2,1]

**Output:** false

**Example 2:**

**Input:** [3,5,5]

**Output:** false

**Example 3:**

**Input:** [0,3,2,1]

**Output:** true

**Note:**

1. 0 <= A.length <= 10000

2. 0 <= A[i] <= 10000

**99. 942-DI String Match**

Given a string S that **only** contains "I" (increase) or "D" (decrease), let N = S.length.

Return **any** permutation A of [0, 1, ..., N] such that for all i = 0, ..., N-1:

• If S[i] == "I", then A[i] < A[i+1]

• If S[i] == "D", then A[i] > A[i+1]

**Example 1:**

**Input:** "IDID"

**Output:** [0,4,1,3,2]

**Example 2:**

**Input:** "III"

**Output:** [0,1,2,3]

**Example 3:**

**Input:** "DDI"

**Output:** [3,2,0,1]

**100. 944-Delete Columns to Make Sorted**

We are given an array A of N lowercase letter strings, all of the same length.

Now, we may choose any set of deletion indices, and for each string, we delete all the characters in those indices.

For **Example**, if we have a string "abcdef" and deletion indices {0, 2, 3}, then the final string after deletion is "bef".

Suppose we chose a set of deletion indices D such that after deletions, each remaining column in A is in **non-decreasing** sorted order.

Formally, the c-th column is [A[0][c], A[1][c], ..., A[A.length-1][c]]

Return the minimum possible value of D.length.

**Example 1:**

**Input:** ["cba","daf","ghi"]

**Output:** 1

**Example 2:**

**Input:** ["a","b"]

**Output:** 0

**Example 3:**

**Input:** ["zyx","wvu","tsr"]

**Output:** 3

**101. (303) Range Sum Query - Immutable**

Given an integer array nums, find the sum of the elements between indices i and j (i ≤ j), inclusive.

Implement the NumArray class:

NumArray(int[] nums) Initializes the object with the integer array nums.

int sumRange(int i, int j) Return the sum of the elements of the nums array in the range [i, j] inclusive (i.e., sum(nums[i], nums[i + 1], ... , nums[j]))

**Example** 1:

**Input**

["NumArray", "sumRange", "sumRange", "sumRange"]

[[[-2, 0, 3, -5, 2, -1]], [0, 2], [2, 5], [0, 5]]

**Output**

[null, 1, -1, -3]

**Explanation**

NumArray numArray = new NumArray([-2, 0, 3, -5, 2, -1]);

numArray.sumRange(0, 2); // return 1 ((-2) + 0 + 3)

numArray.sumRange(2, 5); // return -1 (3 + (-5) + 2 + (-1))

numArray.sumRange(0, 5); // return -3 ((-2) + 0 + 3 + (-5) + 2 + (-1))

**Constrains**:

0 <= nums.length <= 104

-105 <= nums[i] <= 105

0 <= i <= j < nums.length

At most 104 calls will be made to sumRange.

**102. (307) Range Sum Query - Mutable**

Given an integer array nums, find the sum of the elements between indices i and j (i ≤ j), inclusive.

The update(i, val) function modifies nums by updating the element at index i to val.

**Example**:

Given nums = [1, 3, 5]

sumRange(0, 2) -> 9

update(1, 2)

sumRange(0, 2) -> 8

**Constrains**:

The array is only modifiable by the update function.

You may assume the number of calls to update and sumRange function is distributed evenly.

0 <= i <= j <= nums.length - 1

**103. (316) Remove Duplicate Letters**

Given a string s, remove duplicate letters so that every letter appears once and only once. You must make sure your result is the smallest in lexicographical order among all possible results.

**Example** 1:

**Input**: s = "bcabc"

**Output**: "abc"

**Example** 2:

**Input**: s = "cbacdcbc"

**Output**: "acdb"

**Constrains**:

1 <= s.length <= 104

s consists of lowercase English letters.

**104. (334) Increasing Triplet Subsequence**

Given an unsorted array return whether an increasing subsequence of length 3 exists or not in the array.

Formally the function should:

Return true if there exists i, j, k such that arr[i] < arr[j] < arr[k] given 0 ≤ i < j < k ≤ n-1 else return false.

**Note**: Your algorithm should run in O(n) time complexity and O(1) space complexity.

**Example** 1:

**Input**: [1,2,3,4,5]

**Output**: true

**Example** 2:

**Input**: [5,4,3,2,1]

**Output**: false

**105. (338) Counting Bits**

Given a non negative integer number num. For every numbers i in the range 0 ≤ i ≤ num calculate the number of 1's in their binary representation and return them as an array.

**Example** 1:

**Input**: 2

**Output**: [0,1,1]

**Example** 2:

**Input**: 5

**Output**: [0,1,1,2,1,2]

Follow up:

It is very easy to come up with a solution with run time O(n\*sizeof(integer)). But can you do it in linear time O(n) /possibly in a single pass?

Space complexity should be O(n).

Can you do it like a boss? Do it without using any builtin function like \_\_builtin\_popcount in c++ or in any other language.

**106. (345) Reverse Vowels of a String**

Write a function that takes a string as Input and reverse only the vowels of a string.

**Example** 1:

**Input**: "hello"

**Output**: "holle"

**Example** 2:

**Input**: "leetcode"

**Output**: "leotcede"

**Note**: The vowels does not include the letter "y".

**107. (371) Sum of Two Integers**

Calculate the sum of two integers a and b, but you are not allowed to use the operator + and -.

**Example** 1:

**Input**: a = 1, b = 2

**Output**: 3

**Example** 2:

**Input**: a = -2, b = 3

**Output**: 1

**108. (372) Super Pow**

Your task is to calculate ab mod 1337 where a is a positive integer and b is an extremely large positive integer given in the form of an array.

**Example** 1:

**Input**: a = 2, b = [3]

**Output**: 8

**Example** 2:

**Input**: a = 2, b = [1,0]

**Output**: 1024

**Example** 3:

**Input**: a = 1, b = [4,3,3,8,5,2]

**Output**: 1

**Example** 4:

**Input**: a = 2147483647, b = [2,0,0]

**Output**: 1198

**Constrains**:

1 <= a <= 231 - 1

1 <= b.length <= 2000

0 <= b[i] <= 9

b doesn't contain leading zeros.

**109. (380) Insert Delete GetRandom O(1)**

Implement the RandomizedSet class:

bool insert(int val) Inserts an item val into the set if not present. Returns true if the item was not present, false otherwise.

bool remove(int val) Removes an item val from the set if present. Returns true if the item was present, false otherwise.

int getRandom() Returns a random element from the current set of elements (it's guaranteed that at least one element exists when this method is called). Each element must have the same probability of being returned.

Follow up: Could you implement the functions of the class with each function works in average O(1) time?

**Example** 1:

**Input**

["RandomizedSet", "insert", "remove", "insert", "getRandom", "remove", "insert", "getRandom"]

[[], [1], [2], [2], [], [1], [2], []]

**Output**

[null, true, false, true, 2, true, false, 2]

**Explanation:**

RandomizedSet randomizedSet = new RandomizedSet();

randomizedSet.insert(1); // Inserts 1 to the set. Returns true as 1 was inserted successfully.

randomizedSet.remove(2); // Returns false as 2 does not exist in the set.

randomizedSet.insert(2); // Inserts 2 to the set, returns true. Set now contains [1,2].

randomizedSet.getRandom(); // getRandom() should return either 1 or 2 randomly.

randomizedSet.remove(1); // Removes 1 from the set, returns true. Set now contains [2].

randomizedSet.insert(2); // 2 was already in the set, so return false.

randomizedSet.getRandom(); // Since 2 is the only number in the set, getRandom() will always return 2.

**Constrains**:

-231 <= val <= 231 - 1

At most 105 calls will be made to insert, remove, and getRandom.

There will be at least one element in the data structure when getRandom is called.

**110. (384) Shuffle an Array**

Shuffle a set of numbers without duplicates.

**Example**:

// Init an array with set 1, 2, and 3.

int[] nums = {1,2,3};

Solution solution = new Solution(nums);

// Shuffle the array [1,2,3] and return its result. Any permutation of [1,2,3] must equally likely to be returned.

solution.shuffle();

// Resets the array back to its original configuration [1,2,3].

solution.reset();

// Returns the random shuffling of array [1,2,3].

solution.shuffle();

**111. (392) Is Subsequence**

Given a string s and a string t, check if s is subsequence of t.

A subsequence of a string is a new string which is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (ie, "ace" is a subsequence of "abcde" while "aec" is not).

Follow up:

If there are lots of incoming S, say S1, S2, ... , Sk where k >= 1B, and you want to check one by one to see if T has its subsequence. In this scenario, how would you change your code?

Credits:

Special thanks to @pbrother for adding this problem and creating all test cases.

**Example** 1:

**Input**: s = "abc", t = "ahbgdc"

**Output**: true

**Example** 2:

**Input**: s = "axc", t = "ahbgdc"

**Output**: false

**Constrains**:

0 <= s.length <= 100

0 <= t.length <= 10^4

Both strings consists only of lowercase characters.

**112. (412) Fizz Buzz**

Write a program that Outputs the string representation of numbers from 1 to n. But for multiples of three it should Output “Fizz” instead of the number and for the multiples of five Output “Buzz”. For numbers which are multiples of both three and five Output “FizzBuzz”.

**Example**:

n = 15,

Return:

[

"1",

"2",

"Fizz",

"4",

"Buzz",

"Fizz",

"7",

"8",

"Fizz",

"Buzz",

"11",

"Fizz",

"13",

"14",

"FizzBuzz"

]

**113. (415) Add Strings**

Given two non-negative integers num1 and num2 represented as string, return the sum of num1 and num2.

**Note**: The length of both num1 and num2 is < 5100.

Both num1 and num2 contains only digits 0-9.

Both num1 and num2 does not contain any leading zero.

You must not use any built-in BigInteger library or convert the Inputs to

integer directly.

**114. (537) Complex Number Multiplication**

Given two strings representing two complex numbers.

You need to return a string representing their multiplication. Note i2 = -1 according to the definition.

**Example** 1:

**Input**: "1+1i", "1+1i"

**Output**: "0+2i"

**Explanation**: (1 + i) \* (1 + i) = 1 + i2 + 2 \* i = 2i, and you need convert it to the form of 0+2i.

**Example** 2:

**Input**: "1+-1i", "1+-1i"

**Output**: "0+-2i"

**Explanation**: (1 - i) \* (1 - i) = 1 + i2 - 2 \* i = -2i, and you need convert it to the form of 0+-2i.

**Note**:

The Input strings will not have extra blank.

The Input strings will be given in the form of a+bi, where the integer a and b will both belong to the range of [-100, 100]. And the Output should be also in this form.

**115. (541) Reverse String II**

Given a string and an integer k, you need to reverse the first k characters for every 2k characters counting from the start of the string. If there are less than k characters left, reverse all of them. If there are less than 2k but greater than or equal to k characters, then reverse the first k characters and left the other as original.

**Example**:

**Input**: s = "abcdefg", k = 2

**Output**: "bacdfeg"

Restrictions:

The string consists of lower English letters only.

Length of the given string and k will in the range [1, 10000]

116 .(594) Longest Harmonious Subsequence

We define a harmonious array as an array where the difference between its maximum value and its minimum value is exactly 1.

Given an integer array nums, return the length of its longest harmonious subsequence among all its possible subsequences.

A subsequence of array is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements.

**Example** 1:

**Input**: nums = [1,3,2,2,5,2,3,7]

**Output**: 5

**Explanation**: The longest harmonious subsequence is [3,2,2,2,3].

**Example** 2:

**Input**: nums = [1,2,3,4]

**Output**: 2

**Example** 3:

**Input**: nums = [1,1,1,1]

**Output**: 0

**117. (599) Minimum Index Sum of Two Lists**

Suppose Andy and Doris want to choose a restaurant for dinner, and they both have a list of favorite restaurants represented by strings.

You need to help them find out their common interest with the least list index sum. If there is a choice tie between answers, Output all of them with no order requirement. You could assume there always exists an answer.

**Example** 1:

**Input**: list1 = ["Shogun","Tapioca Express","Burger King","KFC"], list2 = ["Piatti","The Grill at Torrey Pines","Hungry Hunter Steakhouse","Shogun"]

**Output**: ["Shogun"]

**Explanation**: The only restaurant they both like is "Shogun".

**Example** 2:

**Input**: list1 = ["Shogun","Tapioca Express","Burger King","KFC"], list2 = ["KFC","Shogun","Burger King"]

**Output**: ["Shogun"]

**Explanation**: The restaurant they both like and have the least index sum is "Shogun" with index sum 1 (0+1).

**Example** 3:

**Input**: list1 = ["Shogun","Tapioca Express","Burger King","KFC"], list2 = ["KFC","Burger King","Tapioca Express","Shogun"]

**Output**: ["KFC","Burger King","Tapioca Express","Shogun"]

**Example** 4:

**Input**: list1 = ["Shogun","Tapioca Express","Burger King","KFC"], list2 = ["KNN","KFC","Burger King","Tapioca Express","Shogun"]

**Output**: ["KFC","Burger King","Tapioca Express","Shogun"]

**Example** 5:

**Input**: list1 = ["KFC"], list2 = ["KFC"]

**Output**: ["KFC"]

**Constrains**:

1 <= list1.length, list2.length <= 1000

1 <= list1[i].length, list2[i].length <= 30

list1[i] and list2[i] consist of spaces ' ' and English letters.

All the stings of list1 are unique.

All the stings of list2 are unique.

**118. (561) Array Partition I**

Given an array of 2n integers, your task is to group these integers into n pairs of integer, say (a1, b1), (a2, b2), ..., (an, bn) which makes sum of min(ai, bi) for all i from 1 to n as large as possible.

**Example** 1:

**Input**: [1,4,3,2]

**Output**: 4

**Explanation**: n is 2, and the maximum sum of pairs is 4 = min(1, 2) + min(3, 4).

**Note**:

n is a positive integer, which is in the range of [1, 10000].

All the integers in the array will be in the range of [-10000, 10000].

**119. (538) Convert BST to Greater Tree**

Given the root of a Binary Search Tree (BST), convert it to a Greater Tree such that every key of the original BST is changed to the original key plus sum of all keys greater than the original key in BST.

As a reminder, a binary search tree is a tree that satisfies these **Constrains**:

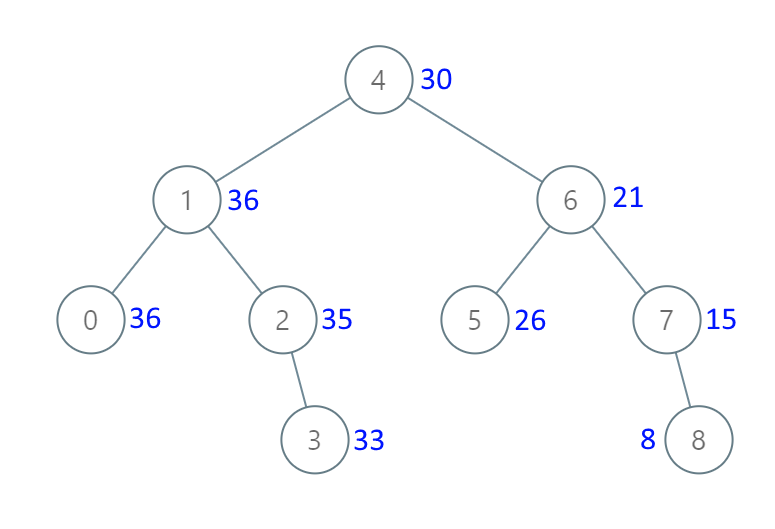
The left subtree of a node contains only nodes with keys less than the node's key.

The right subtree of a node contains only nodes with keys greater than the node's key.

Both the left and right subtrees must also be binary search trees.

**Note**: This question is the same as 1038: <https://leetcode.com/problems> /binary-search-tree-to-greater-sum-tree/

**Example** 1:



**Input**: root = [4,1,6,0,2,5,7,null,null,null,3,null,null,null,8]

**Output**: [30,36,21,36,35,26,15,null,null,null,33,null,null,null,8]

**Example** 2:

**Input**: root = [0,null,1]

**Output**: [1,null,1]

**Example** 3:

**Input**: root = [1,0,2]

**Output**: [3,3,2]

**Example** 4:

**Input**: root = [3,2,4,1]

**Output**: [7,9,4,10]

**Constrains**:

The number of nodes in the tree is in the range [0, 104].

-104 <= Node.val <= 104

All the values in the tree are unique.

root is guaranteed to be a valid binary search tree.

**120. (539) Minimum Time Difference**

Given a list of 24-hour clock time points in "Hour:Minutes" format, find the minimum minutes difference between any two time points in the list.

**Example** 1:

**Input**: ["23:59","00:00"]

**Output**: 1

**Note**:

The number of time points in the given list is at least 2 and won't exceed 20000.

The **Input** time is legal and ranges from 00:00 to 23:59.

**121. (540) Single Element in a Sorted Array**

You are given a sorted array consisting of only integers where every element appears exactly twice, except for one element which appears exactly once. Find this single element that appears only once.

Follow up: Your solution should run in O(log n) time and O(1) space.

**Example** 1:

**Input**: nums = [1,1,2,3,3,4,4,8,8]

**Output**: 2

**Example** 2:

**Input**: nums = [3,3,7,7,10,11,11]

**Output**: 10

**Constrains**:

1 <= nums.length <= 10^5

0 <= nums[i] <= 10^5

**122. (543) Diameter of Binary Tree**

Given a binary tree, you need to compute the length of the diameter of the tree. The diameter of a binary tree is the length of the longest path between any two nodes in a tree. This path may or may not pass through the root.

**Example**:

Given a binary tree

1

/ \

2 3

/ \

4 5

Return 3, which is the length of the path [4,2,1,3] or [5,2,1,3].

**Note**: The length of path between two nodes is represented by the number of edges between them.

**123. (551) Student Attendance Record I**

You are given a string representing an attendance record for a student. The record only contains the following three characters:

'A' : Absent.

'L' : Late.

'P' : Present.

A student could be rewarded if his attendance record doesn't contain more than one 'A' (absent) or more than two continuous 'L' (late).

You need to return whether the student could be rewarded according to his attendance record.

**Example** 1:

**Input**: "PPALLP"

**Output**: True

**Example** 2:

**Input**: "PPALLL"

**Output**: False

**124. (557) Reverse Words in a String III**

Given a string, you need to reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

**Example** 1:

**Input**: "Let's take LeetCode contest"

**Output**: "s'teL ekat edoCteeL tsetnoc"

**Note**: In the string, each word is separated by single space and there will not be any extra space in the string.

#### 125. interview 02.03 - [Delete Middle Node LCCI](https://leetcode-cn.com/problems/delete-middle-node-lcci/)

Implement an algorithm to delete a node in the middle (i.e., any node but the first and last node, not necessarily the exact middle) of a singly linked list, given only access to that node.

#### 126. [617. Merge Two Binary Trees](https://leetcode-cn.com/problems/merge-two-binary-trees/)

Given two binary trees and imagine that when you put one of them to cover the other, some nodes of the two trees are overlapped while the others are not.

You need to merge them into a new binary tree. The merge rule is that if two nodes overlap, then sum node values up as the new value of the merged node. Otherwise, the NOT null node will be used as the node of new tree.

**Example** 1:

**Input**:

Tree 1 Tree 2

1 2

/ \ / \

3 2 1 3

/ \ \

5 4 7

**Output**:

Merged tree:

3

/ \

4 5

/ \ \

5 4 7

**Note**: The merging process must start from the root nodes of both trees.

#### 127. 637 - [Average of Levels in Binary Tree](https://leetcode-cn.com/problems/average-of-levels-in-binary-tree/)

Given a non-empty binary tree, return the average value of the nodes on each level in the form of an array.

**Example** 1:

**Input**:

3

/ \

9 20

/ \

15 7

**Output**: [3, 14.5, 11]

**Explanation**:

The average value of nodes on level 0 is 3, on level 1 is 14.5, and on level 2 is 11. Hence return [3, 14.5, 11].

**Note**:

The range of node's value is in the range of 32-bit signed integer.

#### 128. [654. Maximum Binary Tree](https://leetcode-cn.com/problems/maximum-binary-tree/)

Given an integer array with no duplicates. A maximum tree building on this array is defined as follow:

The root is the maximum number in the array.

The left subtree is the maximum tree constructed from left part subarray divided by the maximum number.

The right subtree is the maximum tree constructed from right part subarray divided by the maximum number.

Construct the maximum tree by the given array and Output the root node of this tree.

**Example** 1:

**Input**: [3,2,1,6,0,5]

**Output**: return the tree root node representing the following tree:

6

/ \

3 5

\ /

2 0

\

1

**Note**:

The size of the given array will be in the range [1,1000].

#### 129. [657 - Robot Return to Origin](https://leetcode-cn.com/problems/robot-return-to-origin/)

There is a robot starting at position (0, 0), the origin, on a 2D plane. Given a sequence of its moves, judge if this robot ends up at (0, 0) after it completes its moves.

The move sequence is represented by a string, and the character moves[i] represents its ith move. Valid moves are R (right), L (left), U (up), and D (down). If the robot returns to the origin after it finishes all of its moves, return true. Otherwise, return false.

**Note**: The way that the robot is "facing" is irrelevant. "R" will always make the robot move to the right once, "L" will always make it move left, etc. Also, assume that the magnitude of the robot's movement is the same for each move.

**Example** 1:

**Input**: moves = "UD"

**Output**: true

**Explanation**: The robot moves up once, and then down once. All moves have the same magnitude, so it ended up at the origin where it started. Therefore, we return true.

**Example** 2:

**Input**: moves = "LL"

**Output**: false

**Explanation**: The robot moves left twice. It ends up two "moves" to the left of the origin. We return false because it is not at the origin at the end of its moves.

**Example** 3:

**Input**: moves = "RRDD"

**Output**: false

**Example** 4:

**Input**: moves = "LDRRLRUULR"

**Output**: false

**Constrains**:

1 <= moves.length <= 2 \* 104

moves only contains the characters 'U', 'D', 'L' and 'R'.

#### 130. [700 - Search in a Binary Search Tree](https://leetcode-cn.com/problems/search-in-a-binary-search-tree/)

Given the root node of a binary search tree (BST) and a value. You need to find the node in the BST that the node's value equals the given value. Return the subtree rooted with that node. If such node doesn't exist, you should return NULL.

For **Example**,

Given the tree:

4

/ \

2 7

/ \

1 3

And the value to search: 2

You should return this subtree:

2

/ \

1 3

In the Example above, if we want to search the value 5, since there is no node with value 5, we should return NULL.

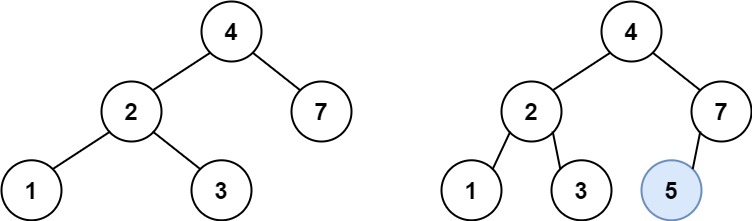
**Note** that an empty tree is represented by NULL, therefore you would see the expected Output (serialized tree format) as [], not null.

#### 131. [701 - Insert into a Binary Search Tree](https://leetcode-cn.com/problems/insert-into-a-binary-search-tree/)

You are given the root node of a binary search tree (BST) and a value to insert into the tree. Return the root node of the BST after the insertion. It is **guaranteed** that the new value does not exist in the original BST.

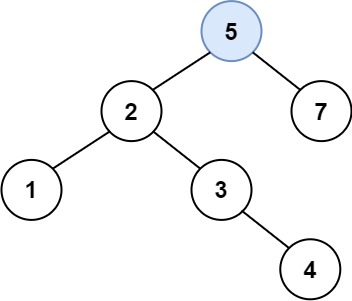
**Notice** that there may exist multiple valid ways for the insertion, as long as the tree remains a BST after insertion. You can return **any of them**.

**Example 1:**



**Input:** root = [4,2,7,1,3], val = 5

**Output:** [4,2,7,1,3,5]**Explanation:** Another accepted tree is:



**Example 2:**

**Input:** root = [40,20,60,10,30,50,70], val = 25

**Output:** [40,20,60,10,30,50,70,null,null,25]

**Example 3:**

**Input:** root = [4,2,7,1,3,null,null,null,null,null,null], val = 5

**Output:** [4,2,7,1,3,5]

**Constrains:**

* The number of nodes in the tree will be in the range [0, 104].
* -108 <= Node.val <= 108
* All the values Node.val are **unique**.
* -108 <= val <= 108
* It's **guaranteed** that val does not exist in the original BST.

#### 132. [709 - To Lower Case](https://leetcode-cn.com/problems/to-lower-case/)

#### Implement function ToLowerCase() that has a string parameter str, and returns the same string in lowercase.

**Example 1:**

**Input:** "Hello"**Output:** "hello"

**Example 2:**

**Input:** "here"**Output:** "here"

**Example 3:**

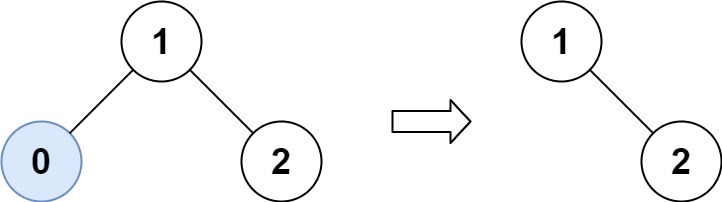
**Input:** "LOVELY"**Output:** "lovely"

#### 133. [669 - Trim a Binary Search Tree](https://leetcode-cn.com/problems/trim-a-binary-search-tree/)

Given the root of a binary search tree and the lowest and highest boundaries as low and high, trim the tree so that all its elements lies in [low, high]. Trimming the tree should **not** change the relative structure of the elements that will remain in the tree (i.e., any node's descendant should remain a descendant). It can be proven that there is a **unique answer**.

Return the root of the trimmed binary search tree. **Note** that the root may change depending on the given bounds.

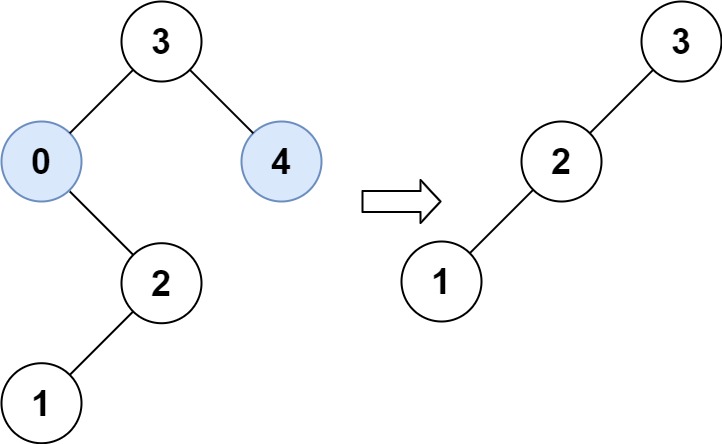
**Example 1:**



**Input:** root = [1,0,2], low = 1, high = 2

**Output:** [1,null,2]

**Example 2:**



**Input:** root = [3,0,4,null,2,null,null,1], low = 1, high = 3

**Output:** [3,2,null,1]

**Example 3:**

**Input:** root = [1], low = 1, high = 2

**Output:** [1]

**Example 4:**

**Input:** root = [1,null,2], low = 1, high = 3

**Output:** [1,null,2]

**Example 5:**

**Input:** root = [1,null,2], low = 2, high = 4

**Output:** [2]

**Constrains:**

* The number of nodes in the tree in the range [1, 104].
* 0 <= Node.val <= 104
* The value of each node in the tree is **unique**.
* root is guaranteed to be a valid binary search tree.
* 0 <= low <= high <= 104

#### 134. [682 - Baseball Game](https://leetcode-cn.com/problems/baseball-game/)

You are keeping score for a baseball game with strange rules. The game consists of several rounds, where the scores of past rounds may affect future rounds' scores.

At the beginning of the game, you start with an empty record. You are given a list of strings ops, where ops[i] is the ith operation you must apply to the record and is one of the following:

1. An integer x - Record a new score of x.
2. "+" - Record a new score that is the sum of the previous two scores. It is guaranteed there will always be two previous scores.
3. "D" - Record a new score that is double the previous score. It is guaranteed there will always be a previous score.
4. "C" - Invalidate the previous score, removing it from the record. It is guaranteed there will always be a previous score.

Return the sum of all the scores on the record.

**Example 1:**

**Input:** ops = ["5","2","C","D","+"]

**Output:** 30

**Explanation:**

"5" - Add 5 to the record, record is now [5].

"2" - Add 2 to the record, record is now [5, 2].

"C" - Invalidate and remove the previous score, record is now [5].

"D" - Add 2 \* 5 = 10 to the record, record is now [5, 10].

"+" - Add 5 + 10 = 15 to the record, record is now [5, 10, 15].

The total sum is 5 + 10 + 15 = 30.

**Example 2:**

**Input:** ops = ["5","-2","4","C","D","9","+","+"]

**Output:** 27

**Explanation:**

"5" - Add 5 to the record, record is now [5].

"-2" - Add -2 to the record, record is now [5, -2].

"4" - Add 4 to the record, record is now [5, -2, 4].

"C" - Invalidate and remove the previous score, record is now [5, -2].

"D" - Add 2 \* -2 = -4 to the record, record is now [5, -2, -4].

"9" - Add 9 to the record, record is now [5, -2, -4, 9].

"+" - Add -4 + 9 = 5 to the record, record is now [5, -2, -4, 9, 5].

"+" - Add 9 + 5 = 14 to the record, record is now [5, -2, -4, 9, 5, 14].

The total sum is 5 + -2 + -4 + 9 + 5 + 14 = 27.

**Example 3:**

**Input:** ops = ["1"]

**Output:** 1

**Constrains:**

* 1 <= ops.length <= 1000
* ops[i] is "C", "D", "+", or a string representing an integer in the range [-3 \* 104, 3 \* 104].
* For operation "+", there will always be at least two previous scores on the record.
* For operations "C" and "D", there will always be at least one previous score on the record.

#### 135. [690 - Employee Importance](https://leetcode-cn.com/problems/employee-importance/)

You are given a data structure of employee information, which includes the employee's unique id, their importance value and their direct subordinates' id.

For **Example**, employee 1 is the leader of employee 2, and employee 2 is the leader of employee 3. They have importance value 15, 10 and 5, respectively. Then employee 1 has a data structure like [1, 15, [2]], and employee 2 has [2, 10, [3]], and employee 3 has [3, 5, []]. **Note** that although employee 3 is also a subordinate of employee 1, the relationship is not direct.

Now given the employee information of a company, and an employee id, you need to return the total importance value of this employee and all their subordinates.

**Example** 1:

**Input**: [[1, 5, [2, 3]], [2, 3, []], [3, 3, []]], 1**Output**: 11**Explanation**:

Employee 1 has importance value 5, and he has two direct subordinates: employee 2 and employee 3. They both have importance value 3. So the total importance value of employee 1 is 5 + 3 + 3 = 11.

**Note**:

1. One employee has at most one direct leader and may have several subordinates.
2. The maximum number of employees won't exceed 2000.

#### 136. [693 - Binary Number with Alternating Bits](https://leetcode-cn.com/problems/binary-number-with-alternating-bits/)

Given a positive integer, check whether it has alternating bits: namely, if two adjacent bits will always have different values.

**Example 1:**

**Input:** n = 5

**Output:** true

**Explanation:** The binary representation of 5 is: 101

**Example 2:**

**Input:** n = 7

**Output:** false

**Explanation:** The binary representation of 7 is: 111.

**Example 3:**

**Input:** n = 11

**Output:** false

**Explanation:** The binary representation of 11 is: 1011.

**Example 4:**

**Input:** n = 10

**Output:** true

**Explanation:** The binary representation of 10 is: 1010.

**Example 5:**

**Input:** n = 3

**Output:** false

**Constrains:**

* 1 <= n <= 231 - 1

**137.** [**744 - Find Smallest Letter Greater Than Target**](https://leetcode-cn.com/problems/find-smallest-letter-greater-than-target/)

Given a list of sorted characters letters containing only lowercase letters, and given a target letter target, find the smallest element in the list that is larger than the given target.

Letters also wrap around. For **Example**, if the target is target = 'z' and letters = ['a', 'b'], the answer is 'a'.

**Example**s:

**Input**:

letters = ["c", "f", "j"]

target = "a"

**Output**: "c"

**Input**:

letters = ["c", "f", "j"]

target = "c"

**Output**: "f"

**Input**:

letters = ["c", "f", "j"]

target = "d"

**Output**: "f"

**Input**:

letters = ["c", "f", "j"]

target = "g"

**Output**: "j"

**Input**:

letters = ["c", "f", "j"]

target = "j"

**Output**: "c"

**Input**:

letters = ["c", "f", "j"]

target = "k"

**Output**: "c"

**Note**:

1. letters has a length in range [2, 10000].
2. letters consists of lowercase letters, and contains at least 2 unique letters.
3. target is a lowercase letter.

#### 138. [747 - Largest Number At Least Twice of Others](https://leetcode-cn.com/problems/largest-number-at-least-twice-of-others/)

In a given integer array nums, there is always exactly one largest element.

Find whether the largest element in the array is at least twice as much as every other number in the array.

If it is, return the **index** of the largest element, otherwise return -1.

**Example 1:**

**Input:** nums = [3, 6, 1, 0]

**Output:** 1

**Explanation:** 6 is the largest integer, and for every other number in the array x,

6 is more than twice as big as x. The index of value 6 is 1, so we return 1.

**Example 2:**

**Input:** nums = [1, 2, 3, 4]

**Output:** -1

**Explanation:** 4 isn't at least as big as twice the value of 3, so we return -1.

**Note:**

1. nums will have a length in the range [1, 50].
2. Every nums[i] will be an integer in the range [0, 99].

#### 139. [704 - Binary Search](https://leetcode-cn.com/problems/binary-search/)

Given a **sorted** (in ascending order) integer array nums of n elements and a target value, write a function to search target in nums. If target exists, then return its index, otherwise return -1.

**Example 1:**

**Input:** nums = [-1,0,3,5,9,12], target = 9

**Output:** 4

**Explanation:** 9 exists in nums and its index is 4

**Example 2:**

**Input:** nums = [-1,0,3,5,9,12], target = 2

**Output:** -1

**Explanation:** 2 does not exist in nums so return -1

**Note:**

1. You may assume that all elements in nums are unique.
2. n will be in the range [1, 10000].
3. The value of each element in nums will be in the range [-9999, 9999].

#### 140. [interview 08.04 - Power Set LCCI](https://leetcode-cn.com/problems/power-set-lcci/)

Write a method to return all subsets of a set. The elements in a set are pairwise distinct.

**Note**: The result set should not contain duplicated subsets.

**Example:**

**Input**: nums = [1,2,3]

**Output**:

[

[3],

  [1],

  [2],

  [1,2,3],

  [1,3],

  [2,3],

  [1,2],

  []

]

#### 141. [interview 16.01 - Swap Numbers LCCI](https://leetcode-cn.com/problems/swap-numbers-lcci/)

Write a function to swap a number in place (that is, without temporary vari­ ables).

**Example:**

**Input:** numbers = [1,2]

**Output:** [2,1]

**Note:**

* numbers.length == 2

**142.** [**interview 16.07 - Maximum LCCI**](https://leetcode-cn.com/problems/maximum-lcci/)

Write a method that finds the maximum of two numbers. You should not use if-else or any other comparison operator.

**Example:**

**Input:**  a = 1, b = 2

**Output:**  2

#### 143. [interview 05.06 - Convert Integer LCCI](https://leetcode-cn.com/problems/convert-integer-lcci/)

Write a function to determine the number of bits you would need to flip to convert integer A to integer B.

**Example1:**

**Input**: A = 29 (0b11101), B = 15 (0b01111)

**Output**: 2

**Example2:**

**Input**: A = 1，B = 2

**Output**: 2

**Note:**

1. -2147483648 <= A, B <= 2147483647

**144.** [**interview 08.03 - Magic Index LCCI**](https://leetcode-cn.com/problems/magic-index-lcci/)

A magic index in an array A[0...n-1] is defined to be an index such that A[i] = i. Given a sorted array of integers, write a method to find a magic index, if one exists, in array A. If not, return -1. If there are more than one magic index, return the smallest one.

**Example1:**

**Input**: nums = [0, 2, 3, 4, 5]

**Output**: 0

**Example2:**

**Input**: nums = [1, 1, 1]

**Output**: 1

**Note:**

1. 1 <= nums.length <= 1000000
2. This problem is the follow-up of the original problem in the book, i.e. the values are not distinct.

**145. 1304. Find N Unique Integers Sum up to Zero**

Given an integer n, return **any** array containing n **unique** integers such that they add up to 0.

**Example 1:**

**Input:** n = 5

**Output:** [-7,-1,1,3,4]

**Explanation:** These arrays also are accepted [-5,-1,1,2,3] , [-3,-1,2,-2,4].

**Example 2:**

**Input:** n = 3

**Output:** [-1,0,1]

**Example 3:**

**Input:** n = 1

**Output:** [0]

**Constrains:**

* 1 <= n <= 1000

#### 146. Decrypt String from Alphabet to Integer Mapping

Given a string s formed by digits ('0' - '9') and '#' . We want to map s to English lowercase characters as follows:

* Characters ('a' to 'i') are represented by ('1' to '9') respectively.
* Characters ('j' to 'z') are represented by ('10#' to '26#') respectively.

Return the string formed after mapping.

It's guaranteed that a unique mapping will always exist.

**Example 1:**

**Input:** s = "10#11#12"

**Output:** "jkab"

**Explanation:** "j" -> "10#" , "k" -> "11#" , "a" -> "1" , "b" -> "2".

**Example 2:**

**Input:** s = "1326#"

**Output:** "acz"

**Example 3:**

**Input:** s = "25#"

**Output:** "y"

**Example 4:**

**Input:** s = "12345678910#11#12#13#14#15#16#17#18#19#20#21#22#23#24#25#26#"

**Output:** "abcdefghijklmnopqrstuvwxyz"

**Constrains:**

* 1 <= s.length <= 1000
* s[i] only contains digits letters ('0'-'9') and '#' letter.
* s will be valid string such that mapping is always possible.

#### 147. 1310. XOR Queries of a Subarray

Given the array arr of positive integers and the array queries where

queries[i] = [Li,Ri], for each query i compute the **XOR** of elements from Li to Ri (that is, arr[Li] **xor** arr[Li+1] **xor** ... **xor** arr[Ri] ). Return an array containing the result for the given queries.

**Example 1:**

**Input:** arr = [1,3,4,8], queries = [[0,1],[1,2],[0,3],[3,3]]

**Output:** [2,7,14,8]

**Explanation:**

The binary representation of the elements in the array are:

1 = 0001

3 = 0011

4 = 0100

8 = 1000

The XOR values for queries are:

[0,1] = 1 xor 3 = 2

[1,2] = 3 xor 4 = 7

[0,3] = 1 xor 3 xor 4 xor 8 = 14

[3,3] = 8

**Example 2:**

**Input:** arr = [4,8,2,10], queries = [[2,3],[1,3],[0,0],[0,3]]

**Output:** [8,0,4,4]

**Constrains:**

* 1 <= arr.length <= 3 \* 10^4
* 1 <= arr[i] <= 10^9
* 1 <= queries.length <= 3 \* 10^4
* queries[i].length == 2
* 0 <= queries[i][0] <= queries[i][1] < arr.length

#### 148. 1313. Decompress Run-Length Encoded List

We are given a list nums of integers representing a list compressed with run-length encoding.

Consider each adjacent pair of elements [freq, val] = [nums[2\*i], nums[2\*i+1]] (with i >= 0).  For each such pair, there are freq elements with value val concatenated in a sublist. Concatenate all the sublists from left to right to generate the decompressed list.

Return the decompressed list.

**Example 1:**

**Input:** nums = [1,2,3,4]

**Output:** [2,4,4,4]

**Explanation:** The first pair [1,2] means we have freq = 1 and val = 2 so we generate the array [2].

The second pair [3,4] means we have freq = 3 and val = 4 so we generate [4,4,4].

At the end the concatenation [2] + [4,4,4] is [2,4,4,4].

**Example 2:**

**Input:** nums = [1,1,2,3]

**Output:** [1,3,3]

**Constrains:**

* 2 <= nums.length <= 100
* nums.length % 2 == 0
* 1 <= nums[i] <= 100

#### 149. 1317. Convert Integer to the Sum of Two No-Zero Integers

Given an integer n. No-Zero integer is a positive integer which **doesn't contain any 0** in its decimal representation.

Return a list of two integers [A, B] where:

* A and B are No-Zero integers.
* A + B = n

It's guarateed that there is at least one valid solution. If there are many valid solutions you can return any of them.

**Example 1:**

**Input:** n = 2

**Output:** [1,1]

**Explanation:** A = 1, B = 1. A + B = n and both A and B don't contain any 0 in their decimal representation.

**Example 2:**

**Input:** n = 11

**Output:** [2,9]

**Example 3:**

**Input:** n = 10000

**Output:** [1,9999]

**Example 4:**

**Input:** n = 69

**Output:** [1,68]

**Example 5:**

**Input:** n = 1010

**Output:** [11,999]

**Constrains:**

* 2 <= n <= 10^4

#### 150. 1323. Maximum 69 Number

Given a positive integer num consisting only of digits 6 and 9.

Return the maximum number you can get by changing **at most** one digit (6 becomes 9, and 9 becomes 6).

**Example 1:**

**Input:** num = 9669

**Output:** 9969

**Explanation:**

Changing the first digit results in 6669.

Changing the second digit results in 9969.

Changing the third digit results in 9699.

Changing the fourth digit results in 9666.

The maximum number is 9969.

**Example 2:**

**Input:** num = 9996

**Output:** 9999

**Explanation:** Changing the last digit 6 to 9 results in the maximum number.

**Example 3:**

**Input:** num = 9999

**Output:** 9999

**Explanation:** It is better not to apply any change.

**Constrains:**

* 1 <= num <= 10^4
* num's digits are 6 or 9.

#### 151. 1331. Rank Transform of an Array

Given an array of integers arr, replace each element with its rank.

The rank represents how large the element is. The rank has the following rules:

* Rank is an integer starting from 1.
* The larger the element, the larger the rank. If two elements are equal, their rank must be the same.
* Rank should be as small as possible.

**Example 1:**

**Input:** arr = [40,10,20,30]

**Output:** [4,1,2,3]

**Explanation**: 40 is the largest element. 10 is the smallest. 20 is the second smallest. 30 is the third smallest.

**Example 2:**

**Input:** arr = [100,100,100]

**Output:** [1,1,1]

**Explanation**: Same elements share the same rank.

**Example 3:**

**Input:** arr = [37,12,28,9,100,56,80,5,12]

**Output:** [5,3,4,2,8,6,7,1,3]

**Constrains:**

* 0 <= arr.length <= 105
* -109 <= arr[i] <= 109

#### 152. 1338. Reduce Array Size to The Half

Given an array arr.  You can choose a set of integers and remove all the occurrences of these integers in the array.

Return the minimum size of the set so that **at least** half of the integers of the array are removed.

**Example 1:**

**Input:** arr = [3,3,3,3,5,5,5,2,2,7]

**Output:** 2

**Explanation:** Choosing {3,7} will make the new array [5,5,5,2,2] which has size 5 (i.e equal to half of the size of the old array).

Possible sets of size 2 are {3,5},{3,2},{5,2}.

Choosing set {2,7} is not possible as it will make the new array [3,3,3,3,5,5,5] which has size greater than half of the size of the old array.

**Example 2:**

**Input:** arr = [7,7,7,7,7,7]

**Output:** 1

**Explanation:** The only possible set you can choose is {7}. This will make the new array empty.

**Example 3:**

**Input:** arr = [1,9]

**Output:** 1

**Example 4:**

**Input:** arr = [1000,1000,3,7]

**Output:** 1

**Example 5:**

**Input:** arr = [1,2,3,4,5,6,7,8,9,10]

**Output:** 5

**Constrains:**

* 1 <= arr.length <= 10^5
* arr.length is even.
* 1 <= arr[i] <= 10^5

#### 153. 1342. Number of Steps to Reduce a Number to Zero

Given a non-negative integer num, return the number of steps to reduce it to zero. If the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it.

**Example 1:**

**Input:** num = 14

**Output:** 6

**Explanation:**

Step 1) 14 is even; divide by 2 and obtain 7.

Step 2) 7 is odd; subtract 1 and obtain 6.

Step 3) 6 is even; divide by 2 and obtain 3.

Step 4) 3 is odd; subtract 1 and obtain 2.

Step 5) 2 is even; divide by 2 and obtain 1.

Step 6) 1 is odd; subtract 1 and obtain 0.

**Example 2:**

**Input:** num = 8

**Output:** 4

**Explanation:**

Step 1) 8 is even; divide by 2 and obtain 4.

Step 2) 4 is even; divide by 2 and obtain 2.

Step 3) 2 is even; divide by 2 and obtain 1.

Step 4) 1 is odd; subtract 1 and obtain 0.

**Example 3:**

**Input:** num = 123

**Output:** 12

**Constrains:**

* 0 <= num <= 10^6

#### 154. 1343. Number of Sub-arrays of Size K and Average Greater than or Equal to Threshold

Given an array of integers arr and two integers k and threshold.

Return the number of sub-arrays of size k and average greater than or equal to threshold.

**Example 1:**

**Input:** arr = [2,2,2,2,5,5,5,8], k = 3, threshold = 4

**Output:** 3

**Explanation:** Sub-arrays [2,5,5],[5,5,5] and [5,5,8] have averages 4, 5 and 6 respectively. All other sub-arrays of size 3 have averages less than 4 (the threshold).

**Example 2:**

**Input:** arr = [1,1,1,1,1], k = 1, threshold = 0

**Output:** 5

**Example 3:**

**Input:** arr = [11,13,17,23,29,31,7,5,2,3], k = 3, threshold = 5

**Output:** 6

**Explanation:** The first 6 sub-arrays of size 3 have averages greater than 5. **Note** that averages are not integers.

**Example 4:**

**Input:** arr = [7,7,7,7,7,7,7], k = 7, threshold = 7

**Output:** 1

**Example 5:**

**Input:** arr = [4,4,4,4], k = 4, threshold = 1

**Output:** 1

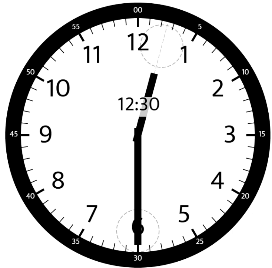
**Constrains:**

* 1 <= arr.length <= 10^5
* 1 <= arr[i] <= 10^4
* 1 <= k <= arr.length
* 0 <= threshold <= 10^4

#### 155. 1344. Angle Between Hands of a Clock

Given two numbers, hour and minutes. Return the smaller angle (in degrees) formed between the hour and the minute hand.

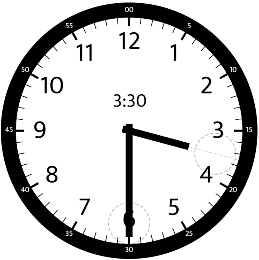
**Example 1:**



**Input:** hour = 12, minutes = 30

**Output:** 165

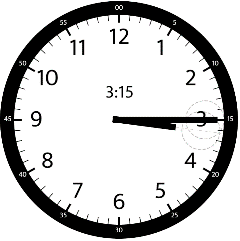
**Example 2:**



**Input:** hour = 3, minutes = 30

**Output:** 75

**Example 3:**

****

**Input:** hour = 3, minutes = 15

**Output:** 7.5

**Example 4:**

**Input:** hour = 4, minutes = 50

**Output:** 155

**Example 5:**

**Input:** hour = 12, minutes = 0

**Output:** 0

**Constrains:**

* 1 <= hour <= 12
* 0 <= minutes <= 59
* Answers within 10^-5 of the actual value will be accepted as correct.

#### 156. 1346. Check If N and Its Double Exist

Given an array arr of integers, check if there exists two integers N and M such that N is the double of M ( i.e. N = 2 \* M).

More formally check if there exists two indices i and j such that :

* i != j
* 0 <= i, j < arr.length
* arr[i] == 2 \* arr[j]

**Example 1:**

**Input:** arr = [10,2,5,3]

**Output:** true

**Explanation:** N = 10 is the double of M = 5,that is, 10 = 2 \* 5.

**Example 2:**

**Input:** arr = [7,1,14,11]

**Output:** true

**Explanation:** N = 14 is the double of M = 7,that is, 14 = 2 \* 7.

**Example 3:**

**Input:** arr = [3,1,7,11]

**Output:** false

**Explanation:** In this case does not exist N and M, such that N = 2 \* M.

**Constrains:**

* 2 <= arr.length <= 500
* -10^3 <= arr[i] <= 10^3

#### 157. 1356. Sort Integers by The Number of 1 Bits

Given an integer array arr. You have to sort the integers in the array in ascending order by the number of **1's** in their binary representation and in case of two or more integers have the same number of **1's** you have to sort them in ascending order.

Return the sorted array.

**Example 1:**

**Input:** arr = [0,1,2,3,4,5,6,7,8]

**Output:** [0,1,2,4,8,3,5,6,7]

**Explantion:** [0] is the only integer with 0 bits.

[1,2,4,8] all have 1 bit.

[3,5,6] have 2 bits.

[7] has 3 bits.

The sorted array by bits is [0,1,2,4,8,3,5,6,7]

**Example 2:**

**Input:** arr = [1024,512,256,128,64,32,16,8,4,2,1]

**Output:** [1,2,4,8,16,32,64,128,256,512,1024]

**Explantion:** All integers have 1 bit in the binary representation, you should just sort them in ascending order.

**Example 3:**

**Input:** arr = [10000,10000]

**Output:** [10000,10000]

**Example 4:**

**Input:** arr = [2,3,5,7,11,13,17,19]

**Output:** [2,3,5,17,7,11,13,19]

**Example 5:**

**Input:** arr = [10,100,1000,10000]

**Output:** [10,100,10000,1000]

**Constrains:**

* 1 <= arr.length <= 500
* 0 <= arr[i] <= 10^4

#### 158. 1360. Number of Days Between Two Dates

Write a program to count the number of days between two dates.

The two dates are given as strings, their format is YYYY-MM-DD as shown in the **Example**s.

**Example 1:**

**Input:** date1 = "2019-06-29", date2 = "2019-06-30"

**Output:** 1

**Example 2:**

**Input:** date1 = "2020-01-15", date2 = "2019-12-31"

**Output:** 15

**Constrains:**

* The given dates are valid dates between the years 1971 and 2100.

#### 159. 1365. How Many Numbers Are Smaller Than the Current Number

Given the array nums, for each nums[i] find out how many numbers in the array are smaller than it. That is, for each nums[i] you have to count the number of valid j's such that j != i **and** nums[j] < nums[i].

Return the answer in an array.

**Example 1:**

**Input:** nums = [8,1,2,2,3]

**Output:** [4,0,1,1,3]

**Explanation:**

For nums[0]=8 there exist four smaller numbers than it (1, 2, 2 and 3).

For nums[1]=1 does not exist any smaller number than it.

For nums[2]=2 there exist one smaller number than it (1).

For nums[3]=2 there exist one smaller number than it (1).

For nums[4]=3 there exist three smaller numbers than it (1, 2 and 2).

**Example 2:**

**Input:** nums = [6,5,4,8]

**Output:** [2,1,0,3]

**Example 3:**

**Input:** nums = [7,7,7,7]

**Output:** [0,0,0,0]

**Constrains:**

* 2 <= nums.length <= 500
* 0 <= nums[i] <= 100

#### 160. 1370. Increasing Decreasing String

Given a string s. You should re-order the string using the following algorithm:

1. Pick the **smallest** character from s and **append** it to the result.
2. Pick the **smallest** character from s which is greater than the last appended character to the result and **append** it.
3. Repeat step 2 until you cannot pick more characters.
4. Pick the **largest** character from s and **append** it to the result.
5. Pick the **largest** character from s which is smaller than the last appended character to the result and **append** it.
6. Repeat step 5 until you cannot pick more characters.
7. Repeat the steps from 1 to 6 until you pick all characters from s.

In each step, If the smallest or the largest character appears more than once you can choose any occurrence and append it to the result.

Return the result string after sorting s with this algorithm.

**Example 1:**

**Input:** s = "aaaabbbbcccc"

**Output:** "abccbaabccba"

**Explanation:** After steps 1, 2 and 3 of the first iteration, result = "abc"

After steps 4, 5 and 6 of the first iteration, result = "abccba"

First iteration is done. Now s = "aabbcc" and we go back to step 1

After steps 1, 2 and 3 of the second iteration, result = "abccbaabc"

After steps 4, 5 and 6 of the second iteration, result = "abccbaabccba"

**Example 2:**

**Input:** s = "rat"

**Output:** "art"

**Explanation:** The word "rat" becomes "art" after re-ordering it with the mentioned algorithm.

**Example 3:**

**Input:** s = "leetcode"

**Output:** "cdelotee"

**Example 4:**

**Input:** s = "ggggggg"

**Output:** "ggggggg"

**Example 5:**

**Input:** s = "spo"

**Output:** "ops"

**Constrains:**

* 1 <= s.length <= 500
* s contains only lower-case English letters.

#### 161. 1371. Find the Longest Substring Containing Vowels in Even Counts

Given the string s, return the size of the longest substring containing each vowel an even number of times. That is, 'a', 'e', 'i', 'o', and 'u' must appear an even number of times.

**Example 1:**

**Input:** s = "eleetminicoworoep"

**Output:** 13

**Explanation:** The longest substring is "leetminicowor" which contains two each of the vowels: **e**, **i** and **o** and zero of the vowels: **a** and **u**.

**Example 2:**

**Input:** s = "leetcodeisgreat"

**Output:** 5

**Explanation:** The longest substring is "leetc" which contains two e's.

**Example 3:**

**Input:** s = "bcbcbc"

**Output:** 6

**Explanation:** In this case, the given string "bcbcbc" is the longest because all vowels: **a**, **e**, **i**, **o** and **u** appear zero times.

**Constrains:**

* 1 <= s.length <= 5 x 10^5
* s contains only lowercase English letters.

#### 162. 1380. Lucky Numbers in a Matrix

Given a m \* n matrix of **distinct**numbers, return all lucky numbers in the matrix in **any**order.

A lucky number is an element of the matrix such that it is the minimum element in its row and maximum in its column.

**Example 1:**

**Input:** matrix = [[3,7,8],[9,11,13],[15,16,17]]

**Output:** [15]

**Explanation:** 15 is the only lucky number since it is the minimum in its row and the maximum in its column

**Example 2:**

**Input:** matrix = [[1,10,4,2],[9,3,8,7],[15,16,17,12]]

**Output:** [12]

**Explanation:** 12 is the only lucky number since it is the minimum in its row and the maximum in its column.

**Example 3:**

**Input:** matrix = [[7,8],[1,2]]

**Output:** [7]

**Constrains:**

* m == mat.length
* n == mat[i].length
* 1 <= n, m <= 50
* 1 <= matrix[i][j] <= 10^5.
* All elements in the matrix are distinct.

#### 163. 1385. Find the Distance Value Between Two Arrays

Given two integer arrays arr1 and arr2, and the integer d, return the distance value between the two arrays.

The distance value is defined as the number of elements arr1[i] such that there is not any element arr2[j] where |arr1[i]-arr2[j]| <= d.

**Example 1:**

**Input:** arr1 = [4,5,8], arr2 = [10,9,1,8], d = 2

**Output:** 2

**Explanation:**

For arr1[0]=4 we have:

|4-10|=6 > d=2

|4-9|=5 > d=2

|4-1|=3 > d=2

|4-8|=4 > d=2

For arr1[1]=5 we have:

|5-10|=5 > d=2

|5-9|=4 > d=2

|5-1|=4 > d=2

|5-8|=3 > d=2

For arr1[2]=8 we have:

**|8-10|=2 <= d=2**

**|8-9|=1 <= d=2**

|8-1|=7 > d=2

**|8-8|=0 <= d=2**

**Example 2:**

**Input:** arr1 = [1,4,2,3], arr2 = [-4,-3,6,10,20,30], d = 3

**Output:** 2

**Example 3:**

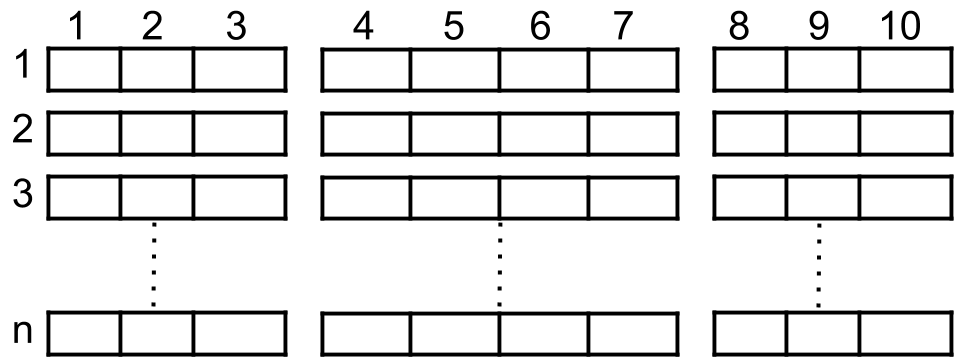
**Input:** arr1 = [2,1,100,3], arr2 = [-5,-2,10,-3,7], d = 6

**Output:** 1

**Constrains:**

* 1 <= arr1.length, arr2.length <= 500
* -10^3 <= arr1[i], arr2[j] <= 10^3
* 0 <= d <= 100

#### 164. 1386. Cinema Seat Allocation

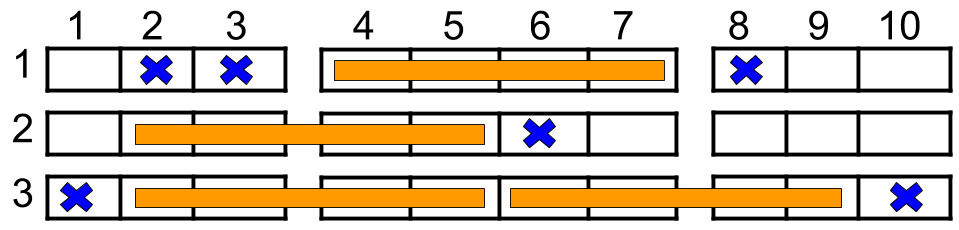


A cinema has n rows of seats, numbered from 1 to n and there are ten seats in each row, labelled from 1 to 10 as shown in the figure above.

Given the array reservedSeats containing the numbers of seats already reserved, for Example, reservedSeats[i] = [3,8] means the seat located in row **3** and labelled with **8** is already reserved.

Return the maximum number of four-person groups you can assign on the cinema seats. A four-person group occupies four adjacent seats **in one single row**. Seats across an aisle (such as [3,3] and [3,4]) are not considered to be adjacent, but there is an exceptional case on which an aisle split a four-person group, in that case, the aisle split a four-person group in the middle, which means to have two people on each side.

**Example 1:**



**Input:** n = 3, reservedSeats = [[1,2],[1,3],[1,8],[2,6],[3,1],[3,10]]

**Output:** 4

**Explanation:** The figure above shows the optimal allocation for four groups, where seats mark with blue are already reserved and contiguous seats mark with orange are for one group.

**Example 2:**

**Input:** n = 2, reservedSeats = [[2,1],[1,8],[2,6]]

**Output:** 2

**Example 3:**

**Input:** n = 4, reservedSeats = [[4,3],[1,4],[4,6],[1,7]]

**Output:** 4

**Constrains:**

* 1 <= n <= 10^9
* 1 <= reservedSeats.length <= min(10\*n, 10^4)
* reservedSeats[i].length == 2
* 1 <= reservedSeats[i][0] <= n
* 1 <= reservedSeats[i][1] <= 10
* All reservedSeats[i] are distinct.

#### 165. 1394. Find Lucky Integer in an Array

Given an array of integers arr, a lucky integer is an integer which has a frequency in the array equal to its value.

Return *a lucky integer* in the array. If there are multiple lucky integers return the **largest** of them. If there is no lucky integer return **-1**.

**Example 1:**

**Input:** arr = [2,2,3,4]

**Output:** 2

**Explanation:** The only lucky number in the array is 2 because frequency[2] == 2.

**Example 2:**

**Input:** arr = [1,2,2,3,3,3]

**Output:** 3

**Explanation:** 1, 2 and 3 are all lucky numbers, return the largest of them.

**Example 3:**

**Input:** arr = [2,2,2,3,3]

**Output:** -1

**Explanation:** There are no lucky numbers in the array.

**Example 4:**

**Input:** arr = [5]

**Output:** -1

**Example 5:**

**Input:** arr = [7,7,7,7,7,7,7]

**Output:** 7

**Constrains:**

* 1 <= arr.length <= 500
* 1 <= arr[i] <= 500

#### 166. 1396. Design Underground System

Implement the class UndergroundSystem that supports three methods:

1. checkIn(int id, string stationName, int t)

* A customer with id card equal to id, gets in the station stationName at time t.
* A customer can only be checked into one place at a time.

2. checkOut(int id, string stationName, int t)

* A customer with id card equal to id, gets out from the station stationName at time t.

3. getAverageTime(string startStation, string endStation)

* Returns the average time to travel between the startStation and the endStation.
* The average time is computed from all the previous traveling from startStation to endStation that happened **directly**.
* Call to getAverageTime is always valid.

You can assume all calls to checkIn and checkOut methods are consistent. That is, if a customer gets in at time **t1** at some station, then it gets out at time **t2** with **t2 > t1**. All events happen in chronological order.

**Example 1:**

**Input**

["UndergroundSystem","checkIn","checkIn","checkIn","checkOut","checkOut","checkOut","getAverageTime","getAverageTime","checkIn","getAverageTime","checkOut","getAverageTime"]

[[],[45,"Leyton",3],[32,"Paradise",8],[27,"Leyton",10],[45,"Waterloo",15],[27,"Waterloo",20],[32,"Cambridge",22],["Paradise","Cambridge"],["Leyton","Waterloo"],[10,"Leyton",24],["Leyton","Waterloo"],[10,"Waterloo",38],["Leyton","Waterloo"]]

**Output**

[null,null,null,null,null,null,null,14.00000,11.00000,null,11.00000,null,12.00000]

**Explanation**

UndergroundSystem undergroundSystem = new UndergroundSystem();

undergroundSystem.checkIn(45, "Leyton", 3);

undergroundSystem.checkIn(32, "Paradise", 8);

undergroundSystem.checkIn(27, "Leyton", 10);

undergroundSystem.checkOut(45, "Waterloo", 15);

undergroundSystem.checkOut(27, "Waterloo", 20);

undergroundSystem.checkOut(32, "Cambridge", 22);

undergroundSystem.getAverageTime("Paradise", "Cambridge");       // return 14.00000. There was only one travel from "Paradise" (at time 8) to "Cambridge" (at time 22)

undergroundSystem.getAverageTime("Leyton", "Waterloo");          // return 11.00000. There were two travels from "Leyton" to "Waterloo", a customer with id=45 from time=3 to time=15 and a customer with id=27 from time=10 to time=20. So the average time is ( (15-3) + (20-10) ) / 2 = 11.00000

undergroundSystem.checkIn(10, "Leyton", 24);

undergroundSystem.getAverageTime("Leyton", "Waterloo");          // return 11.00000

undergroundSystem.checkOut(10, "Waterloo", 38);

undergroundSystem.getAverageTime("Leyton", "Waterloo");          // return 12.00000

**Example 2:**

**Input**

["UndergroundSystem","checkIn","checkOut","getAverageTime","checkIn","checkOut","getAverageTime","checkIn","checkOut","getAverageTime"]

[[],[10,"Leyton",3],[10,"Paradise",8],["Leyton","Paradise"],[5,"Leyton",10],[5,"Paradise",16],["Leyton","Paradise"],[2,"Leyton",21],[2,"Paradise",30],["Leyton","Paradise"]]

**Output**

[null,null,null,5.00000,null,null,5.50000,null,null,6.66667]

**Explanation**

UndergroundSystem undergroundSystem = new UndergroundSystem();

undergroundSystem.checkIn(10, "Leyton", 3);

undergroundSystem.checkOut(10, "Paradise", 8);

undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.00000

undergroundSystem.checkIn(5, "Leyton", 10);

undergroundSystem.checkOut(5, "Paradise", 16);

undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.50000

undergroundSystem.checkIn(2, "Leyton", 21);

undergroundSystem.checkOut(2, "Paradise", 30);

undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 6.66667

**Constrains:**

* There will be at most 20000 operations.
* 1 <= id, t <= 10^6
* All strings consist of uppercase, lowercase English letters and digits.
* 1 <= stationName.length <= 10
* Answers within 10^-5 of the actual value will be accepted as correct.

**167. 1403. Minimum Subsequence in Non-Increasing Order**

Given the array nums, obtain a subsequence of the array whose sum of elements is **strictly greater** than the sum of the non included elements in such subsequence.

If there are multiple solutions, return the subsequence with **minimum size** and if there still exist multiple solutions, return the subsequence with the **maximum total sum** of all its elements. A subsequence of an array can be obtained by erasing some (possibly zero) elements from the array.

Note that the solution with the given Constrains is guaranteed to be **unique**. Also return the answer sorted in **non-increasing** order.

**Example 1:**

**Input:** nums = [4,3,10,9,8]

**Output:** [10,9]

**Explanation:** The subsequences [10,9] and [10,8] are minimal such that the sum of their elements is strictly greater than the sum of elements not included, however, the subsequence [10,9] has the maximum total sum of its elements.

**Example 2:**

**Input:** nums = [4,4,7,6,7]

**Output:** [7,7,6]

**Explanation:** The subsequence [7,7] has the sum of its elements equal to 14 which is not strictly greater than the sum of elements not included (14 = 4 + 4 + 6). Therefore, the subsequence [7,6,7] is the minimal satisfying the conditions. Note the subsequence has to returned in non-decreasing order.

**Example 3:**

**Input:** nums = [6]

**Output:** [6]

**Constrains:**

* 1 <= nums.length <= 500
* 1 <= nums[i] <= 100

**168. 1408. String Matching in an Array**

Given an array of string words. Return all strings in words which is substring of another word in **any** order.

String words[i] is substring of words[j], if can be obtained removing some characters to left and/or right side of words[j].

**Example 1:**

**Input:** words = ["mass","as","hero","superhero"]

**Output:** ["as","hero"]

**Explanation:** "as" is substring of "mass" and "hero" is substring of "superhero".

["hero","as"] is also a valid answer.

**Example 2:**

**Input:** words = ["leetcode","et","code"]

**Output:** ["et","code"]

**Explanation:** "et", "code" are substring of "leetcode".

**Example 3:**

**Input:** words = ["blue","green","bu"]

**Output:** []

**Constrains:**

* 1 <= words.length <= 100
* 1 <= words[i].length <= 30
* words[i] contains only lowercase English letters.
* It's **guaranteed** that words[i] will be unique.

**169. 1413. Minimum Value to Get Positive Step by Step Sum**

Given an array of integers nums, you start with an initial **positive** value *startValue.*

In each iteration, you calculate the step by step sum of *startValue* plus elements in nums (from left to right).

Return the minimum **positive** value of *startValue* such that the step by step sum is never less than 1.

**Example 1:**

**Input:** nums = [-3,2,-3,4,2]

**Output:** 5

**Explanation:** If you choose startValue = 4, in the third iteration your step by step sum is less than 1.

**step by step sum**

**startValue = 4 | startValue = 5 | nums**

  (4 **-3** ) = 1 | (5 **-3** ) = 2 | -3

  (1 **+2** ) = 3 | (2 **+2** ) = 4 | 2

  (3 **-3** ) = 0 | (4 **-3** ) = 1 | -3

  (0 **+4** ) = 4 | (1 **+4** ) = 5 | 4

  (4 **+2** ) = 6 | (5 **+2** ) = 7 | 2

**Example 2:**

**Input:** nums = [1,2]

**Output:** 1

**Explanation:** Minimum start value should be positive.

**Example 3:**

**Input:** nums = [1,-2,-3]

**Output:** 5

**Constrains:**

* 1 <= nums.length <= 100
* -100 <= nums[i] <= 100

**170. 1417. Reformat The String**

Given alphanumeric string s. (**Alphanumeric string** is a string consisting of lowercase English letters and digits).

You have to find a permutation of the string where no letter is followed by another letter and no digit is followed by another digit. That is, no two adjacent characters have the same type.

Return *the reformatted string* or return **an empty string** if it is impossible to reformat the string.

**Example 1:**

**Input:** s = "a0b1c2"

**Output:** "0a1b2c"

**Explanation:** No two adjacent characters have the same type in "0a1b2c". "a0b1c2", "0a1b2c", "0c2a1b" are also valid permutations.

**Example 2:**

**Input:** s = "leetcode"

**Output:** ""

**Explanation:** "leetcode" has only characters so we cannot separate them by digits.

**Example 3:**

**Input:** s = "1229857369"

**Output:** ""

**Explanation:** "1229857369" has only digits so we cannot separate them by characters.

**Example 4:**

**Input:** s = "covid2019"

**Output:** "c2o0v1i9d"

**Example 5:**

**Input:** s = "ab123"

**Output:** "1a2b3"

**Constrains:**

* 1 <= s.length <= 500
* s consists of only lowercase English letters and/or digits.

**171. 1422. Maximum Score After Splitting a String**

Given a string s of zeros and ones, *return the maximum score after splitting the string into two****non-empty****substrings* (i.e. **left** substring and **right** substring).

The score after splitting a string is the number of **zeros** in the **left** substring plus the number of **ones** in the **right** substring.

**Example 1:**

**Input:** s = "011101"

**Output:** 5

**Explanation:**

All possible ways of splitting s into two non-empty substrings are:

left = "0" and right = "11101", score = 1 + 4 = 5

left = "01" and right = "1101", score = 1 + 3 = 4

left = "011" and right = "101", score = 1 + 2 = 3

left = "0111" and right = "01", score = 1 + 1 = 2

left = "01110" and right = "1", score = 2 + 1 = 3

**Example 2:**

**Input:** s = "00111"

**Output:** 5

**Explanation:** When left = "00" and right = "111", we get the maximum score = 2 + 3 = 5

**Example 3:**

**Input:** s = "1111"

**Output:** 3

**Constrains:**

* 2 <= s.length <= 500
* The string s consists of characters '0' and '1' only.

**172. 1431. Kids With the Greatest Number of Candies**

Given the array candies and the integer extraCandies, where candies[i] represents the number of candies that the ***ith*** kid has.

For each kid check if there is a way to distribute extraCandies among the kids such that he or she can have the **greatest** number of candies among them. Notice that multiple kids can have the **greatest** number of candies.

**Example 1:**

**Input:** candies = [2,3,5,1,3], extraCandies = 3

**Output:** [true,true,true,false,true]

**Explanation:**

Kid 1 has 2 candies and if he or she receives all extra candies (3) will have 5 candies --- the greatest number of candies among the kids.

Kid 2 has 3 candies and if he or she receives at least 2 extra candies will have the greatest number of candies among the kids.

Kid 3 has 5 candies and this is already the greatest number of candies among the kids.

Kid 4 has 1 candy and even if he or she receives all extra candies will only have 4 candies.

Kid 5 has 3 candies and if he or she receives at least 2 extra candies will have the greatest number of candies among the kids.

**Example 2:**

**Input:** candies = [4,2,1,1,2], extraCandies = 1

**Output:** [true,false,false,false,false]

**Explanation:** There is only 1 extra candy, therefore only kid 1 will have the greatest number of candies among the kids regardless of who takes the extra candy.

**Example 3:**

**Input:** candies = [12,1,12], extraCandies = 10

**Output:** [true,false,true]

**Constrains:**

* 2 <= candies.length <= 100
* 1 <= candies[i] <= 100
* 1 <= extraCandies <= 50

**173. 1441. Build an Array With Stack Operations**

Given an array target and an integer n. In each iteration, you will read a number from  list = {1,2,3..., n}.

Build the target array using the following operations:

* **Push**: Read a new element from the beginning list, and push it in the array.
* **Pop**: delete the last element of the array.
* If the target array is already built, stop reading more elements.

You are guaranteed that the target array is strictly increasing, only containing numbers between 1 to n inclusive.

Return the operations to build the target array.

You are guaranteed that the answer is unique.

**Example 1:**

**Input:** target = [1,3], n = 3

**Output:** ["Push","Push","Pop","Push"]

**Explanation:**

Read number 1 and automatically push in the array -> [1]

Read number 2 and automatically push in the array then Pop it -> [1]

Read number 3 and automatically push in the array -> [1,3]

**Example 2:**

**Input:** target = [1,2,3], n = 3

**Output:** ["Push","Push","Push"]

**Example 3:**

**Input:** target = [1,2], n = 4

**Output:** ["Push","Push"]

**Explanation:** You only need to read the first 2 numbers and stop.

**Example 4:**

**Input:** target = [2,3,4], n = 4

**Output:** ["Push","Pop","Push","Push","Push"]

**Constrains:**

* 1 <= target.length <= 100
* 1 <= target[i] <= 100
* 1 <= n <= 100
* target is strictly increasing.

**174. 1446. Consecutive Characters**

Given a string s, the power of the string is the maximum length of a non-empty substring that contains only one unique character.

Return *the power* of the string.

**Example 1:**

**Input:** s = "leetcode"

**Output:** 2

**Explanation:** The substring "ee" is of length 2 with the character 'e' only.

**Example 2:**

**Input:** s = "abbcccddddeeeeedcba"

**Output:** 5

**Explanation:** The substring "eeeee" is of length 5 with the character 'e' only.

**Example 3:**

**Input:** s = "triplepillooooow"

**Output:** 5

**Example 4:**

**Input:** s = "hooraaaaaaaaaaay"

**Output:** 11

**Example 5:**

**Input:** s = "tourist"

**Output:** 1

**Constrains:**

* 1 <= s.length <= 500
* s contains only lowercase English letters.

**175. 1450. Number of Students Doing Homework at a Given Time**

Given two integer arrays startTime and endTime and given an integer queryTime.

The ith student started doing their homework at the time startTime[i] and finished it at time endTime[i].

Return *the number of students* doing their homework at time queryTime. More formally, return the number of students where queryTime lays in the interval [startTime[i], endTime[i]] inclusive.

**Example 1:**

**Input:** startTime = [1,2,3], endTime = [3,2,7], queryTime = 4

**Output:** 1

**Explanation:** We have 3 students where:

The first student started doing homework at time 1 and finished at time 3 and wasn't doing anything at time 4.

The second student started doing homework at time 2 and finished at time 2 and also wasn't doing anything at time 4.

The third student started doing homework at time 3 and finished at time 7 and was the only student doing homework at time 4.

**Example 2:**

**Input:** startTime = [4], endTime = [4], queryTime = 4

**Output:** 1

**Explanation:** The only student was doing their homework at the queryTime.

**Example 3:**

**Input:** startTime = [4], endTime = [4], queryTime = 5

**Output:** 0

**Example 4:**

**Input:** startTime = [1,1,1,1], endTime = [1,3,2,4], queryTime = 7

**Output:** 0

**Example 5:**

**Input:** startTime = [9,8,7,6,5,4,3,2,1], endTime = [10,10,10,10,10,10,10,10,10], queryTime = 5

**Output:** 5

**Constrains:**

* startTime.length == endTime.length
* 1 <= startTime.length <= 100
* 1 <= startTime[i] <= endTime[i] <= 1000
* 1 <= queryTime <= 1000

**176. 1455. Check If a Word Occurs As a Prefix of Any Word in a Sentence**

Given a sentence that consists of some words separated by a **single space**, and a searchWord.

You have to check if searchWord is a prefix of any word in sentence.

Return *the index of the word* in sentence where searchWord is a prefix of this word (**1-indexed**).

If searchWord is a prefix of more than one word, return the index of the first word **(minimum index)**. If there is no such word return **-1**.

A **prefix** of a string S is any leading contiguous substring of S.

**Example 1:**

**Input:** sentence = "i love eating burger", searchWord = "burg"

**Output:** 4

**Explanation:** "burg" is prefix of "burger" which is the 4th word in the sentence.

**Example 2:**

**Input:** sentence = "this problem is an easy problem", searchWord = "pro"

**Output:** 2

**Explanation:** "pro" is prefix of "problem" which is the 2nd and the 6th word in the sentence, but we return 2 as it's the minimal index.

**Example 3:**

**Input:** sentence = "i am tired", searchWord = "you"

**Output:** -1

**Explanation:** "you" is not a prefix of any word in the sentence.

**Example 4:**

**Input:** sentence = "i use triple pillow", searchWord = "pill"

**Output:** 4

**Example 5:**

**Input:** sentence = "hello from the other side", searchWord = "they"

**Output:** -1

**Constrains:**

* 1 <= sentence.length <= 100
* 1 <= searchWord.length <= 10
* sentence consists of lowercase English letters and spaces.
* searchWord consists of lowercase English letters.

**177. 1460. Make Two Arrays Equal by Reversing Sub-arrays**

Given two integer arrays of equal length target and arr.

In one step, you can select any **non-empty sub-array** of arr and reverse it. You are allowed to make any number of steps.

Return *True* if you can make arr equal to target, or *False* otherwise.

**Example 1:**

**Input:** target = [1,2,3,4], arr = [2,4,1,3]

**Output:** true

**Explanation:** You can follow the next steps to convert arr to target:

1- Reverse sub-array [2,4,1], arr becomes [1,4,2,3]

2- Reverse sub-array [4,2], arr becomes [1,2,4,3]

3- Reverse sub-array [4,3], arr becomes [1,2,3,4]

There are multiple ways to convert arr to target, this is not the only way to do so.

**Example 2:**

**Input:** target = [7], arr = [7]

**Output:** true

**Explanation:** arr is equal to target without any reverses.

**Example 3:**

**Input:** target = [1,12], arr = [12,1]

**Output:** true

**Example 4:**

**Input:** target = [3,7,9], arr = [3,7,11]

**Output:** false

**Explanation:** arr doesn't have value 9 and it can never be converted to target.

**Example 5:**

**Input:** target = [1,1,1,1,1], arr = [1,1,1,1,1]

**Output:** true

**Constrains:**

* target.length == arr.length
* 1 <= target.length <= 1000
* 1 <= target[i] <= 1000
* 1 <= arr[i] <= 1000

**178. 1464. Maximum Product of Two Elements in an Array**

Given the array of integers nums, you will choose two different indices i and j of that array. *Return the maximum value of* (nums[i]-1)\*(nums[j]-1).

**Example 1:**

**Input:** nums = [3,4,5,2]

**Output:** 12

**Explanation:** If you choose the indices i=1 and j=2 (indexed from 0), you will get the maximum value, that is, (nums[1]-1)\*(nums[2]-1) = (4-1)\*(5-1) = 3\*4 = 12.

**Example 2:**

**Input:** nums = [1,5,4,5]

**Output:** 16

**Explanation:** Choosing the indices i=1 and j=3 (indexed from 0), you will get the maximum value of (5-1)\*(5-1) = 16.

**Example 3:**

**Input:** nums = [3,7]

**Output:** 12

**Constrains:**

* 2 <= nums.length <= 500
* 1 <= nums[i] <= 10^3

**179. 1470. Shuffle the Array**

Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn].

*Return the array in the form* [x1,y1,x2,y2,...,xn,yn].

**Example 1:**

**Input:** nums = [2,5,1,3,4,7], n = 3

**Output:** [2,3,5,4,1,7]

**Explanation:** Since x1=2, x2=5, x3=1, y1=3, y2=4, y3=7 then the answer is [2,3,5,4,1,7].

**Example 2:**

**Input:** nums = [1,2,3,4,4,3,2,1], n = 4

**Output:** [1,4,2,3,3,2,4,1]

**Example 3:**

**Input:** nums = [1,1,2,2], n = 2

**Output:** [1,2,1,2]

**Constrains:**

* 1 <= n <= 500
* nums.length == 2n
* 1 <= nums[i] <= 10^3

**180. 1475. Final Prices With a Special Discount in a Shop**

Given the array prices where prices[i] is the price of the ith item in a shop. There is a special discount for items in the shop, if you buy the ith item, then you will receive a discount equivalent to prices[j] where j is the **minimum** index such that j > i and prices[j] <= prices[i], otherwise, you will not receive any discount at all.

*Return an array where the ith element is the final price you will pay for the ith item of the shop considering the special discount.*

**Example 1:**

**Input:** prices = [8,4,6,2,3]

**Output:** [4,2,4,2,3]

**Explanation:**

For item 0 with price[0]=8 you will receive a discount equivalent to prices[1]=4, therefore, the final price you will pay is 8 - 4 = 4.

For item 1 with price[1]=4 you will receive a discount equivalent to prices[3]=2, therefore, the final price you will pay is 4 - 2 = 2.

For item 2 with price[2]=6 you will receive a discount equivalent to prices[3]=2, therefore, the final price you will pay is 6 - 2 = 4.

For items 3 and 4 you will not receive any discount at all.

**Example 2:**

**Input:** prices = [1,2,3,4,5]

**Output:** [1,2,3,4,5]

**Explanation:** In this case, for all items, you will not receive any discount at all.

**Example 3:**

**Input:** prices = [10,1,1,6]

**Output:** [9,0,1,6]

**Constrains:**

* 1 <= prices.length <= 500
* 1 <= prices[i] <= 10^3

**181. 1480. Running Sum of 1d Array**

Given an array nums. We define a running sum of an array as runningSum[i] = sum(nums[0]…nums[i]).

Return the running sum of nums.

**Example 1:**

**Input:** nums = [1,2,3,4]

**Output:** [1,3,6,10]

**Explanation:** Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].

**Example 2:**

**Input:** nums = [1,1,1,1,1]

**Output:** [1,2,3,4,5]

**Explanation:** Running sum is obtained as follows: [1, 1+1, 1+1+1, 1+1+1+1, 1+1+1+1+1].

**Example 3:**

**Input:** nums = [3,1,2,10,1]

**Output:** [3,4,6,16,17]

**Constrains:**

* 1 <= nums.length <= 1000
* -10^6 <= nums[i] <= 10^6

**182. 1486. XOR Operation in an Array**

Given an integer n and an integer start.

Define an array nums where nums[i] = start + 2\*i (0-indexed) and n == nums.length.

Return the bitwise XOR of all elements of nums.

**Example 1:**

**Input:** n = 5, start = 0

**Output:** 8

**Explanation:** Array nums is equal to [0, 2, 4, 6, 8] where (0 ^ 2 ^ 4 ^ 6 ^ 8) = 8.

Where "^" corresponds to bitwise XOR operator.

**Example 2:**

**Input:** n = 4, start = 3

**Output:** 8

**Explanation:** Array nums is equal to [3, 5, 7, 9] where (3 ^ 5 ^ 7 ^ 9) = 8.

**Example 3:**

**Input:** n = 1, start = 7

**Output:** 7

**Example 4:**

**Input:** n = 10, start = 5

**Output:** 2

**Constrains:**

* 1 <= n <= 1000
* 0 <= start <= 1000
* n == nums.length

**183. 1491. Average Salary Excluding the Minimum and Maximum Salary**

Given an array of **unique** integers salary where salary[i] is the salary of the employee i.

Return the average salary of employees excluding the minimum and maximum salary.

**Example 1:**

**Input:** salary = [4000,3000,1000,2000]

**Output:** 2500.00000

**Explanation:** Minimum salary and maximum salary are 1000 and 4000 respectively.

Average salary excluding minimum and maximum salary is (2000+3000)/2= 2500

**Example 2:**

**Input:** salary = [1000,2000,3000]

**Output:** 2000.00000

**Explanation:** Minimum salary and maximum salary are 1000 and 3000 respectively.

Average salary excluding minimum and maximum salary is (2000)/1= 2000

**Example 3:**

**Input:** salary = [6000,5000,4000,3000,2000,1000]

**Output:** 3500.00000

**Example 4:**

**Input:** salary = [8000,9000,2000,3000,6000,1000]

**Output:** 4750.00000

**Constrains:**

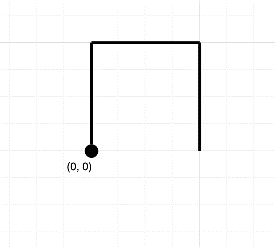
* 3 <= salary.length <= 100
* 10^3 <= salary[i] <= 10^6
* salary[i] is unique.
* Answers within 10^-5 of the actual value will be accepted as correct.

**184. 1496. Path Crossing**

Given a string path, where path[i] = 'N', 'S', 'E' or 'W', each representing moving one unit north, south, east, or west, respectively. You start at the origin (0, 0) on a 2D plane and walk on the path specified by path.

Return True if the path crosses itself at any point, that is, if at any time you are on a location you've previously visited. Return False otherwise.

**Example 1:**

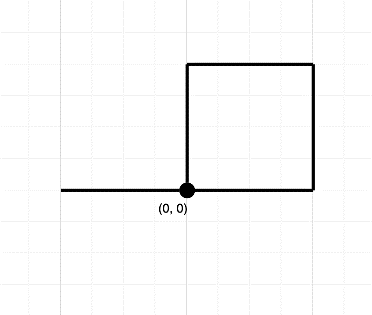


**Input:** path = "NES"

**Output:** false

**Explanation:** Notice that the path doesn't cross any point more than once.

**Example 2:**



**Input:** path = "NESWW"

**Output:** true

**Explanation:** Notice that the path visits the origin twice.

**Constrains:**

* 1 <= path.length <= 10^4
* path will only consist of characters in {'N', 'S', 'E', 'W}

**185. 1502. Can Make Arithmetic Progression From Sequence**

Given an array of numbers arr. A sequence of numbers is called an arithmetic progression if the difference between any two consecutive elements is the same.

Return true if the array can be rearranged to form an arithmetic progression, otherwise, return false.

**Example 1:**

**Input:** arr = [3,5,1]

**Output:** true

**Explanation:** We can reorder the elements as [1,3,5] or [5,3,1] with differences 2 and -2 respectively, between each consecutive elements.

**Example 2:**

**Input:** arr = [1,2,4]

**Output:** false

**Explanation:** There is no way to reorder the elements to obtain an arithmetic progression.

**Constrains:**

* 2 <= arr.length <= 1000
* -10^6 <= arr[i] <= 10^6

**186. 1512. Number of Good Pairs**

Given an array of integers nums.

A pair (i,j) is called *good* if nums[i] == nums[j] and i < j.

Return the number of *good* pairs.

**Example 1:**

**Input:** nums = [1,2,3,1,1,3]

**Output:** 4

**Explanation:** There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.

**Example 2:**

**Input:** nums = [1,1,1,1]

**Output:** 6

**Explanation:** Each pair in the array are *good*.

**Example 3:**

**Input:** nums = [1,2,3]

**Output:** 0

**Constrains:**

* 1 <= nums.length <= 100
* 1 <= nums[i] <= 100

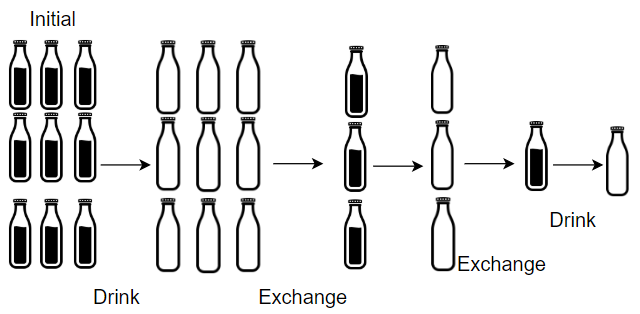
**187. 1518. Water Bottles**

Given numBottles full water bottles, you can exchange numExchange empty water bottles for one full water bottle.

The operation of drinking a full water bottle turns it into an empty bottle.

Return the **maximum** number of water bottles you can drink.

**Example 1:**

****

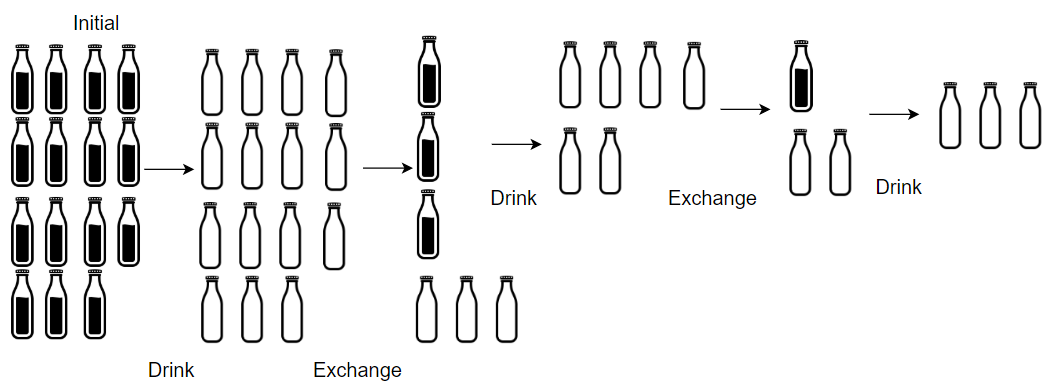
**Input:** numBottles = 9, numExchange = 3

**Output:** 13

**Explanation:** You can exchange 3 empty bottles to get 1 full water bottle.

Number of water bottles you can drink: 9 + 3 + 1 = 13.

**Example 2:**



**Input:** numBottles = 15, numExchange = 4

**Output:** 19

**Explanation:** You can exchange 4 empty bottles to get 1 full water bottle.

Number of water bottles you can drink: 15 + 3 + 1 = 19.

**Example 3:**

**Input:** numBottles = 5, numExchange = 5

**Output:** 6

**Example 4:**

**Input:** numBottles = 2, numExchange = 3

**Output:** 2

**Constrains:**

* 1 <= numBottles <= 100
* 2 <= numExchange <= 100

**188. 1523. Count Odd Numbers in an Interval Range**

Given two non-negative integers low and high. Return the *count of odd numbers between*low*and*high*(inclusive)*.

**Example 1:**

**Input:** low = 3, high = 7

**Output:** 3

**Explanation:** The odd numbers between 3 and 7 are [3,5,7].

**Example 2:**

**Input:** low = 8, high = 10

**Output:** 1

**Explanation:** The odd numbers between 8 and 10 are [9].

**Constrains:**

* 0 <= low <= high <= 10^9

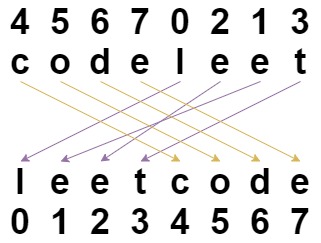
**189. 1528. Shuffle String**

Given a string s and an integer array indices of the **same length**.

The string s will be shuffled such that the character at the ith position moves to indices[i] in the shuffled string.

Return *the shuffled string*.

**Example 1:**



**Input:** s = "codeleet", indices = [4,5,6,7,0,2,1,3]

**Output:** "leetcode"

**Explanation:** As shown, "codeleet" becomes "leetcode" after shuffling.

**Example 2:**

**Input:** s = "abc", indices = [0,1,2]

**Output:** "abc"

**Explanation:** After shuffling, each character remains in its position.

**Example 3:**

**Input:** s = "aiohn", indices = [3,1,4,2,0]

**Output:** "nihao"

**Example 4:**

**Input:** s = "aaiougrt", indices = [4,0,2,6,7,3,1,5]

**Output:** "arigatou"

**Example 5:**

**Input:** s = "art", indices = [1,0,2]

**Output:** "rat"

**Constrains:**

* s.length == indices.length == n
* 1 <= n <= 100
* s contains only lower-case English letters.
* 0 <= indices[i] < n
* All values of indices are unique (i.e. indices is a permutation of the integers from 0 to n - 1).

**190. 1534. Count Good Triplets**

Given an array of integers arr, and three integers a, b and c. You need to find the number of good triplets.

A triplet (arr[i], arr[j], arr[k]) is **good** if the following conditions are true:

* 0 <= i < j < k < arr.length
* |arr[i] - arr[j]| <= a
* |arr[j] - arr[k]| <= b
* |arr[i] - arr[k]| <= c

Where |x| de**Note**s the absolute value of x.

Return*the number of good triplets*.

**Example 1:**

**Input:** arr = [3,0,1,1,9,7], a = 7, b = 2, c = 3

**Output:** 4

**Explanation:** There are 4 good triplets: [(3,0,1), (3,0,1), (3,1,1), (0,1,1)].

**Example 2:**

**Input:** arr = [1,1,2,2,3], a = 0, b = 0, c = 1

**Output:** 0

**Explanation:** No triplet satisfies all conditions.

**Constrains:**

* 3 <= arr.length <= 100
* 0 <= arr[i] <= 1000
* 0 <= a, b, c <= 1000

**191. 1539. Kth Missing Positive Number**

Given an array arr of positive integers sorted in a **strictly increasing order**, and an integer k.

*Find the*kth*positive integer that is missing from this array.*

**Example 1:**

**Input:** arr = [2,3,4,7,11], k = 5

**Output:** 9

**Explanation:** The missing positive integers are [1,5,6,8,9,10,12,13,...]. The 5th missing positive integer is 9.

**Example 2:**

**Input:** arr = [1,2,3,4], k = 2

**Output:** 6

**Explanation:** The missing positive integers are [5,6,7,...]. The 2nd missing positive integer is 6.

**Constrains:**

* 1 <= arr.length <= 1000
* 1 <= arr[i] <= 1000
* 1 <= k <= 1000
* arr[i] < arr[j] for 1 <= i < j <= arr.length

**192. 1550. Three Consecutive Odds**

Given an integer array arr, return true if there are three consecutive odd numbers in the array. Otherwise, return false.

**Example 1:**

**Input:** arr = [2,6,4,1]

**Output:** false

**Explanation:** There are no three consecutive odds.

**Example 2:**

**Input:** arr = [1,2,34,3,4,5,7,23,12]

**Output:** true

**Explanation:** [5,7,23] are three consecutive odds.

**Constrains:**

* 1 <= arr.length <= 1000
* 1 <= arr[i] <= 1000

**193. 766. Toeplitz Matrix**

A matrix is *Toeplitz* if every diagonal from top-left to bottom-right has the same element.

Now given an M x N matrix, return True if and only if the matrix is *Toeplitz*.

**Example 1:**

**Input:**

matrix = [

  [1,2,3,4],

  [5,1,2,3],

  [9,5,1,2]

]

**Output:** True

**Explanation:**

In the above grid, the diagonals are:

"[9]", "[5, 5]", "[1, 1, 1]", "[2, 2, 2]", "[3, 3]", "[4]".

In each diagonal all elements are the same, so the answer is True.

**Example 2:**

**Input:**

matrix = [

  [1,2],

  [2,2]

]

**Output:** False

**Explanation:**

The diagonal "[1, 2]" has different elements.

**Note:**

1. matrix will be a 2D array of integers.
2. matrix will have a number of rows and columns in range [1, 20].
3. matrix[i][j] will be integers in range [0, 99].

**Follow up:**

1. What if the matrix is stored on disk, and the memory is limited such that you can only load at most one row of the matrix into the memory at once?
2. What if the matrix is so large that you can only load up a partial row into the memory at once?

**194. 771. Jewels and Stones**

You're given strings J representing the types of stones that are jewels, and S representing the stones you have.  Each character in S is a type of stone you have.  You want to know how many of the stones you have are also jewels.

The letters in J are guaranteed distinct, and all characters in J and S are letters. Letters are case sensitive, so "a" is considered a different type of stone from "A".

**Example 1:**

**Input:** J = "aA", S = "aAAbbbb"

**Output:** 3

**Example 2:**

**Input:** J = "z", S = "ZZ"

**Output:** 0

**Note:**

* S and J will consist of letters and have length at most 50.
* The characters in J are distinct.

**195. 796. Rotate String**

We are given two strings, A and B.

A *shift on A* consists of taking string A and moving the leftmost character to the rightmost position. For **Example**, if A = 'abcde', then it will be 'bcdea' after one shift on A. Return True if and only if A can become B after some number of shifts on A.

**Example 1:**

**Input:** A = 'abcde', B = 'cdeab'

**Output:** true

**Example 2:**

**Input:** A = 'abcde', B = 'abced'

**Output:** false

**Note:**

* A and B will have length at most 100.

**196. 804. Unique Morse Code Words**

International Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes, as follows: "a" maps to ".-", "b" maps to "-...", "c" maps to "-.-.", and so on.

For convenience, the full table for the 26 letters of the English alphabet is given below:

[".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-..-","-.--","--.."]

Now, given a list of words, each word can be written as a concatenation of the Morse code of each letter. For **Example**, "cab" can be written as "-.-..--...", (which is the concatenation "-.-." + ".-" + "-..."). We'll call such a concatenation, the transformation of a word.

Return the number of different transformations among all words we have.

**Example:**

**Input:** words = ["gin", "zen", "gig", "msg"]

**Output:** 2

**Explanation:**

The transformation of each word is:

"gin" -> "--...-."

"zen" -> "--...-."

"gig" -> "--...--."

"msg" -> "--...--."

There are 2 different transformations, "--...-." and "--...--.".

**Note:**

* The length of words will be at most 100.
* Each words[i] will have length in range [1, 12].
* words[i] will only consist of lowercase letters.

**197. 811. Subdomain Visit Count**

A website domain like "discuss.leetcode.com" consists of various subdomains. At the top level, we have "com", at the next level, we have "leetcode.com", and at the lowest level, "discuss.leetcode.com". When we visit a domain like "discuss.leetcode.com", we will also visit the parent domains "leetcode.com" and "com" implicitly.

Now, call a "count-paired domain" to be a count (representing the number of visits this domain received), followed by a space, followed by the address. An **Example** of a count-paired domain might be "9001 discuss.leetcode.com".

We are given a list cpdomains of count-paired domains. We would like a list of count-paired domains, (in the same format as the **Input**, and in any order), that explicitly counts the number of visits to each subdomain.

**Example 1:**

**Input:**

["9001 discuss.leetcode.com"]

**Output:**

["9001 discuss.leetcode.com", "9001 leetcode.com", "9001 com"]

**Explanation:**

We only have one website domain: "discuss.leetcode.com". As discussed above, the subdomain "leetcode.com" and "com" will also be visited. So they will all be visited 9001 times.

**Example 2:**

**Input:**

["900 google.mail.com", "50 yahoo.com", "1 intel.mail.com", "5 wiki.org"]

**Output:**

["901 mail.com","50 yahoo.com","900 google.mail.com","5 wiki.org","5 org","1 intel.mail.com","951 com"]

**Explanation:**

We will visit "google.mail.com" 900 times, "yahoo.com" 50 times, "intel.mail.com" once and "wiki.org" 5 times. For the subdomains, we will visit "mail.com" 900 + 1 = 901 times, "com" 900 + 50 + 1 = 951 times, and "org" 5 times.

**Notes:**

* The length of cpdomains will not exceed 100.
* The length of each domain name will not exceed 100.
* Each address will have either 1 or 2 "." characters.
* The **Input** count in any count-paired domain will not exceed 10000.
* The answer **Output** can be returned in any order.

**198. 819. Most Common Word**

Given a paragraph and a list of banned words, return the most frequent word that is not in the list of banned words.  It is guaranteed there is at least one word that isn't banned, and that the answer is unique.

Words in the list of banned words are given in lowercase, and free of punctuation.  Words in the paragraph are not case sensitive.  The answer is in lowercase.

**Example:**

**Input:**

paragraph = "Bob hit a ball, the hit BALL flew far after it was hit."

banned = ["hit"]

**Output:** "ball"

**Explanation:**

"hit" occurs 3 times, but it is a banned word.

"ball" occurs twice (and no other word does), so it is the most frequent non-banned word in the paragraph.

**Note** that words in the paragraph are not case sensitive,

that punctuation is ignored (even if adjacent to words, such as "ball,"),

and that "hit" isn't the answer even though it occurs more because it is banned.

**Note:**

* 1 <= paragraph.length <= 1000.
* 0 <= banned.length <= 100.
* 1 <= banned[i].length <= 10.
* The answer is unique, and written in lowercase (even if its occurrences in paragraph may have uppercase symbols, and even if it is a proper noun.)
* paragraph only consists of letters, spaces, or the punctuation symbols !?',;.
* There are no hyphens or hyphenated words.
* Words only consist of letters, never apostrophes or other punctuation symbols.

**199. 821. Shortest Distance to a Character**

Given a string S and a character C, return an array of integers representing the shortest distance from the character C in the string.

**Example 1:**

**Input:** S = "loveleetcode", C = 'e'

**Output:** [3, 2, 1, 0, 1, 0, 0, 1, 2, 2, 1, 0]

**Note:**

1. S string length is in [1, 10000].
2. C is a single character, and guaranteed to be in string S.
3. All letters in S and C are lowercase.

**200. 824. Goat Latin**

A sentence S is given, composed of words separated by spaces. Each word consists of lowercase and uppercase letters only.

We would like to convert the sentence to "*Goat Latin"* (a made-up language similar to Pig Latin.)

The rules of Goat Latin are as follows:

* If a word begins with a vowel (a, e, i, o, or u), append "ma" to the end of the word.  
  For **Example**, the word 'apple' becomes 'applema'.
* If a word begins with a consonant (i.e. not a vowel), remove the first letter and append it to the end, then add "ma".  
  For **Example**, the word "goat" becomes "oatgma".
* Add one letter 'a' to the end of each word per its word index in the sentence, starting with 1.  
  For **Example**, the first word gets "a" added to the end, the second word gets "aa" added to the end and so on.

Return the final sentence representing the conversion from S to Goat Latin.

**Example 1:**

**Input:** "I speak Goat Latin"

**Output:** "Imaa peaksmaaa oatGmaaaa atinLmaaaaa"

**Example 2:**

**Input:** "The quick brown fox jumped over the lazy dog"

**Output:** "heTmaa uickqmaaa rownbmaaaa oxfmaaaaa umpedjmaaaaaa overmaaaaaaa hetmaaaaaaaa azylmaaaaaaaaa ogdmaaaaaaaaaa"

**Note**s:

* S contains only uppercase, lowercase and spaces. Exactly one space between each word.
* 1 <= S.length <= 150.

**201. 830. Positions of Large Groups**

In a string s of lowercase letters, these letters form consecutive groups of the same character.

For **Example**, a string like s = "abbxxxxzyy" has the groups "a", "bb", "xxxx", "z", and "yy".

A group is identified by an interval [start, end], where start and end de**Note** the start and end indices (inclusive) of the group. In the above **Example**, "xxxx" has the interval [3,6].

A group is considered **large** if it has 3 or more characters.

Return *the intervals of every* ***large*** *group sorted in****increasing order by start index***.

**Example 1:**

**Input:** s = "abbxxxxzzy"

**Output:** [[3,6]]

**Explanation**: "xxxx" is the only large group with start index 3 and end index 6.

**Example 2:**

**Input:** s = "abc"

**Output:** []

**Explanation**: We have groups "a", "b", and "c", none of which are large groups.

**Example 3:**

**Input:** s = "abcdddeeeeaabbbcd"

**Output:** [[3,5],[6,9],[12,14]]

**Explanation**: The large groups are "ddd", "eeee", and "bbb".

**Example 4:**

**Input:** s = "aba"

**Output:** []

**Constrains:**

* 1 <= s.length <= 1000
* s contains lower-case English letters only.

**202. 836. Rectangle Overlap**

An axis-aligned rectangle is represented as a list [x1, y1, x2, y2], where (x1, y1) is the coordinate of its bottom-left corner, and (x2, y2) is the coordinate of its top-right corner. Its top and bottom edges are parallel to the X-axis, and its left and right edges are parallel to the Y-axis.

Two rectangles overlap if the area of their intersection is **positive**. To be clear, two rectangles that only touch at the corner or edges do not overlap.

Given two axis-aligned rectangles rec1 and rec2, return true *if they overlap, otherwise return* false.

**Example 1:**

**Input:** rec1 = [0,0,2,2], rec2 = [1,1,3,3]

**Output:** true

**Example 2:**

**Input:** rec1 = [0,0,1,1], rec2 = [1,0,2,1]

**Output:** false

**Example 3:**

**Input:** rec1 = [0,0,1,1], rec2 = [2,2,3,3]

**Output:** false

**Constrains:**

* rect1.length == 4
* rect2.length == 4
* -109 <= rec1[i], rec2[i] <= 109
* rec1[0] <= rec1[2] and rec1[1] <= rec1[3]
* rec2[0] <= rec2[2] and rec2[1] <= rec2[3]

**203. 844. Backspace String Compare**

Given two strings S and T, return if they are equal when both are typed into empty text editors. # means a backspace character.

**Note** that after backspacing an empty text, the text will continue empty.

**Example 1:**

**Input:** S = "ab#c", T = "ad#c"

**Output:** true

**Explanation**: Both S and T become "ac".

**Example 2:**

**Input:** S = "ab##", T = "c#d#"

**Output:** true

**Explanation**: Both S and T become "".

**Example 3:**

**Input:** S = "a##c", T = "#a#c"

**Output:** true

**Explanation**: Both S and T become "c".

**Example 4:**

**Input:** S = "a#c", T = "b"

**Output:** false

**Explanation**: S becomes "c" while T becomes "b".

**Note**:

* 1 <= S.length <= 200
* 1 <= T.length <= 200
* S and T only contain lowercase letters and '#' characters.

**Follow up:**

* Can you solve it in O(N) time and O(1) space?

**204. 852. Peak Index in a Mountain Array**

Let's call an array arr a **mountain** if the following properties hold:

* arr.length >= 3
* There exists some i with 0 < i < arr.length - 1 such that:
  + arr[0] < arr[1] < ... arr[i-1] < arr[i]
  + arr[i] > arr[i+1] > ... > arr[arr.length - 1]

Given an integer array arr that is **guaranteed** to be a mountain, return any i such that arr[0] < arr[1] < ... arr[i - 1] < arr[i] > arr[i + 1] > ... > arr[arr.length - 1].

**Example 1:**

**Input:** arr = [0,1,0]

**Output:** 1

**Example 2:**

**Input:** arr = [0,2,1,0]

**Output:** 1

**Example 3:**

**Input:** arr = [0,10,5,2]

**Output:** 1

**Example 4:**

**Input:** arr = [3,4,5,1]

**Output:** 2

**Example 5:**

**Input:** arr = [24,69,100,99,79,78,67,36,26,19]

**Output:** 2

**Constrains:**

* 3 <= arr.length <= 104
* 0 <= arr[i] <= 106
* arr is **guaranteed** to be a mountain array.

**205. 860. Lemonade Change**

At a lemonade stand, each lemonade costs $5.

Customers are standing in a queue to buy from you, and order one at a time (in the order specified by bills).

Each customer will only buy one lemonade and pay with either a $5, $10, or $20 bill.  You must provide the correct change to each customer, so that the net transaction is that the customer pays $5.

**Note** that you don't have any change in hand at first.

Return true if and only if you can provide every customer with correct change.

**Example 1:**

**Input:** [5,5,5,10,20]

**Output:** true

**Explanation:**

From the first 3 customers, we collect three $5 bills in order.

From the fourth customer, we collect a $10 bill and give back a $5.

From the fifth customer, we give a $10 bill and a $5 bill.

Since all customers got correct change, we **Output** true.

**Example 2:**

**Input:** [5,5,10]

**Output:** true

**Example 3:**

**Input:** [10,10]

**Output:** false

**Example 4:**

**Input:** [5,5,10,10,20]

**Output:** false

**Explanation:**

From the first two customers in order, we collect two $5 bills.

For the next two customers in order, we collect a $10 bill and give back a $5 bill.

For the last customer, we can't give change of $15 back because we only have two $10 bills.

Since not every customer received correct change, the answer is false.

**Note:**

* 0 <= bills.length <= 10000
* bills[i] will be either 5, 10, or 20.

**206. 867. Transpose Matrix**

Given a matrix A, return the transpose of A.

The transpose of a matrix is the matrix flipped over it's main diagonal, switching the row and column indices of the matrix.

**Example 1:**

**Input:** [[1,2,3],[4,5,6],[7,8,9]]

**Output:** [[1,4,7],[2,5,8],[3,6,9]]

**Example 2:**

**Input:** [[1,2,3],[4,5,6]]

**Output:** [[1,4],[2,5],[3,6]]

**Note:**

1. 1 <= A.length <= 1000
2. 1 <= A[0].length <= 1000

**207. 868. Binary Gap**

Given a positive integer n, find and return *the* ***longest distance*** *between any two* ***adjacent***1*'s in the binary representation of* n*. If there are no two adjacent* 1*'s, return* 0*.*

Two 1's are **adjacent** if there are only 0's separating them (possibly no 0's). The **distance** between two 1's is the absolute difference between their bit positions. For **Example**, the two 1's in "1001" have a distance of 3.

**Example 1:**

**Input:** n = 22

**Output:** 2

**Explanation:** 22 in binary is "10110".

The first adjacent pair of 1's is "10110" with a distance of 2.

The second adjacent pair of 1's is "10110" with a distance of 1.

The answer is the largest of these two distances, which is 2.

**Note** that "10110" is not a valid pair since there is a 1 separating the two 1's underlined.

**Example 2:**

**Input:** n = 5

**Output:** 2

**Explanation:** 5 in binary is "101".

**Example 3:**

**Input:** n = 6

**Output:** 1

**Explanation:** 6 in binary is "110".

**Example 4:**

**Input:** n = 8

**Output:** 0

**Explanation:** 8 in binary is "1000".

There aren't any adjacent pairs of 1's in the binary representation of 8, so we return 0.

**Example 5:**

**Input:** n = 1

**Output:** 0

**Constrains:**

* 1 <= n <= 109

**208. 876. Middle of the Linked List**

Given a non-empty, singly linked list with head node head, return a middle node of linked list.

If there are two middle nodes, return the second middle node.

**Example 1:**

**Input:** [1,2,3,4,5]

**Output:** Node 3 from this list (Serialization: [3,4,5])

The returned node has value 3. (The judge's serialization of this node is [3,4,5]).

**Note** that we returned a ListNode object ans, such that:

ans.val = 3, ans.next.val = 4, ans.next.next.val = 5, and ans.next.next.next = NULL.

**Example 2:**

**Input:** [1,2,3,4,5,6]

**Output:** Node 4 from this list (Serialization: [4,5,6])

Since the list has two middle nodes with values 3 and 4, we return the second one.

**Note:**

* The number of nodes in the given list will be between 1 and 100.

**209. 884. Uncommon Words from Two Sentences**

We are given two sentences A and B.  (A *sentence* is a string of space separated words.  Each *word* consists only of lowercase letters.)

A word is *uncommon* if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Return a list of all uncommon words.

You may return the list in any order.

**Example 1:**

**Input:** A = "this apple is sweet", B = "this apple is sour"

**Output:** ["sweet","sour"]

**Example 2:**

**Input:** A = "apple apple", B = "banana"

**Output:** ["banana"]

**Note:**

1. 0 <= A.length <= 200
2. 0 <= B.length <= 200
3. A and B both contain only spaces and lowercase letters.

**210. 896. Monotonic Array**

An array is *monotonic* if it is either monotone increasing or monotone decreasing.

An array A is monotone increasing if for all i <= j, A[i] <= A[j].  An array A is monotone decreasing if for all i <= j, A[i] >= A[j].

Return true if and only if the given array A is monotonic.

**Example 1:**

**Input:** [1,2,2,3]

**Output:** true

**Example 2:**

**Input:** [6,5,4,4]

**Output:** true

**Example 3:**

**Input:** [1,3,2]

**Output:** false

**Example 4:**

**Input:** [1,2,4,5]

**Output:** true

**Example 5:**

**Input:** [1,1,1]

**Output:** true

**Note:**

1. 1 <= A.length <= 50000
2. -100000 <= A[i] <= 100000

**211. 905. Sort Array By Parity**

Given an array A of non-negative integers, return an array consisting of all the even elements of A, followed by all the odd elements of A.

You may return any answer array that satisfies this condition.

**Example 1:**

**Input:** [3,1,2,4]

**Output:** [2,4,3,1]

The **Output**s [4,2,3,1], [2,4,1,3], and [4,2,1,3] would also be accepted.

**Note:**

1. 1 <= A.length <= 5000
2. 0 <= A[i] <= 5000

**212. 917. Reverse Only Letters**

Given a string S, return the "reversed" string where all characters that are not a letter stay in the same place, and all letters reverse their positions.

**Example 1:**

**Input:** "ab-cd"

**Output:** "dc-ba"

**Example 2:**

**Input:** "a-bC-dEf-ghIj"

**Output:** "j-Ih-gfE-dCba"

**Example 3:**

**Input:** "Test1ng-Leet=code-Q!"

**Output:** "Qedo1ct-eeLg=ntse-T!"

**Note:**

1. S.length <= 100
2. 33 <= S[i].ASCIIcode <= 122
3. S doesn't contain \ or "

**213. 922. Sort Array By Parity II**

Given an array A of non-negative integers, half of the integers in A are odd, and half of the integers are even.

Sort the array so that whenever A[i] is odd, i is odd; and whenever A[i] is even, i is even.

You may return any answer array that satisfies this condition.

**Example 1:**

**Input:** [4,2,5,7]

**Output:** [4,5,2,7]

**Explanation:** [4,7,2,5], [2,5,4,7], [2,7,4,5] would also have been accepted.

**Note:**

1. 2 <= A.length <= 20000
2. A.length % 2 == 0
3. 0 <= A[i] <= 1000

**214. 925. Long Pressed Name**

Your friend is typing his name into a keyboard.  Sometimes, when typing a character c, the key might get *long pressed*, and the character will be typed 1 or more times.

You examine the typed characters of the keyboard.  Return True if it is possible that it was your friends name, with some characters (possibly none) being long pressed.

**Example 1:**

**Input:** name = "alex", typed = "aaleex"

**Output:** true

**Explanation:** 'a' and 'e' in 'alex' were long pressed.

**Example 2:**

**Input:** name = "saeed", typed = "ssaaedd"

**Output:** false

**Explanation:** 'e' must have been pressed twice, but it wasn't in the typed **Output**.

**Example 3:**

**Input:** name = "leelee", typed = "lleeelee"

**Output:** true

**Example 4:**

**Input:** name = "laiden", typed = "laiden"

**Output:** true

**Explanation:** It's not necessary to long press any character.

**Constrains:**

* 1 <= name.length <= 1000
* 1 <= typed.length <= 1000
* The characters of name and typed are lowercase letters.

**215. 929. Unique Email Addresses**

Every email consists of a local name and a domain name, separated by the @ sign.

For **Example**, in alice@leetcode.com, alice is the local name, and leetcode.com is the domain name.

Besides lowercase letters, these emails may contain '.'s or '+'s.

If you add periods ('.') between some characters in the **local name** part of an email address, mail sent there will be forwarded to the same address without dots in the local name.  For **Example**, "alice.z@leetcode.com" and "alicez@leetcode.com" forward to the same email address.  (**Note** that this rule does not apply for domain names.)

If you add a plus ('+') in the **local name**, everything after the first plus sign will be **ignored**. This allows certain emails to be filtered, for **Example** m.y+name@email.com will be forwarded to my@email.com.  (Again, this rule does not apply for domain names.)

It is possible to use both of these rules at the same time.

Given a list of emails, we send one email to each address in the list.  How many different addresses actually receive mails?

**Example 1:**

**Input:** ["test.email+alex@leetcode.com","test.e.mail+bob.cathy@leetcode.com","testemail+david@lee.tcode.com"]

**Output:** 2

**Explanation:** "testemail@leetcode.com" and "testemail@lee.tcode.com" actually receive mails

**Note:**

* 1 <= emails[i].length <= 100
* 1 <= emails.length <= 100
* Each emails[i] contains exactly one '@' character.
* All local and domain names are non-empty.
* Local names do not start with a '+' character.

**216. 941. Valid Mountain Array**

Given an array A of integers, return true if and only if it is a *valid mountain array*.

Recall that A is a mountain array if and only if:

* A.length >= 3
* There exists some i with 0 < i < A.length - 1 such that:
  + A[0] < A[1] < ... A[i-1] < A[i]
  + A[i] > A[i+1] > ... > A[A.length - 1]

**Example 1:**

**Input:** [2,1]

**Output:** false

**Example 2:**

**Input:** [3,5,5]

**Output:** false

**Example 3:**

**Input:** [0,3,2,1]

**Output:** true

**Note:**

1. 0 <= A.length <= 10000
2. 0 <= A[i] <= 10000

**217. 942. DI String Match**

Given a string S that **only** contains "I" (increase) or "D" (decrease), let N = S.length.

Return **any** permutation A of [0, 1, ..., N] such that for all i = 0, ..., N-1:

* If S[i] == "I", then A[i] < A[i+1]
* If S[i] == "D", then A[i] > A[i+1]

**Example 1:**

**Input:** "IDID"

**Output:** [0,4,1,3,2]

**Example 2:**

**Input:** "III"

**Output:** [0,1,2,3]

**Example 3:**

**Input:** "DDI"

**Output:** [3,2,0,1]

**Note:**

1. 1 <= S.length <= 10000
2. S only contains characters "I" or "D".

**218. 953. Verifying an Alien Dictionary**

In an alien language, surprisingly they also use english lowercase letters, but possibly in a different order. The order of the alphabet is some permutation of lowercase letters.

Given a sequence of words written in the alien language, and the order of the alphabet, return true if and only if the given words are sorted lexicographicaly in this alien language.

**Example 1:**

**Input:** words = ["hello","leetcode"], order = "hlabcdefgijkmnopqrstuvwxyz"

**Output:** true

**Explanation:** As 'h' comes before 'l' in this language, then the sequence is sorted.

**Example 2:**

**Input:** words = ["word","world","row"], order = "worldabcefghijkmnpqstuvxyz"

**Output:** false

**Explanation:** As 'd' comes after 'l' in this language, then words[0] > words[1], hence the sequence is unsorted.

**Example 3:**

**Input:** words = ["apple","app"], order = "abcdefghijklmnopqrstuvwxyz"

**Output:** false

**Explanation:** The first three characters "app" match, and the second string is shorter (in size.) According to lexicographical rules "apple" > "app", because 'l' > '∅', where '∅' is defined as the blank character which is less than any other character ([More info](https://en.wikipedia.org/wiki/Lexicographical_order)).

**Constrains:**

* 1 <= words.length <= 100
* 1 <= words[i].length <= 20
* order.length == 26
* All characters in words[i] and order are English lowercase letters.

**219. 977. Squares of a Sorted Array**

Given an array of integers A sorted in non-decreasing order, return an array of the squares of each number, also in sorted non-decreasing order.

**Example 1:**

**Input:** [-4,-1,0,3,10]

**Output:** [0,1,9,16,100]

**Example 2:**

**Input:** [-7,-3,2,3,11]

**Output:** [4,9,9,49,121]

**Note:**

1. 1 <= A.length <= 10000
2. -10000 <= A[i] <= 10000
3. A is sorted in non-decreasing order.

**220. 985. Sum of Even Numbers After Queries**

We have an array A of integers, and an array queries of queries.

For the i-th query val = queries[i][0], index = queries[i][1], we add val to A[index].  Then, the answer to the i-th query is the sum of the even values of A.

*(Here, the given index = queries[i][1] is a 0-based index, and each query permanently modifies the array A.)*

Return the answer to all queries.  Your answer array should have answer[i] as the answer to the i-th query.

**Example 1:**

**Input:** A = [1,2,3,4], queries = [[1,0],[-3,1],[-4,0],[2,3]]

**Output:** [8,6,2,4]

**Explanation:**

At the beginning, the array is [1,2,3,4].

After adding 1 to A[0], the array is [2,2,3,4], and the sum of even values is 2 + 2 + 4 = 8.

After adding -3 to A[1], the array is [2,-1,3,4], and the sum of even values is 2 + 4 = 6.

After adding -4 to A[0], the array is [-2,-1,3,4], and the sum of even values is -2 + 4 = 2.

After adding 2 to A[3], the array is [-2,-1,3,6], and the sum of even values is -2 + 6 = 4.

**Note:**

1. 1 <= A.length <= 10000
2. -10000 <= A[i] <= 10000
3. 1 <= queries.length <= 10000
4. -10000 <= queries[i][0] <= 10000
5. 0 <= queries[i][1] < A.length

**221. 989. Add to Array-Form of Integer**

For a non-negative integer X, the *array-form of X* is an array of its digits in left to right order.  For **Example**, if X = 1231, then the array form is [1,2,3,1].

Given the array-form A of a non-negative integer X, return the array-form of the integer X+K.

**Example 1:**

**Input:** A = [1,2,0,0], K = 34

**Output:** [1,2,3,4]

**Explanation:** 1200 + 34 = 1234

**Example 2:**

**Input:** A = [2,7,4], K = 181

**Output:** [4,5,5]

**Explanation:** 274 + 181 = 455

**Example 3:**

**Input:** A = [2,1,5], K = 806

**Output:** [1,0,2,1]

**Explanation:** 215 + 806 = 1021

**Example 4:**

**Input:** A = [9,9,9,9,9,9,9,9,9,9], K = 1

**Output:** [1,0,0,0,0,0,0,0,0,0,0]

**Explanation:** 9999999999 + 1 = 10000000000

**Note：**

1. 1 <= A.length <= 10000
2. 0 <= A[i] <= 9
3. 0 <= K <= 10000
4. If A.length > 1, then A[0] != 0

**222. 1002. Find Common Characters**

Given an array A of strings made only from lowercase letters, return a list of all characters that show up in all strings within the list **(including duplicates)**.  For **Example**, if a character occurs 3 times in all strings but not 4 times, you need to include that character three times in the final answer.

You may return the answer in any order.

**Example 1:**

**Input:** ["bella","label","roller"]

**Output:** ["e","l","l"]

**Example 2:**

**Input:** ["cool","lock","cook"]

**Output:** ["c","o"]

**Note:**

1. 1 <= A.length <= 100
2. 1 <= A[i].length <= 100
3. A[i][j] is a lowercase letter

**223. 1005. Maximize Sum Of Array After K Negations**

Given an array A of integers, we **must** modify the array in the following way: we choose an i and replace A[i] with -A[i], and we repeat this process K times in total.  (We may choose the same index i multiple times.)

Return the largest possible sum of the array after modifying it in this way.

**Example 1:**

**Input:** A = [4,2,3], K = 1

**Output:** 5

**Explanation:** Choose indices (1,) and A becomes [4,-2,3].

**Example 2:**

**Input:** A = [3,-1,0,2], K = 3

**Output:** 6

**Explanation:** Choose indices (1, 2, 2) and A becomes [3,1,0,2].

**Example 3:**

**Input:** A = [2,-3,-1,5,-4], K = 2

**Output:** 13

**Explanation:** Choose indices (1, 4) and A becomes [2,3,-1,5,4].

**Note:**

1. 1 <= A.length <= 10000
2. 1 <= K <= 10000
3. -100 <= A[i] <= 100

**224. 1009. Complement of Base 10 Integer**

Every non-negative integer N has a binary representation.  For **Example**, 5 can be represented as "101" in binary, 11 as "1011" in binary, and so on.  Note that except for N = 0, there are no leading zeroes in any binary representation.

The *complement* of a binary representation is the number in binary you get when changing every 1 to a 0 and 0 to a 1.  For **Example**, the complement of "101" in binary is "010" in binary.

For a given number N in base-10, return the complement of it's binary representation as a base-10 integer.

**Example 1:**

**Input:** 5

**Output:** 2

**Explanation:** 5 is "101" in binary, with complement "010" in binary, which is 2 in base-10.

**Example 2:**

**Input:** 7

**Output:** 0

**Explanation:** 7 is "111" in binary, with complement "000" in binary, which is 0 in base-10.

**Example 3:**

**Input:** 10

**Output:** 5

**Explanation:** 10 is "1010" in binary, with complement "0101" in binary, which is 5 in base-10.

**Note:**

1. 0 <= N < 10^9
2. This question is the same as 476: <https://leetcode.com/problems/number-complement/>

**225. 1010. Pairs of Songs With Total Durations Divisible by 60**

In a list of songs, the i-th song has a duration of time[i] seconds.

Return the number of pairs of songs for which their total duration in seconds is divisible by 60.  Formally, we want the number of indices i, j such that i < j with (time[i] + time[j]) % 60 == 0.

**Example 1:**

**Input:** [30,20,150,100,40]

**Output:** 3

**Explanation:** Three pairs have a total duration divisible by 60:

(time[0] = 30, time[2] = 150): total duration 180

(time[1] = 20, time[3] = 100): total duration 120

(time[1] = 20, time[4] = 40): total duration 60

**Example 2:**

**Input:** [60,60,60]

**Output:** 3

**Explanation:** All three pairs have a total duration of 120, which is divisible by 60.

**Note:**

* 1 <= time.length <= 60000
* 1 <= time[i] <= 500

**226. 1013. Partition Array Into Three Parts With Equal Sum**

Given an array A of integers, return true if and only if we can partition the array into three **non-empty** parts with equal sums.

Formally, we can partition the array if we can find indexes i+1 < j with (A[0] + A[1] + ... + A[i] == A[i+1] + A[i+2] + ... + A[j-1] == A[j] + A[j-1] + ... + A[A.length - 1])

**Example 1:**

**Input:** A = [0,2,1,-6,6,-7,9,1,2,0,1]

**Output:** true

**Explanation:** 0 + 2 + 1 = -6 + 6 - 7 + 9 + 1 = 2 + 0 + 1

**Example 2:**

**Input:** A = [0,2,1,-6,6,7,9,-1,2,0,1]

**Output:** false

**Example 3:**

**Input:** A = [3,3,6,5,-2,2,5,1,-9,4]

**Output:** true

**Explanation:** 3 + 3 = 6 = 5 - 2 + 2 + 5 + 1 - 9 + 4

**Constraints:**

* 3 <= A.length <= 50000
* -10^4 <= A[i] <= 10^4

**227. 1018. Binary Prefix Divisible By 5**

Given an array A of 0s and 1s, consider N\_i: the i-th subarray from A[0] to A[i] interpreted as a binary number (from most-significant-bit to least-significant-bit.)

Return a list of booleans answer, where answer[i] is true if and only if N\_i is divisible by 5.

**Example 1:**

**Input:** [0,1,1]

**Output:** [true,false,false]

**Explanation:**

The **Input** numbers in binary are 0, 01, 011; which are 0, 1, and 3 in base-10. Only the first number is divisible by 5, so answer[0] is true.

**Example 2:**

**Input:** [1,1,1]

**Output:** [false,false,false]

**Example 3:**

**Input:** [0,1,1,1,1,1]

**Output:** [true,false,false,false,true,false]

**Example 4:**

**Input:** [1,1,1,0,1]

**Output:** [false,false,false,false,false]

**Note:**

1. 1 <= A.length <= 30000
2. A[i] is 0 or 1

**228. 1025. Divisor Game**

Alice and Bob take turns playing a game, with Alice starting first.

Initially, there is a number N on the chalkboard.  On each player's turn, that player makes a *move* consisting of:

* Choosing any x with 0 < x < N and N % x == 0.
* Replacing the number N on the chalkboard with N - x.

Also, if a player cannot make a move, they lose the game.

Return True if and only if Alice wins the game, assuming both players play optimally.

**Example 1:**

**Input:** 2

**Output:** true

**Explanation:** Alice chooses 1, and Bob has no more moves.

**Example 2:**

**Input:** 3

**Output:** false

**Explanation:** Alice chooses 1, Bob chooses 1, and Alice has no more moves.

**Note:**

1. 1 <= N <= 1000

**229. 1030. Matrix Cells in Distance Order**

We are given a matrix with R rows and C columns has cells with integer coordinates (r, c), where 0 <= r < R and 0 <= c < C.

Additionally, we are given a cell in that matrix with coordinates (r0, c0).

Return the coordinates of all cells in the matrix, sorted by their distance from (r0, c0) from smallest distance to largest distance.  Here, the distance between two cells (r1, c1) and (r2, c2) is the Manhattan distance, |r1 - r2| + |c1 - c2|.  (You may return the answer in any order that satisfies this condition.)

**Example 1:**

**Input:** R = 1, C = 2, r0 = 0, c0 = 0

**Output:** [[0,0],[0,1]]

**Explanation:** The distances from (r0, c0) to other cells are: [0,1]

**Example 2:**

**Input:** R = 2, C = 2, r0 = 0, c0 = 1

**Output:** [[0,1],[0,0],[1,1],[1,0]]

**Explanation:** The distances from (r0, c0) to other cells are: [0,1,1,2]

The answer [[0,1],[1,1],[0,0],[1,0]] would also be accepted as correct.

**Example 3:**

**Input:** R = 2, C = 3, r0 = 1, c0 = 2

**Output:** [[1,2],[0,2],[1,1],[0,1],[1,0],[0,0]]

**Explanation:** The distances from (r0, c0) to other cells are: [0,1,1,2,2,3]

There are other answers that would also be accepted as correct, such as [[1,2],[1,1],[0,2],[1,0],[0,1],[0,0]].

**Note:**

1. 1 <= R <= 100
2. 1 <= C <= 100
3. 0 <= r0 < R
4. 0 <= c0 < C

**230. 1033. Moving Stones Until Consecutive**

Three stones are on a number line at positions a, b, and c.

Each turn, you pick up a stone at an endpoint (ie., either the lowest or highest position stone), and move it to an unoccupied position between those endpoints.  Formally, let's say the stones are currently at positions x, y, z with x < y < z.  You pick up the stone at either position x or position z, and move that stone to an integer position k, with x < k < z and k != y.

The game ends when you cannot make any more moves, ie. the stones are in consecutive positions.

When the game ends, what is the minimum and maximum number of moves that you could have made?  Return the answer as an length 2 array: answer = [minimum\_moves, maximum\_moves]

**Example 1:**

**Input:** a = 1, b = 2, c = 5

**Output:** [1,2]

**Explanation:** Move the stone from 5 to 3, or move the stone from 5 to 4 to 3.

**Example 2:**

**Input:** a = 4, b = 3, c = 2

**Output:** [0,0]

**Explanation:** We cannot make any moves.

**Example 3:**

**Input:** a = 3, b = 5, c = 1

**Output:** [1,2]

**Explanation:** Move the stone from 1 to 4; or move the stone from 1 to 2 to 4.

**Note:**

1. 1 <= a <= 100
2. 1 <= b <= 100
3. 1 <= c <= 100
4. a != b, b != c, c != a

**231. 1037. Valid Boomerang**

A *boomerang* is a set of 3 points that are all distinct and **not** in a straight line.

Given a list of three points in the plane, return whether these points are a boomerang.

**Example 1:**

**Input:** [[1,1],[2,3],[3,2]]

**Output:** true

**Example 2:**

**Input:** [[1,1],[2,2],[3,3]]

**Output:** false

**Note:**

1. points.length == 3
2. points[i].length == 2
3. 0 <= points[i][j] <= 100

**232. 1047. Remove All Adjacent Duplicates In String**

Given a string S of lowercase letters, a *duplicate removal* consists of choosing two adjacent and equal letters, and removing them.

We repeatedly make duplicate removals on S until we no longer can.

Return the final string after all such duplicate removals have been made.  It is guaranteed the answer is unique.

**Example 1:**

**Input:** "abbaca"

**Output:** "ca"

**Explanation:**

For **Example**, in "abbaca" we could remove "bb" since the letters are adjacent and equal, and this is the only possible move.  The result of this move is that the string is "aaca", of which only "aa" is possible, so the final string is "ca".

**Note:**

1. 1 <= S.length <= 20000
2. S consists only of English lowercase letters.

**233. 1051. Height Checker**

Students are asked to stand in non-decreasing order of heights for an annual photo.

Return the minimum number of students that must move in order for all students to be standing in non-decreasing order of height.

Notice that when a group of students is selected they can reorder in any possible way between themselves and the non selected students remain on their seats.

**Example 1:**

**Input:** heights = [1,1,4,2,1,3]

**Output:** 3

**Explanation:**

Current array : [1,1,4,2,1,3]

Target array : [1,1,1,2,3,4]

On index 2 (0-based) we have 4 vs 1 so we have to move this student.

On index 4 (0-based) we have 1 vs 3 so we have to move this student.

On index 5 (0-based) we have 3 vs 4 so we have to move this student.

**Example 2:**

**Input:** heights = [5,1,2,3,4]

**Output:** 5

**Example 3:**

**Input:** heights = [1,2,3,4,5]

**Output:** 0

**Constraints:**

* 1 <= heights.length <= 100
* 1 <= heights[i] <= 100

**234. 1078. Occurrences After Bigram**

Given words first and second, consider occurrences in some text of the form "first second third", where second comes immediately after first, and third comes immediately after second.

For each such occurrence, add "third" to the answer, and return the answer.

**Example 1:**

**Input:** text = "alice is a good girl she is a good student", first = "a", second = "good"

**Output:** ["girl","student"]

**Example 2:**

**Input:** text = "we will we will rock you", first = "we", second = "will"

**Output:** ["we","rock"]

**Note:**

1. 1 <= text.length <= 1000
2. text consists of space separated words, where each word consists of lowercase English letters.
3. 1 <= first.length, second.length <= 10
4. first and second consist of lowercase English letters.

**235. 1103. Distribute Candies to People**

We distribute some number of candies, to a row of **n = num\_people** people in the following way:

We then give 1 candy to the first person, 2 candies to the second person, and so on until we give n candies to the last person.

Then, we go back to the start of the row, giving n + 1 candies to the first person, n + 2 candies to the second person, and so on until we give 2 \* n candies to the last person.

This process repeats (with us giving one more candy each time, and moving to the start of the row after we reach the end) until we run out of candies.  The last person will receive all of our remaining candies (not necessarily one more than the previous gift).

Return an array (of length num\_people and sum candies) that represents the final distribution of candies.

**Example 1:**

**Input:** candies = 7, num\_people = 4

**Output:** [1,2,3,1]

**Explanation:**

On the first turn, ans[0] += 1, and the array is [1,0,0,0].

On the second turn, ans[1] += 2, and the array is [1,2,0,0].

On the third turn, ans[2] += 3, and the array is [1,2,3,0].

On the fourth turn, ans[3] += 1 (because there is only one candy left), and the final array is [1,2,3,1].

**Example 2:**

**Input:** candies = 10, num\_people = 3

**Output:** [5,2,3]

**Explanation:**

On the first turn, ans[0] += 1, and the array is [1,0,0].

On the second turn, ans[1] += 2, and the array is [1,2,0].

On the third turn, ans[2] += 3, and the array is [1,2,3].

On the fourth turn, ans[0] += 4, and the final array is [5,2,3].

**Constraints:**

* 1 <= candies <= 10^9
* 1 <= num\_people <= 1000

**236. 1108. Defanging an IP Address**

Given a valid (IPv4) IP address, return a defanged version of that IP address.

A *defanged IP address* replaces every period "." with "[.]".

**Example 1:**

**Input:** address = "1.1.1.1"

**Output:** "1[.]1[.]1[.]1"

**Example 2:**

**Input:** address = "255.100.50.0"

**Output:** "255[.]100[.]50[.]0"

**Constraints:**

* The given address is a valid IPv4 address.

**237. 1122. Relative Sort Array**

Given two arrays arr1 and arr2, the elements of arr2 are distinct, and all elements in arr2 are also in arr1.

Sort the elements of arr1 such that the relative ordering of items in arr1 are the same as in arr2.  Elements that don't appear in arr2 should be placed at the end of arr1 in **ascending** order.

**Example 1:**

**Input:** arr1 = [2,3,1,3,2,4,6,7,9,2,19], arr2 = [2,1,4,3,9,6]

**Output:** [2,2,2,1,4,3,3,9,6,7,19]

**Constraints:**

* arr1.length, arr2.length <= 1000
* 0 <= arr1[i], arr2[i] <= 1000
* Each arr2[i] is distinct.
* Each arr2[i] is in arr1.

**238. 1137. N-th Tribonacci Number**

The Tribonacci sequence Tn is defined as follows:

T0 = 0, T1 = 1, T2 = 1, and Tn+3 = Tn + Tn+1 + Tn+2 for n >= 0.

Given n, return the value of Tn.

**Example 1:**

**Input:** n = 4

**Output:** 4

**Explanation:**

T\_3 = 0 + 1 + 1 = 2

T\_4 = 1 + 1 + 2 = 4

**Example 2:**

**Input:** n = 25

**Output:** 1389537

**Constraints:**

* 0 <= n <= 37
* The answer is guaranteed to fit within a 32-bit integer, ie. answer <= 2^31 - 1.

239. [**1556. Thousand Separator**](https://leetcode-cn.com/problems/thousand-separator/)

Given an integer , add a dot (".") as the thousands separator and return it in string format.

**Example1:**

**Input: n = 987**

**Output: "987"**

**Example2:**

**Input: n = 1234**

**Output: "1.234"**

**Example3:**

**Input: n = 123456789**

**Output: "123.456.789"**

**Example4:**

**Input: n = 0**

**Output: "0"**

**Constraints:**

* 0 <= n < 2^31

240. [**1560. Most Visited Sector in a Circular Track**](https://leetcode-cn.com/problems/most-visited-sector-in-a-circular-track/)

Given an integer and an integer array . We have a circular track which consists of sectors labeled from 1 to . A marathon will be held on this track, the marathon consists of rounds. The ith round starts at sector and ends at sector. For **Example**, round 1 starts at sector and ends at sector

Return an array of the most visited sectors sorted in ascending order.

Notice that you circulate the track in ascending order of sector numbers in the counter-clockwise direction (See the first **Example**).

**Example 1:**

**Input: n = 4, rounds = [1,3,1,2]**

**Output: [1,2]**

**Explanation: The marathon starts at sector 1. The order of the visited sectors is as follows:**

**1 --> 2 --> 3 (end of round 1) --> 4 --> 1 (end of round 2) --> 2 (end of round 3 and the marathon)**

**We can see that both sectors 1 and 2 are visited twice and they are the most visited sectors. Sectors 3 and 4 are visited only once.**

**Example 2:**

**Input: n = 2, rounds = [2,1,2,1,2,1,2,1,2]**

**Output: [2]**

**Example 3:**

**Input: n = 7, rounds = [1,3,5,7]**

**Output: [1,2,3,4,5,6,7]**

**Constraints:**

* 2 <= n <= 100
* 1 <= m <= 100
* rounds.length == m + 1
* 1 <= rounds[i] <= n
* rounds[i] != rounds[i + 1] for 0 <= i < m

241. [**1572. Matrix Diagonal Sum**](https://leetcode-cn.com/problems/matrix-diagonal-sum/)

Given a square matrix , return the sum of the matrix diagonals.

Only include the sum of all the elements on the primary diagonal and all the elements on the secondary diagonal that are not part of the primary diagonal.

**Example 1:**

**Input: mat = [[1,2,3],**

**[4,5,6],**

**[7,8,9]]**

**Output: 25**

**Explanation: Diagonals sum: 1 + 5 + 9 + 3 + 7 = 25**

**Notice that element mat[1][1] = 5 is counted only once.**

**Example 2:**

**Input: mat = [[1,1,1,1],**

**[1,1,1,1],**

**[1,1,1,1],**

**[1,1,1,1]]**

**Output: 8**

**Example 3:**

**Input: mat = [[5]]**

**Output: 5**

**Constraints:**

* n == mat.length == mat[i].length
* 1 <= n <= 100
* 1 <= mat[i][j] <= 100

242. [**1573. Number of Ways to Split a String**](https://leetcode-cn.com/problems/number-of-ways-to-split-a-string/)

Given a binary string (a string consisting only of '0's and '1's), we can split s into 3 non-empty strings .

Return the number of ways s can be split such that the number of characters '1' is the same in and .

Since the answer may be too large, return it modulo .

**Example 1:**

**Input: s = "10101"**

**Output: 4**

**Explanation: There are four ways to split s in 3 parts where each part contains the same number of letters '1'.**

**"1|010|1"**

**"1|01|01"**

**"10|10|1"**

**"10|1|01"**

**Example 2:**

**Input: s = "1001"**

**Output: 0**

**Example 3:**

**Input: s = "0000"**

**Output: 3**

**Explanation: There are three ways to split s in 3 parts.**

**"0|0|00"**

**"0|00|0"**

**"00|0|0"**

**Example 4:**

**Input: s = "100100010100110"**

**Output: 12**

**Constraints:**

* 3 <= s.length <= 10^5
* s[i] is '0' or '1'.

243. [**1576. Replace All ?'s to Avoid Consecutive Repeating Characters**](https://leetcode-cn.com/problems/replace-all-s-to-avoid-consecutive-repeating-characters/)

Given a string  containing only lower case English letters and the '?' character, convert **all** the '?' characters into lower case letters such that the final string does not contain any **consecutive repeating** characters. You **cannot** modify the non '?' characters.

It is **guaranteed** that there are no consecutive repeating characters in the given string except for '?'.

Return the final string after all the conversions (possibly zero) have been made. If there is more than one solution, return any of them. It can be shown that an answer is always possible with the given constraints.

**Example 1:**

**Input: s = "?zs"**

**Output: "azs"**

**Explanation: There are 25 solutions for this problem. From "azs" to "yzs", all are valid. Only "z" is an invalid modification as the string will consist of consecutive repeating characters in "zzs".**

**Example 2:**

**Input: s = "ubv?w"**

**Output: "ubvaw"**

**Explanation: There are 24 solutions for this problem. Only "v" and "w" are invalid modifications as the strings will consist of consecutive repeating characters in "ubvvw" and "ubvww".**

**Example 3:**

**Input: s = "j?qg??b"**

**Output: "jaqgacb"**

**Example 4：**

**Input: s = "??yw?ipkj?"**

**Output: "acywaipkja"**

**Constraints:**

* 1 <= s.length <= 100
* s contains only lower case English letters and '?'.

244. [**1578. Minimum Deletion Cost to Avoid Repeating Letters**](https://leetcode-cn.com/problems/minimum-deletion-cost-to-avoid-repeating-letters/)

Given a string and an array of integers cost where is the cost of deleting the character in .

Return the minimum cost of deletions such that there are no two identical letters next to each other.

Notice that you will delete the chosen characters at the same time, in other words, after deleting a character, the costs of deleting other characters will not change.

**Example 1:**

**Input: s = "abaac", cost = [1,2,3,4,5]**

**Output: 3**

**Explanation: Delete the letter "a" with cost 3 to get "abac" (String without two identical letters next to each other).**

**Example 2:**

**Input: s = "abc", cost = [1,2,3]**

**Output: 0**

**Explanation: You don't need to delete any character because there are no identical letters next to each other.**

**Example 3:**

**Input: s = "aabaa", cost = [1,2,3,4,1]**

**Output: 2**

**Explanation: Delete the first and the last character, getting the string ("aba").**

**Constraints:**

* s.length == cost.length
* 1 <= s.length, cost.length <= 10^5
* 1 <= cost[i] <= 10^4
* s contains only lowercase English letters.

245. [**1582. Special Positions in a Binary Matrix**](https://leetcode-cn.com/problems/special-positions-in-a-binary-matrix/)

Given a  matrix , where is either 0 or 1, return the number of special positions in .

A position is called special if  and all other elements in row  and column  are 0 (rows and columns are 0-indexed).

**Example 1:**

**Input: mat = [[1,0,0],**

**[0,0,1],**

**[1,0,0]]**

**Output: 1**

**Explanation: (1,2) is a special position because mat[1][2] == 1 and all other elements in row 1 and column 2 are 0.**

**Example 2:**

**Input: mat = [[1,0,0],**

**[0,1,0],**

**[0,0,1]]**

**Output: 3**

**Explanation: (0,0), (1,1) and (2,2) are special positions.**

**Example 3:**

**Input: mat = [[0,0,0,1],**

**[1,0,0,0],**

**[0,1,1,0],**

**[0,0,0,0]]**

**Output: 2**

**Example 4:**

**Input: mat = [[0,0,0,0,0],**

**[1,0,0,0,0],**

**[0,1,0,0,0],**

**[0,0,1,0,0],**

**[0,0,0,1,1]]**

**Output: 3**

**Constraints:**

* rows == mat.length
* cols == mat[i].length
* 1 <= rows, cols <= 100
* mat[i][j] is 0 or 1.

246. [**1592. Rearrange Spaces Between Words**](https://leetcode-cn.com/problems/rearrange-spaces-between-words/)

You are given a string of words that are placed among some number of spaces. Each word consists of one or more lowercase English letters and are separated by at least one space. It's guaranteed that text **contains at least one word**.

Rearrange the spaces so that there is an **equal** number of spaces between every pair of adjacent words and that number is **maximized**. If you cannot redistribute all the spaces equally, place the **extra spaces at the end**, meaning the returned string should be the same length as text.

Return the string after rearranging the spaces.

**Example 1:**

**Input: text = " this is a sentence "**

**Output: "this is a sentence"**

**Explanation: There are a total of 9 spaces and 4 words. We can evenly divide the 9 spaces between the words: 9 / (4-1) = 3 spaces.**

**Example 2:**

**Input: text = " practice makes perfect"**

**Output: "practice makes perfect "**

**Explanation: There are a total of 7 spaces and 3 words. 7 / (3-1) = 3 spaces plus 1 extra space. We place this extra space at the end of the string.**

**Example 3:**

**Input: text = "hello world"**

**Output: "hello world"**

**Example 4:**

**Input: text = " walks udp package into bar a"**

**Output: "walks udp package into bar a "**

**Example 5:**

**Input: text = "a"**

**Output: "a"**

**Constraints:**

* 1 <= text.length <= 100
* text consists of lowercase English letters and ' '.
* text contains at least one word.

247. [**1598. Crawler Log Folder**](https://leetcode-cn.com/problems/crawler-log-folder/)

The Leetcode file system keeps a log each time some user performs a change folder operation.

The operations are described below:

"../" : Move to the parent folder of the current folder. (If you are already in the main folder, remain in the same folder).

"./" : Remain in the same folder.

"x/" : Move to the child folder named x (This folder is guaranteed to always exist).

You are given a list of strings where is the operation performed by the user at the step.

The file system starts in the main folder, then the operations in logs are performed.

Return the minimum number of operations needed to go back to the main folder after the change folder operations.

**Example 1:**

**Input: logs = ["d1/","d2/","../","d21/","./"]**

**Output: 2**

**Explanation: Use this change folder operation "../" 2 times and go back to the main folder.**

**Example 2:**

**Input: logs = ["d1/","d2/","./","d3/","../","d31/"]**

**Output: 3**

**Example 3:**

**Input: logs = ["d1/","../","../","../"]**

**Output: 0**

**Constraints:**

* 1 <= logs.length <= 1000
* 2 <= logs[i].length <= 10
* logs[i] contains lowercase English letters, digits, '.', and '/'.
* logs[i] follows the format described in the statement.
* Folder names consist of lowercase English letters and digits.

248. [**1603. Design Parking System**](https://leetcode-cn.com/problems/design-parking-system/)

Design a parking system for a parking lot. The parking lot has three kinds of parking spaces: big, medium, and small, with a fixed number of slots for each size.

Implement the ParkingSystem class:

* ParkingSystem(int big, int medium, int small) Initializes object of the ParkingSystem class. The number of slots for each parking space are given as part of the constructor.
* bool addCar(int carType) Checks whether there is a parking space of carType for the car that wants to get into the parking lot. carType can be of three kinds: big, medium, or small, which are represented by 1, 2, and 3 respectively. A car can only park in a parking space of its carType. If there is no space available, return false, else park the car in that size space and return true.

**Example 1:**

**Input**

**["ParkingSystem", "addCar", "addCar", "addCar", "addCar"]**

**[[1, 1, 0], [1], [2], [3], [1]]**

**Output**

**[null, true, true, false, false]**

**Explanation**

**ParkingSystem parkingSystem = new ParkingSystem(1, 1, 0);**

**parkingSystem.addCar(1); // return true because there is 1 available slot for a big car**

**parkingSystem.addCar(2); // return true because there is 1 available slot for a medium car**

**parkingSystem.addCar(3); // return false because there is no available slot for a small car**

**parkingSystem.addCar(1); // return false because there is no available slot for a big car. It is already occupied.**

**Constraints:**

* 0 <= big, medium, small <= 1000
* carType is 1, 2, or 3
* At most 1000 calls will be made to addCar

249. [**1608. Special Array With X Elements Greater Than or Equal X**](https://leetcode-cn.com/problems/special-array-with-x-elements-greater-than-or-equal-x/)

You are given an array of non-negative integers. is considered special if there exists a number such that there are exactly numbers in that are greater than or equal to .

Notice that does not have to be an element in .

Return if the array is special, otherwise, return -1. It can be proven that if is special, the value for is unique.

**Example 1:**

**Input: nums = [3,5]**

**Output: 2**

**Explanation: There are 2 values (3 and 5) that are greater than or equal to 2.**

**Example 2:**

**Input: nums = [0,0]**

**Output: -1**

**Explanation: No numbers fit the criteria for x.**

**If x = 0, there should be 0 numbers >= x, but there are 2.**

**If x = 1, there should be 1 number >= x, but there are 0.**

**If x = 2, there should be 2 numbers >= x, but there are 0.**

**x cannot be greater since there are only 2 numbers in nums.**

**Example 3:**

**Input: nums = [0,4,3,0,4]**

**Output: 3**

**Explanation: There are 3 values that are greater than or equal to 3.**

**Example 4:**

**Input: nums = [3,6,7,7,0]**

**Output: -1**

**Constraints:**

* 1 <= nums.length <= 100
* 0 <= nums[i] <= 1000

250. [**1609. Even Odd Tree**](https://leetcode-cn.com/problems/even-odd-tree/)

A binary tree is named **Even-Odd** if it meets the following conditions:

* The root of the binary tree is at level index 0, its children are at level index 1, their children are at level index 2, etc.
* For every **even-indexed** level, all nodes at the level have **odd** integer values in **strictly increasing** order (from left to right).
* For every **odd-indexed** level, all nodes at the level have **even** integer values in **strictly decreasing** order (from left to right).
* Given theof a binary tree, return true if the binary tree is **Even-Odd**, otherwise return false.

**Example 1:**

**Input: root = [1,10,4,3,null,7,9,12,8,6,null,null,2]**

**Output: true**

**Explanation: The node values on each level are:**

**Level 0: [1]**

**Level 1: [10,4]**

**Level 2: [3,7,9]**

**Level 3: [12,8,6,2]**

**Since levels 0 and 2 are all odd and increasing, and levels 1 and 3 are all even and decreasing, the tree is Even-Odd.**

**Example 2:**

**Input: root = [5,4,2,3,3,7]**

**Output: false**

**Explanation: The node values on each level are:**

**Level 0: [5]**

**Level 1: [4,2]**

**Level 2: [3,3,7]**

**Node values in the level 2 must be in strictly increasing order, so the tree is not Even-Odd.**

**Example 3:**

**Input: root = [5,9,1,3,5,7]**

**Output: false**

**Explanation: Node values in the level 1 should be even integers.**

**Example 4:**

**Input: root = [1]**

**Output: true**

**Example 5:**

**Input: root = [11,8,6,1,3,9,11,30,20,18,16,12,10,4,2,17]**

**Output: true**

**Constraints:**

* The number of nodes in the tree is in the range [1, 105].
* 1 <= Node.val <=

251. [**1614. Maximum Nesting Depth of the Parentheses**](https://leetcode-cn.com/problems/maximum-nesting-depth-of-the-parentheses/)

A string is a **valid parentheses string** (denoted **VPS**) if it meets one of the following:

* It is an empty string "", or a single character not equal to "(" or ")",
* It can be written as AB (A concatenated with B), where A and B are **VPS**'s, or
* It can be written as (A), where A is a **VPS**.

We can similarly define the nesting depth of any **VPS** as follows:

* , where and are **VPS**'s
* , where is a **VPS**.

For **Example**, "", "()()", and "()(()())" are **VPS**'s (with nesting depths 0, 1, and 2), and ")(" and "(()" are not **VPS**'s.

Given a **VPS** represented as string, return the nesting depth of .

**Example 1:**

**Input: s = "(1+(2\*3)+((8)/4))+1"**

**Output: 3**

**Explanation: Digit 8 is inside of 3 nested parentheses in the string.**

**Example 2:**

**Input: s = "(1)+((2))+(((3)))"**

**Output: 3**

**Example 3:**

**Input: s = "1+(2\*3)/(2-1)"**

**Output: 1**

**Example 4:**

**Input: s = "1"**

**Output: 0**

**Constraints:**

* 1 <= s.length <= 100
* s consists of digits 0-9 and characters '+', '-', '\*', '/', '(', and ')'.
* It is guaranteed that parentheses expression s is a VPS.

252. [**1615. Maximal Network Rank**](https://leetcode-cn.com/problems/maximal-network-rank/)

There is an infrastructure of cities with some number of connecting these cities. Each indicates that there is a bidirectional road between cities and .

The **network rank** of **two different cities** is defined as the total number of **directly** connected roads to **either** city. If a road is directly connected to both cities, it is only counted once.

The **maximal network rank** of the infrastructure is the **maximum network rank** of all pairs of different cities.

Given the integer n and the array roads, return the **maximal network rank** of the entire infrastructure.

**Example 1:**

**Input: n = 4, roads = [[0,1],[0,3],[1,2],[1,3]]**

**Output: 4**

**Explanation: The network rank of cities 0 and 1 is 4 as there are 4 roads that are connected to either 0 or 1. The road between 0 and 1 is only counted once.**

**Example 2:**

**Input: n = 5, roads = [[0,1],[0,3],[1,2],[1,3],[2,3],[2,4]]**

**Output: 5**

**Explanation: There are 5 roads that are connected to cities 1 or 2.**

**Example 3:**

**Input: n = 8, roads = [[0,1],[1,2],[2,3],[2,4],[5,6],[5,7]]**

**Output: 5**

**Explanation: The network rank of 2 and 5 is 5. Notice that all the cities do not have to be connected.**

**Constraints:**

* 2 <= n <= 100
* 0 <= roads.length <= n \* (n - 1) / 2
* roads[i].length == 2
* 0 <= ai, bi <= n-1
* ai != bi
* Each pair of cities has at most one road connecting them.

253. [**5122. Mean of Array After Removing Some Elements**](https://leetcode-cn.com/problems/mean-of-array-after-removing-some-elements/)

Given an integer array , return the mean of the remaining integers after removing the smallest and the largest of the elements.

Answers withinof the actual answer will be considered accepted.

**Example 1:**

**Input: arr = [1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3]**

**Output: 2.00000**

**Explanation: After erasing the minimum and the maximum values of this array, all elements are equal to 2, so the mean is 2.**

**Example 2:**

**Input: arr = [6,2,7,5,1,2,0,3,10,2,5,0,5,5,0,8,7,6,8,0]**

**Output: 4.00000**

**Example 3:**

**Input: arr = [6,0,7,0,7,5,7,8,3,4,0,7,8,1,6,8,1,1,2,4,8,1,9,5,4,3,8,5,10,8,6,6,1,0,6,10,8,2,3,4]**

**Output: 4.77778**

**Example 4:**

**Input: arr = [9,7,8,7,7,8,4,4,6,8,8,7,6,8,8,9,2,6,0,0,1,10,8,6,3,3,5,1,10,9,0,7,10,0,10,4,1,10,6,9,3,6,0,0,2,7,0,6,7,2,9,7,7,3,0,1,6,1,10,3]**

**Output: 5.27778**

**Example 5:**

**Input: arr = [4,8,4,10,0,7,1,3,7,8,8,3,4,1,6,2,1,1,8,0,9,8,0,3,9,10,3,10,1,10,7,3,2,1,4,9,10,7,6,4,0,8,5,1,2,1,6,2,5,0,7,10,9,10,3,7,10,5,8,5,7,6,7,6,10,9,5,10,5,5,7,2,10,7,7,8,2,0,1,1]**

**Output: 5.29167**

**Constraints:**

* 20 <= arr.length <= 1000
* arr.length is a multiple of 20.
* 0 <= arr[i] <=

254. [**5543. Largest Substring Between Two Equal Characters**](https://leetcode-cn.com/problems/largest-substring-between-two-equal-characters/)

Given a string , return the length of the longest substring between two equal characters, excluding the two characters. If there is no such substring return -1.

A substring is a contiguous sequence of characters within a string.

**Example 1:**

**Input: s = "aa"**

**Output: 0**

**Explanation: The optimal substring here is an empty substring between the two 'a's.**

**Example 2:**

**Input: s = "abca"**

**Output: 2**

**Explanation: The optimal substring here is "bc".**

**Example 3:**

**Input: s = "cbzxy"**

**Output: -1**

**Explanation: There are no characters that appear twice in s.**

**Example 4:**

**Input: s = "cabbac"**

**Output: 4**

**Explanation: The optimal substring here is "abba". Other non-optimal substrings include "bb" and "".**

**Constraints:**

* 1 <= s.length <= 300
* s contains only lowercase English letters.

255. [**5544. Lexicographically Smallest String After Applying Operations**](https://leetcode-cn.com/problems/lexicographically-smallest-string-after-applying-operations/)

You are given a stringof even length consisting of digits from 0 to 9, and two integers and .

You can apply either of the following two operations any number of times and in any order on :

* Add to all odd indices of **(0-indexed)**. Digits post 9 are cycled back to 0. For **Example**, if and , becomes.
* Rotate to the right by positions. For **Example**, if and , becomes .

Return the lexicographically smallest string you can obtain by applying the above operations any number of times on .

A string is **lexicographically smaller** than a string (of the same length) if in the first position where and differ, string has a letter that appears earlier in the alphabet than the corresponding letter in . For **Example**, "0158" is lexicographically smaller than "0190" because the first position they differ is at the third letter, and '5' comes before '9'.

**Example 1:**

**Input: s = "5525", a = 9, b = 2**

**Output: "2050"**

**Explanation: We can apply the following operations:**

**Start: "5525"**

**Rotate: "2555"**

**Add: "2454"**

**Add: "2353"**

**Rotate: "5323"**

**Add: "5222"**

**​​​​​​​Add: "5121"**

**​​​​​​​Rotate: "2151"**

**​​​​​​​Add: "2050"​​​​​​​​​​​​**

**There is no way to obtain a string that is lexicographically smaller then "2050".**

**Example 2:**

**Input: s = "74", a = 5, b = 1**

**Output: "24"**

**Explanation: We can apply the following operations:**

**Start: "74"**

**Rotate: "47"**

**​​​​​​​Add: "42"**

**​​​​​​​Rotate: "24"​​​​​​​​​​​​**

**There is no way to obtain a string that is lexicographically smaller then "24".**

**Example 3:**

**Input: s = "0011", a = 4, b = 2**

**Output: "0011"**

**Explanation: There are no sequence of operations that will give us a lexicographically smaller string than "0011".**

**Example 4:**

**Input: s = "43987654", a = 7, b = 3**

**Output: "00553311"**

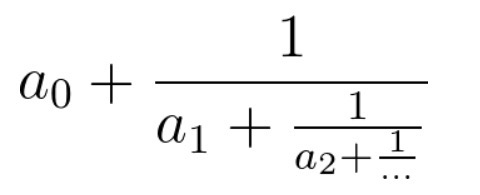
**Constraints:**

* 2 <= s.length <= 100
* s.length is even.
* s consists of digits from 0 to 9 only.
* 1 <= a <= 9
* 1 <= b <= s.length – 1

**以下部分无英文题面**

256. [**LCP 02. 分式化简**](https://leetcode-cn.com/problems/deep-dark-fraction/)

有一个同学在学习分式。他需要将一个连分数化成最简分数，你能帮助他吗？



连分数是形如上图的分式。在本题中，所有系数都是大于等于0的整数。

输入的代表连分数的系数（代表上图的，以此类推）。返回一个长度为2的数组，使得连分数的值等于，且最大公约数为1。

**示例 1：**

**输入：cont = [3, 2, 0, 2]**

**输出：[13, 4]**

**解释：原连分数等价于3 + (1 / (2 + (1 / (0 + 1 / 2))))。注意[26, 8], [-13, -4]都不是正确答案。**

**示例 2：**

**输入：cont = [0, 0, 3]**

**输出：[3, 1]**

**解释：如果答案是整数，令分母为1即可。**

**限制：**

* cont[i] >= 0
* 1 <= cont的长度 <= 10
* cont最后一个元素不等于0
* 答案的n, m的取值都能被32位int整型存下（即不超过2 ^ 31 - 1）。

257. [**LCP 06. 拿硬币**](https://leetcode-cn.com/problems/na-ying-bi/)

桌上有 堆力扣币，每堆的数量保存在数组 中。我们每次可以选择任意一堆，拿走其中的一枚或者两枚，求拿完所有力扣币的最少次数。

**示例 1：**

**输入：[4,2,1]**

**输出：4**

**解释：第一堆力扣币最少需要拿 2 次，第二堆最少需要拿 1 次，第三堆最少需要拿 1 次，总共 4 次即可拿完。**

**示例 2：**

**输入：[2,3,10]**

**输出：8**

**限制：**

* 1 <= n <= 4
* 1 <= coins[i] <= 10

**258.** [**LCP 07. 传递信息**](https://leetcode-cn.com/problems/chuan-di-xin-xi/)

小朋友 A 在和 ta 的小伙伴们玩传信息游戏，游戏规则如下：

有 名玩家，所有玩家编号分别为，其中小朋友 A 的编号为 0

每个玩家都有固定的若干个可传信息的其他玩家（也可能没有）。传信息的关系是单向的（比如 A 可以向 B 传信息，但 B 不能向 A 传信息）。

每轮信息必须需要传递给另一个人，且信息可重复经过同一个人

给定总玩家数 ，以及按 [玩家编号,对应可传递玩家编号] 关系组成的二维数组 。返回信息从小 A (编号 0 ) 经过 轮传递到编号为 的小伙伴处的方案数；若不能到达，返回 0。

**示例 1：**

**输入：n = 5, relation = [[0,2],[2,1],[3,4],[2,3],[1,4],[2,0],[0,4]], k = 3**

**输出：3**

**解释：信息从小 A 编号 0 处开始，经 3 轮传递，到达编号 4。共有 3 种方案，分别是 0->2->0->4， 0->2->1->4， 0->2->3->4。**

**示例 2：**

**输入：n = 3, relation = [[0,2],[2,1]], k = 2**

**输出：0**

**解释：信息不能从小 A 处经过 2 轮传递到编号 2**

**限制：**

* 2 <= n <= 10
* 1 <= k <= 5
* 1 <= relation.length <= 90, 且 relation[i].length == 2
* 0 <= relation[i][0],relation[i][1] < n 且 relation[i][0] != relation[i][1]

259. [**LCP 11. 期望个数统计**](https://leetcode-cn.com/problems/qi-wang-ge-shu-tong-ji/)

某互联网公司一年一度的春招开始了，一共有 名面试者入选。每名面试者都会提交一份简历，公司会根据提供的简历资料产生一个预估的能力值，数值越大代表越有可能通过面试。

小 A 和小 B 负责审核面试者，他们均有所有面试者的简历，并且将各自根据面试者能力值从大到小的顺序浏览。由于简历事先被打乱过，能力值相同的简历的出现顺序是从它们的全排列中等可能地取一个。现在给定 名面试者的能力值 ，设 代表小 A 和小 B 的浏览顺序中出现在同一位置的简历数，求 的期望。

**示例 1：**

**输入：scores = [1,2,3]**

**输出：3**

**解释：由于面试者能力值互不相同，小 A 和小 B 的浏览顺序一定是相同的。X的期望是 3 。**

**示例 2:**

**输入：scores = [1,1]**

**输出：1**

**解释：设两位面试者的编号为 0, 1。由于他们的能力值都是 1，小 A 和小 B 的浏览顺序都为从全排列 [[0,1],[1,0]] 中等可能地取一个。如果小 A 和小 B 的浏览顺序都是 [0,1] 或者 [1,0] ，那么出现在同一位置的简历数为 2 ，否则是 0 。所以 X 的期望是 (2+0+2+0) \* 1/4 = 1**

**示例 3：**

**输入：scores = [1,1,2]**

**输出：2**

**限制：**

* 1 <= scores.length <= 10^5
* 0 <= scores[i] <= 10^6

260. [**LCP 18. 早餐组合**](https://leetcode-cn.com/problems/2vYnGI/)

小扣在秋日市集选择了一家早餐摊位，一维整型数组 中记录了每种主食的价格，一维整型数组 中记录了每种饮料的价格。小扣的计划选择一份主食和一款饮料，且花费不超过 元。请返回小扣共有多少种购买方案。

注意：答案需要以 1e9 + 7 (1000000007) 为底取模，如：计算初始结果为：1000000008，请返回 1

**示例 1：**

**输入：staple = [10,20,5], drinks = [5,5,2], x = 15**

**输出：6**

**解释：小扣有 6 种购买方案，所选主食与所选饮料在数组中对应的下标分别是：**

**第 1 种方案：staple[0] + drinks[0] = 10 + 5 = 15；**

**第 2 种方案：staple[0] + drinks[1] = 10 + 5 = 15；**

**第 3 种方案：staple[0] + drinks[2] = 10 + 2 = 12；**

**第 4 种方案：staple[2] + drinks[0] = 5 + 5 = 10；**

**第 5 种方案：staple[2] + drinks[1] = 5 + 5 = 10；**

**第 6 种方案：staple[2] + drinks[2] = 5 + 2 = 7。**

**示例 2：**

**输入：staple = [2,1,1], drinks = [8,9,5,1], x = 9**

**输出：8**

**解释：小扣有 8 种购买方案，所选主食与所选饮料在数组中对应的下标分别是：**

**第 1 种方案：staple[0] + drinks[2] = 2 + 5 = 7；**

**第 2 种方案：staple[0] + drinks[3] = 2 + 1 = 3；**

**第 3 种方案：staple[1] + drinks[0] = 1 + 8 = 9；**

**第 4 种方案：staple[1] + drinks[2] = 1 + 5 = 6；**

**第 5 种方案：staple[1] + drinks[3] = 1 + 1 = 2；**

**第 6 种方案：staple[2] + drinks[0] = 1 + 8 = 9；**

**第 7 种方案：staple[2] + drinks[2] = 1 + 5 = 6；**

**第 8 种方案：staple[2] + drinks[3] = 1 + 1 = 2；**

**限制：**

* 1 <= staple.length <= 10^5
* 1 <= drinks.length <= 10^5
* 1 <= staple[i],drinks[i] <= 10^5
* 1 <= x <= 2\*10^5

261. [**LCP 22. 黑白方格画**](https://leetcode-cn.com/problems/ccw6C7/)

小扣注意到秋日市集上有一个创作黑白方格画的摊位。摊主给每个顾客提供一个固定在墙上的白色画板，画板不能转动。画板上有 n \* n 的网格。绘画规则为，小扣可以选择任意多行以及任意多列的格子涂成黑色，所选行数、列数均可为 0。

小扣希望最终的成品上需要有 k 个黑色格子，请返回小扣共有多少种涂色方案。

注意：两个方案中任意一个相同位置的格子颜色不同，就视为不同的方案。

**示例 1：**

**输入：n = 2, k = 2**

**输出：4**

**解释：一共有四种不同的方案：**

**第一种方案：涂第一列；**

**第二种方案：涂第二列；**

**第三种方案：涂第一行；**

**第四种方案：涂第二行。**

**示例 2：**

**输入：n = 2, k = 1**

**输出：0**

**解释：不可行，因为第一次涂色至少会涂两个黑格。**

**示例 3：**

**输入：n = 2, k = 4**

**输出：1**

**解释：共有 2\*2=4 个格子，仅有一种涂色方案。**

**限制：**

* 1 <= n <= 6
* 0 <= k <= n \* n

**262. 1154. Day of the Year**

Given a string date representing a Gregorian calendar date formatted as YYYY-MM-DD, return the day number of the year.

**Example** 1:

**Input**: date = "2019-01-09"

**Output**: 9

**Explanation**: Given date is the 9th day of the year in 2019.

**Example** 2:

**Input**: date = "2019-02-10"

**Output**: 41

**Example** 3:

**Input**: date = "2003-03-01"

**Output**: 60

**Example** 4:

**Input**: date = "2004-03-01"

**Output**: 61

**263. 1160. Find Words That Can Be Formed by Characters**

You are given an array of strings words and a string chars.

A string is good if it can be formed by characters from chars (each character can only be used once).

Return the sum of lengths of all good strings in words.

**Example** 1:

**Input**: words = ["cat","bt","hat","tree"], chars = "atach"

**Output**: 6

**Explanation**:

The strings that can be formed are "cat" and "hat" so the answer is 3 + 3 = 6.

**Example** 2:

**Input**: words = ["hello","world","leetcode"], chars = "welldonehoneyr"

**Output**: 10

**Explanation**:

The strings that can be formed are "hello" and "world" so the answer is 5 + 5 = 10.

**264. 1170. Compare Strings by Frequency of the Smallest Character**

Let's define a function f(s) over a non-empty string s, which calculates the frequency of the smallest character in s. For **Example**, if s = "dcce" then f(s) = 2 because the smallest character is "c" and its frequency is 2.

Now, given string arrays queries and words, return an integer array answer, where each answer[i] is the number of words such that f(queries[i]) < f(W), where W is a word in words.

**Example** 1:

**Input**: queries = ["cbd"], words = ["zaaaz"]

**Output**: [1]

**Explanation**: On the first query we have f("cbd") = 1, f("zaaaz") = 3 so f("cbd") < f("zaaaz").

**Example** 2:

**Input**: queries = ["bbb","cc"], words = ["a","aa","aaa","aaaa"]

**Output**: [1,2]

**Explanation**: On the first query only f("bbb") < f("aaaa"). On the second query both f("aaa") and f("aaaa") are both > f("cc").

**265. 1175. Prime Arrangements**

Return the number of permutations of 1 to n so that prime numbers are at prime indices (1-indexed.)

(Recall that an integer is prime if and only if it is greater than 1, and cannot be written as a product of two positive integers both smaller than it.)

Since the answer may be large, return the answer modulo 10^9 + 7.

**Example** 1:

**Input**: n = 5

**Output**: 12

**Explanation**: For **Example** [1,2,5,4,3] is a valid permutation, but [5,2,3,4,1] is not because the prime number 5 is at index 1.

**Example** 2:

**Input**: n = 100

**Output**: 682289015

**266. 1184. Distance Between Bus Stops**

A bus has n stops numbered from 0 to n - 1 that form a circle. We know the distance between all pairs of neighboring stops where distance[i] is the distance between the stops number i and (i + 1) % n.

The bus goes along both directions i.e. clockwise and counterclockwise.

Return the shortest distance between the given start and destination stops.

**Example** 1:

**Input**: distance = [1,2,3,4], start = 0, destination = 1

**Output**: 1

**Explanation**: Distance between 0 and 1 is 1 or 9, minimum is 1.

**267. 1185. Day of the Week**

Given a date, return the corresponding day of the week for that date.

The Input is given as three integers representing the day, month and year respectively.

Return the answer as one of the following values {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"}.

**Example** 1:

**Input**: day = 31, month = 8, year = 2019

**Output**: "Saturday"

**Example** 2:

**Input**: day = 18, month = 7, year = 1999

**Output**: "Sunday"

**268. 1189. Maximum Number of Balloons**

Given a string text, you want to use the characters of text to form as many instances of the word "balloon" as possible.

You can use each character in text at most once. Return the maximum number of instances that can be formed.

**Example** 1:

**Input**: text = "nlaebolko"

**Output**: 1

**Example** 2:

**Input**: text = "loonbalxballpoon"

**Output**: 2

**269. 1200. Minimum Absolute Difference**

Given an array of distinct integers arr, find all pairs of elements with the minimum absolute difference of any two elements.

Return a list of pairs in ascending order(with respect to pairs), each pair [a, b] follows

* a, b are from arr
* a < b
* b - a equals to the minimum absolute difference of any two elements in arr

**Example** 1:

**Input**: arr = [4,2,1,3]

**Output**: [[1,2],[2,3],[3,4]]

**Explanation**: The minimum absolute difference is 1. List all pairs with difference equal to 1 in ascending order.

**Example** 2:

**Input**: arr = [1,3,6,10,15]

**Output**: [[1,3]]

**Example** 3:

**Input**: arr = [3,8,-10,23,19,-4,-14,27]

**Output**: [[-14,-10],[19,23],[23,27]]

**270. 1207. Unique Number of Occurrences**

Given an array of integers arr, write a function that returns true if and only if the number of occurrences of each value in the array is unique.

**Example** 1:

**Input**: arr = [1,2,2,1,1,3]

**Output**: true

**Explanation**: The value 1 has 3 occurrences, 2 has 2 and 3 has 1. No two values have the same number of occurrences.

**Example** 2:

**Input**: arr = [1,2]

**Output**: false

**Example** 3:

**Input**: arr = [-3,0,1,-3,1,1,1,-3,10,0]

**Output**: true

**271. 1217. Minimum Cost to Move Chips to The Same Position**

We have n chips, where the position of the ith chip is position[i].

We need to move all the chips to the same position. In one step, we can change the position of the ith chip from position[i] to:

position[i] + 2 or position[i] - 2 with cost = 0.

position[i] + 1 or position[i] - 1 with cost = 1.

Return the minimum cost needed to move all the chips to the same position.

**Example** 1:

**Input**: position = [1,2,3]

**Output**: 1

**Explanation**: First step: Move the chip at position 3 to position 1 with cost = 0.

Second step: Move the chip at position 2 to position 1 with cost = 1.

Total cost is 1.

**272. 1221. Split a String in Balanced Strings**

Balanced strings are those who have equal quantity of 'L' and 'R' characters.

Given a balanced string s split it in the maximum amount of balanced strings.

Return the maximum amount of splitted balanced strings.

**Example** 1:

**Input**: s = "RLRRLLRLRL"

**Output**: 4

**Explanation**: s can be split into "RL", "RRLL", "RL", "RL", each substring contains same number of 'L' and 'R'.

**Example** 2:

**Input**: s = "RLLLLRRRLR"

**Output**: 3

**Explanation**: s can be split into "RL", "LLLRRR", "LR", each substring contains same number of 'L' and 'R'.

**Example** 3:

**Input**: s = "LLLLRRRR"

**Output**: 1

**Explanation**: s can be split into "LLLLRRRR".

**Example** 4:

**Input**: s = "RLRRRLLRLL"

**Output**: 2

**Explanation**: s can be split into "RL", "RRRLLRLL", since each substring contains an equal number of 'L' and 'R'

**273. 1232. Check If It Is a Straight Line**

You are given an array coordinates, coordinates[i] = [x, y], where [x, y] represents the coordinate of a point. Check if these points make a straight line in the XY plane.

**Example** 1:

**Input**: coordinates = [[1,2],[2,3],[3,4],[4,5],[5,6],[6,7]]

**Output**: true

**274. 1252. Cells with Odd Values in a Matrix**

Given n and m which are the dimensions of a matrix initialized by zeros and given an array indices where indices[i] = [ri, ci]. For each pair of [ri, ci] you have to increment all cells in row ri and column ci by 1.

Return the number of cells with odd values in the matrix after applying the increment to all indices.

**Example** 1:

**Input**: n = 2, m = 3, indices = [[0,1],[1,1]]

**Output**: 6

**Explanation**: Initial matrix = [[0,0,0],[0,0,0]].

After applying first increment it becomes [[1,2,1],[0,1,0]].

The final matrix will be [[1,3,1],[1,3,1]] which contains 6 odd numbers.

**275. 1266. Minimum Time Visiting All Points**

On a plane there are n points with integer coordinates points[i] = [xi, yi]. Your task is to find the minimum time in seconds to visit all points.

You can move according to the next rules:

In one second always you can either move vertically, horizontally by one unit or diagonally (it means to move one unit vertically and one unit horizontally in one second).

You have to visit the points in the same order as they appear in the array.

**Example** 1:

**Input**: points = [[1,1],[3,4],[-1,0]]

**Output**: 7

**Explanation**: One optimal path is [1,1] -> [2,2] -> [3,3] -> [3,4] -> [2,3] -> [1,2] -> [0,1] -> [-1,0]

Time from [1,1] to [3,4] = 3 seconds

Time from [3,4] to [-1,0] = 4 seconds

Total time = 7 seconds

**276. 1277. Count Square Submatrices with All Ones**

Given a m \* n matrix of ones and zeros, return how many square submatrices have all ones.

**Example** 1:

**Input**: matrix =

[

[0,1,1,1],

[1,1,1,1],

[0,1,1,1]

]

**Output**: 15

**Explanation**:

There are 10 squares of side 1.

There are 4 squares of side 2.

There is 1 square of side 3.

Total number of squares = 10 + 4 + 1 = 15.

**277. 1281. Subtract the Product and Sum of Digits of an Integer**

Given an integer number n, return the difference between the product of its digits and the sum of its digits.

**Example** 1:

**Input**: n = 234

**Output**: 15

**Explanation**:

Product of digits = 2 \* 3 \* 4 = 24

Sum of digits = 2 + 3 + 4 = 9

Result = 24 - 9 = 15

**278. 1287. Element Appearing More Than 25% In Sorted Array**

Given an integer array sorted in non-decreasing order, there is exactly one integer in the array that occurs more than 25% of the time.

Return that integer.

**Example** 1:

**Input**: arr = [1,2,2,6,6,6,6,7,10]

**Output**: 6

**279. 1291. Sequential Digits**

An integer has sequential digits if and only if each digit in the number is one more than the previous digit.

Return a sorted list of all the integers in the range [low, high] inclusive that have sequential digits.

**Example** 1:

**Input**: low = 100, high = 300

**Output**: [123,234]

**280. 1295. Find Numbers with Even Number of Digits**

Given an array nums of integers, return how many of them contain an even number of digits.

**Example** 1:

**Input**: nums = [12,345,2,6,7896]

**Output**: 2

**Explanation**:

12 contains 2 digits (even number of digits).

345 contains 3 digits (odd number of digits).

2 contains 1 digit (odd number of digits).

6 contains 1 digit (odd number of digits).

7896 contains 4 digits (even number of digits).

Therefore only 12 and 7896 contain an even number of digits.

**281. 1297. Maximum Number of Occurrences of a Substring**

Given a string s, return the maximum number of ocurrences of any substring under the following rules:

The number of unique characters in the substring must be less than or equal to maxLetters.

The substring size must be between minSize and maxSize inclusive.

**Example** 1:

**Input**: s = "aababcaab", maxLetters = 2, minSize = 3, maxSize = 4

**Output**: 2

**Explanation**: Substring "aab" has 2 ocurrences in the original string.

It satisfies the conditions, 2 unique letters and size 3 (between minSize and maxSize).

**282. 1299. Replace Elements with Greatest Element on Right Side**

Given an array arr, replace every element in that array with the greatest element among the elements to its right, and replace the last element with -1.

After doing so, return the array.

**Example** 1:

**Input**: arr = [17,18,5,4,6,1]

**Output**: [18,6,6,6,1,-1]

**283. 1-Two Sum**

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to *target*.

You may assume that each **Input** would have **exactly one solution**, and you may not use the same element twice.

You can return the answer in any order.

**Example 1:**

**Input:** nums = [2,7,11,15], target = 9

**Output:** [0,1]

**Output:** Because nums[0] + nums[1] == 9, we return [0, 1].

**Example 2:**

**Input:** nums = [3,2,4], target = 6

**Output:** [1,2]

**Example 3:**

**Input:** nums = [3,3], target = 6

**Output:** [0,1]

**Constraints:**

* 2 <= nums.length <= 105
* -109 <= nums[i] <= 109
* -109 <= target <= 109
* **Only one valid answer exists.**

**284. 3- Longest Substring Without Repeating Characters**

Given a string s, find the length of the **longest substring** without repeating characters.

**Example 1:**

**Input:** s = "abcabcbb"

**Output:** 3

**Explanation:** The answer is "abc", with the length of 3.

**Example 2:**

**Input:** s = "bbbbb"

**Output:** 1

**Explanation:** The answer is "b", with the length of 1.

**Example 3:**

**Input:** s = "pwwkew"

**Output:** 3

**Explanation:** The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

**Example 4:**

**Input:** s = ""

**Output:** 0

**Constraints:**

* 1 <= s.length <= 1000
* s consist of only digits and English letters (lower-case and/or upper-case),

**285. 5-Longest Palindromic Substring**

Given a string s, return *the longest palindromic substring* in s.

**Example 1:**

**Input:** s = "babad"

**Output:** "bab"

**Note:** "aba" is also a valid answer.

**Example 2:**

**Input:** s = "cbbd"

**Output:** "bb"

**Example 3:**

**Input:** s = "a"

**Output:** "a"

**Example 4:**

**Input:** s = "ac"

**Output:** "a"

**Constraints:**

* 1 <= s.length <= 1000
* s consist of only digits and English letters (lower-case and/or upper-case),

**286. 6- ZigZag Conversion**

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

P A H N

A P L S I I G

Y I R

And then read line by line: "PAHNAPLSIIGYIR"

Write the code that will take a string and make this conversion given a number of rows:

string convert(string s, int numRows);

**Example 1:**

**Input:** s = "PAYPALISHIRING", numRows = 3

**Output:** "PAHNAPLSIIGYIR"

**Example 2:**

**Input:** s = "PAYPALISHIRING", numRows = 4

**Output:** "PINALSIGYAHRPI"

**Explanation:**

P I N

A L S I G

Y A H R

P I

**Example 3:**

**Input:** s = "A", numRows = 1

**Output:** "A"

**Constraints:**

* 1 <= s.length <= 1000
* s consists of English letters (lower-case and upper-case), ',' and '.'.
* 1 <= numRows <= 1000

**287. 13- Roman to Integer**

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

**Symbol** **Value**

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

For **Example**, 2 is written as II in Roman numeral, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

* I can be placed before V (5) and X (10) to make 4 and 9.
* X can be placed before L (50) and C (100) to make 40 and 90.
* C can be placed before D (500) and M (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

**Example 1:**

**Input:** s = "III"

**Output:** 3

**Example 2:**

**Input:** s = "IV"

**Output:** 4

**Example 3:**

**Input:** s = "IX"

**Output:** 9

**Example 4:**

**Input:** s = "LVIII"

**Output:** 58

**Explanation:** L = 50, V= 5, III = 3.

**Example 5:**

**Input:** s = "MCMXCIV"

**Output:** 1994

**Explanation:** M = 1000, CM = 900, XC = 90 and IV = 4.

**Constraints:**

* 1 <= s.length <= 15
* s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').
* It is **guaranteed** that s is a valid roman numeral in the range [1, 3999].

**288. 17. Letter Combinations of a Phone Number**

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



**Example 1:**

**Input:** digits = "23"

**Output:** ["ad","ae","af","bd","be","bf","cd","ce","cf"]

**Example 2:**

**Input:** digits = ""

**Output:** []

**Example 3:**

**Input:** digits = "2"

**Output:** ["a","b","c"]

**Constraints:**

* 0 <= digits.length <= 4
* digits[i] is a digit in the range ['2', '9'].

**289. 20- Valid Parentheses**

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the **Input** string is valid.

An **Input** string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

**Example 1:**

**Input:** s = "()"

**Output:** true

**Example 2:**

**Input:** s = "()[]{}"

**Output:** true

**Example 3:**

**Input:** s = "(]"

**Output:** false

**Example 4:**

**Input:** s = "([)]"

**Output:** false

**Example 5:**

**Input:** s = "{[]}"

**Output:** true

**Constraints:**

* 1 <= s.length <= 104
* s consists of parentheses only '()[]{}'.

**290. 36- Valid Sudoku**

Determine if a 9 x 9 Sudoku board is valid. Only the filled cells need to be validated **according to the following rules**:

1. Each row must contain the digits 1-9 without repetition.
2. Each column must contain the digits 1-9 without repetition.
3. Each of the nine 3 x 3 sub-boxes of the grid must contain the digits 1-9 without repetition.

**Note:**

* A Sudoku board (partially filled) could be valid but is not necessarily solvable.
* Only the filled cells need to be validated according to the mentioned rules.

**Example 1:**



**Input:** board =

[["5","3",".",".","7",".",".",".","."]

,["6",".",".","1","9","5",".",".","."]

,[".","9","8",".",".",".",".","6","."]

,["8",".",".",".","6",".",".",".","3"]

,["4",".",".","8",".","3",".",".","1"]

,["7",".",".",".","2",".",".",".","6"]

,[".","6",".",".",".",".","2","8","."]

,[".",".",".","4","1","9",".",".","5"]

,[".",".",".",".","8",".",".","7","9"]]

**Output:** true

**Example 2:**

**Input:** board =

[["8","3",".",".","7",".",".",".","."]

,["6",".",".","1","9","5",".",".","."]

,[".","9","8",".",".",".",".","6","."]

,["8",".",".",".","6",".",".",".","3"]

,["4",".",".","8",".","3",".",".","1"]

,["7",".",".",".","2",".",".",".","6"]

,[".","6",".",".",".",".","2","8","."]

,[".",".",".","4","1","9",".",".","5"]

,[".",".",".",".","8",".",".","7","9"]]

**Output:** false

**Explanation:** Same as **Example** 1, except with the **5** in the top left corner being modified to **8**. Since there are two 8's in the top left 3x3 sub-box, it is invalid.

**Constraints:**

* board.length == 9
* board[i].length == 9
* board[i][j] is a digit or '.'.

**291. 46- Permutations**

Given a collection of **distinct** integers, return all possible permutations.

**Example:**

**Input:** [1,2,3]

**Output:**

[

[1,2,3],

[1,3,2],

[2,1,3],

[2,3,1],

[3,1,2],

[3,2,1]

]

**292. 48- Rotate Image**

You are given an *n* x *n* 2D matrix representing an image, rotate the image by 90 degrees (clockwise).

You have to rotate the image [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm), which means you have to modify the **Input** 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

**Example 1:**



**Input:** matrix = [[1,2,3],[4,5,6],[7,8,9]]

**Output:** [[7,4,1],[8,5,2],[9,6,3]]

**Example 2:**



**Input:** matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]

**Output:** [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]

**Example 3:**

**Input:** matrix = [[1]]

**Output:** [[1]]

**Example 4:**

**Input:** matrix = [[1,2],[3,4]]

**Output:** [[3,1],[4,2]]

**Constraints:**

* matrix.length == n
* matrix[i].length == n
* 1 <= n <= 20
* -1000 <= matrix[i][j] <= 1000

**293. 49- Group Anagrams**

Given an array of strings strs, group **the anagrams** together. You can return the answer in **any order**.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

**Example 1:**

**Input:** strs = ["eat","tea","tan","ate","nat","bat"]

**Output:** [["bat"],["nat","tan"],["ate","eat","tea"]]

**Example 2:**

**Input:** strs = [""]

**Output:** [[""]]

**Example 3:**

**Input:** strs = ["a"]

**Output:** [["a"]]

**Constraints:**

* 1 <= strs.length <= 104
* 0 <= strs[i].length <= 100
* strs[i] consists of lower-case English letters.

**294. 53- Maximum Subarray**

Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return *its sum*.

**Follow up:** If you have figured out the O(n) solution, try coding another solution using the **divide and conquer** approach, which is more subtle.

**Example 1:**

**Input:** nums = [-2,1,-3,4,-1,2,1,-5,4]

**Output:** 6

**Explanation:** [4,-1,2,1] has the largest sum = 6.

**Example 2:**

**Input:** nums = [1]

**Output:** 1

**Example 3:**

**Input:** nums = [0]

**Output:** 0

**Example 4:**

**Input:** nums = [-1]

**Output:** -1

**Example 5:**

**Input:** nums = [-2147483647]

**Output:** -2147483647

**Constraints:**

* 1 <= nums.length <= 2 \* 104
* -231 <= nums[i] <= 231 - 1

**295. 54- Spiral Matrix**

Given a matrix of *m* x *n* elements (*m* rows, *n* columns), return all elements of the matrix in spiral order.

**Example 1:**

**Input:**

[

[ 1, 2, 3 ],

[ 4, 5, 6 ],

[ 7, 8, 9 ]

]

**Output:** [1,2,3,6,9,8,7,4,5]

**Example 2:**

**Input:**

[

[1, 2, 3, 4],

[5, 6, 7, 8],

[9,10,11,12]

]

**Output:** [1,2,3,4,8,12,11,10,9,5,6,7]

**296. 55- Jump Game**

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Determine if you are able to reach the last index.

**Example 1:**

**Input:** nums = [2,3,1,1,4]

**Output:** true

**Explanation:** Jump 1 step from index 0 to 1, then 3 steps to the last index.

**Example 2:**

**Input:** nums = [3,2,1,0,4]

**Output:** false

**Explanation:** You will always arrive at index 3 no matter what. Its maximum jump length is 0, which makes it impossible to reach the last index.

**Constraints:**

* 1 <= nums.length <= 3 \* 10^4
* 0 <= nums[i][j] <= 10^5

**297. 56- Merge Intervals**

Given a collection of intervals, merge all overlapping intervals.

**Example 1:**

**Input:** intervals = [[1,3],[2,6],[8,10],[15,18]]

**Output:** [[1,6],[8,10],[15,18]]

**Explanation:** Since intervals [1,3] and [2,6] overlaps, merge them into [1,6].

**Example 2:**

**Input:** intervals = [[1,4],[4,5]]

**Output:** [[1,5]]

**Explanation:** Intervals [1,4] and [4,5] are considered overlapping.

**NOTE:** **Input** types have been changed on April 15, 2019. Please reset to default code definition to get new method signature.

**Constraints:**

* intervals[i][0] <= intervals[i][1]

**298. 59- Spiral Matrix II**

Given a positive integer *n*, generate a square matrix filled with elements from 1 to *n*2 in spiral order.

**Example:**

**Input:** 3

**Output:**

[

[ 1, 2, 3 ],

[ 8, 9, 4 ],

[ 7, 6, 5 ]

]

**299. 62- Unique Paths**

A robot is located at the top-left corner of a m x n grid (marked 'Start' in the diagram below).

The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below).

How many possible unique paths are there?

**Example 1:**



**Input:** m = 3, n = 7

**Output:** 28

**Example 2:**

**Input:** m = 3, n = 2

**Output:** 3

**Explanation:**

From the top-left corner, there are a total of 3 ways to reach the bottom-right corner:

1. Right -> Down -> Down

2. Down -> Down -> Right

3. Down -> Right -> Down

**Example 3:**

**Input:** m = 7, n = 3

**Output:** 28

**Example 4:**

**Input:** m = 3, n = 3

**Output:** 6

**Constraints:**

* 1 <= m, n <= 100
* It's guaranteed that the answer will be less than or equal to 2 \* 109.

**300. 71- Simplify Path**

Given an **absolute path** for a file (Unix-style), simplify it. Or in other words, convert it to the **canonical path**.

In a UNIX-style file system, a period . refers to the current directory. Furthermore, a double period .. moves the directory up a level.

Note that the returned canonical path must always begin with a slash /, and there must be only a single slash / between two directory names. The last directory name (if it exists) **must not** end with a trailing /. Also, the canonical path must be the **shortest** string representing the absolute path.

**Example 1:**

**Input: "**/home/"

**Output: "**/home"

**Explanation:** Note that there is no trailing slash after the last directory name.

**Example 2:**

**Input: "**/../"

**Output: "**/"

**Explanation:** Going one level up from the root directory is a no-op, as the root level is the highest level you can go.

**Example 3:**

**Input: "**/home//foo/"

**Output: "**/home/foo"

**Explanation:** In the canonical path, multiple consecutive slashes are replaced by a single one.

**Example 4:**

**Input: "**/a/./b/../../c/"

**Output: "**/c"

**Example 5:**

**Input: "**/a/../../b/../c//.//"

**Output: "**/c"

**Example 6:**

**Input: "**/a//b////c/d//././/.."

**Output: "**/a/b/c"

**301. 75- Sort Colors**

Given an array nums with n objects colored red, white, or blue, sort them [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm)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.

**Follow up:**

* Could you solve this problem without using the library's sort function?
* Could you come up with a one-pass algorithm using only O(1) constant space?

**Example 1:**

**Input:** nums = [2,0,2,1,1,0]

**Output:** [0,0,1,1,2,2]

**Example 2:**

**Input:** nums = [2,0,1]

**Output:** [0,1,2]

**Example 3:**

**Input:** nums = [0]

**Output:** [0]

**Example 4:**

**Input:** nums = [1]

**Output:** [1]

**Constraints:**

* n == nums.length
* 1 <= n <= 300
* nums[i] is 0, 1, or 2.

**302. 89- Gray Code**

The gray code is a binary numeral system where two successive values differ in only one bit.

Given a non-negative integer *n* representing the total number of bits in the code, print the sequence of gray code. A gray code sequence must begin with 0.

**Example 1:**

**Input:** 2

**Output:** [0,1,3,2]

**Explanation:**

00 - 0

01 - 1

11 - 3

10 - 2

For a given *n*, a gray code sequence may not be uniquely defined.

For **Example**, [0,2,3,1] is also a valid gray code sequence.

00 - 0

10 - 2

11 - 3

01 - 1

**Example 2:**

**Input:** 0

**Output:** [0]

**Explanation:** We define the gray code sequence to begin with 0.

  A gray code sequence of *n* has size = 2n, which for *n* = 0 the size is 20 = 1.

  Therefore, for *n* = 0 the gray code sequence is [0].