1. 202-Happy Number

Write an algorithm to determine if a number is "happy".

A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers.

Example:

Input: 19

Output: true

Explanation:

$1^2 + 9^2 = 82$

$8^2 + 2^2 = 68$

$6^2 + 8^2 = 100$

$1^2 + 0^2 + 0^2 = 1$

2. 204-Count Primes

Count the number of prime numbers less than a non-negative number, ***n***.

Example:

Input: 10

Output: 4

Explanation: There are 4 prime numbers less than 10, they are 2, 3, 5, 7.

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.

4. 217-Contains Duplicate

Given an array of integers, find if the array contains any duplicates.

Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

Example 1:

Input: [1,2,3,1]

Output: true

Example 2:

Input: [1,2,3,4]

Output: false

Example 3:

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

Output: true

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

6. 231-Power of Two

Given an integer, write a function to determine if it is a power of two.

Example 1:

Input: 1

Output: true

Explanation: $2^0 = 1$

Example 2:

Input: 16

Output: true

Explanation: $2^4 = 16$

Example 3:

Input: 218

Output: false

7. 242-Valid Anagram

Given two strings *s* and *t* , write a function to determine if *t* is an anagram of *s*.

Example 1:

Input: *s* = "anagram", *t* = "nagaram"

Output: true

Example 2:

Input: *s* = "rat", *t* = "car"

Output: false

Note:

You may assume the string contains only lowercase alphabets.

Follow up:
What if the inputs contain unicode characters? How would you adapt your solution to such case?

8. 258-Add Digits

Given a non-negative integer num, repeatedly add all its digits until the result has only one digit.

Example:

Input: 38

Output: 2

Explanation: The process is like: 3 + 8 = 11, 1 + 1 = 2.

Since 2 has only one digit, return it.

Follow up:
Could you do it without any loop/recursion in O(1) runtime?

9. 263-Ugly Number

Write a program to check whether a given number is an ugly number.

Ugly numbers are **positive numbers** whose prime factors only include 2, 3, 5.

Example 1:

Input: 6

Output: true

Explanation: 6 = 2 × 3

Example 2:

Input: 8

Output: true

Explanation: 8 = 2 × 2 × 2

Example 3:

Input: 14

Output: false

Explanation: 14 is not ugly since it includes another prime factor 7.

Note:

1. 1 is typically treated as an ugly number.

2. Input is within the 32-bit signed integer range: $[-2^{31}, \ 2^{31}-1]$.

10. 268-Missing Number

Given an array containing *n* distinct numbers taken from 0, 1, 2, ..., n, find the one that is missing from the array.

Example 1:

Input: [3,0,1]

Output: 2

Example 2:

Input: [9,6,4,2,3,5,7,0,1]

Output: 8

Note:
Your algorithm should run in linear runtime complexity. Could you implement it using only constant extra space complexity?

11. 283-Move Zeroes

Given an array nums, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements.

Example:

Input: [0,1,0,3,12]

Output: [1,3,12,0,0]

Note:

1. You must do this **in-place** without making a copy of the array.
2. Minimize the total number of operations.

12. 290-Word Pattern

Given a pattern and a string str, find if str follows the same pattern.

Here **follow** means a full match, such that there is a bijection between a letter in pattern and a **non-empty** word in str.

Example 1:

Input: pattern = "abba", str = "dog cat cat dog"

Output: true

Example 2:

Input:pattern = "abba", str = "dog cat cat fish"

Output: false

Example 3:

Input: pattern = "aaaa", str = "dog cat cat dog"

Output: false

Example 4:

Input: pattern = "abba", str = "dog dog dog dog"

Output: false

Notes:
You may assume pattern contains only lowercase letters, and str contains lowercase letters separated by a single space.

13. 292-Nim Game

You are playing the following Nim Game with your friend: There is a heap of stones on the table, each time one of you take turns to remove 1 to 3 stones. The one who removes the last stone will be the winner. You will take the first turn to remove the stones.

Both of you are very clever and have optimal strategies for the game. Write a function to determine whether you can win the game given the number of stones in the heap.

Example:

Input: 4

Output: false

Explanation: If there are 4 stones in the heap, then you will never win the game;

No matter 1, 2, or 3 stones you remove, the last stone will always be

removed by your friend.

14. 326-Power of Three

Given an integer, write a function to determine if it is a power of three.

Example 1:

Input: 27

Output: true

Example 2:

Input: 0

Output: false

Example 3:

Input: 9

Output: true

Example 4:

Input: 45

Output: false

Follow up:
Could you do it without using any loop / recursion?

15. 342-Power of Four

Given an integer (signed 32 bits), write a function to check whether it is a power of 4.

Example 1:

Input: 16

Output: true

Example 2:

Input: 5

Output: false

Follow up: Could you solve it without loops/recursion?

16. 344-Reverse String、

Write a function that takes a string as input and returns the string reversed.

Example 1:

Input: "hello"

Output: "olleh"

Example 2:

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

Output: "amanaP :lanac a ,nalp a ,nam A"

17. 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".

18. 349-Intersection of Two Arrays

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

Example 1:

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

Output: [2]

Example 2:

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

Output: [9,4]

Note:

- Each element in the result must be unique.
- The result can be in any order.
-

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?

20. 367-Valid Perfect Square

Given a positive integer *num*, write a function which returns True if *num* is a perfect square else False.

Note: Do not use any built-in library function such as `sqrt`.

Example 1:

```
Input: 16

Output: true
```

Example 2:

```
Input: 14

Output: false
```

21. 383-Ransom Note

Given an arbitrary ransom note string and another string containing letters from all the magazines, write a function that will return true if the ransom note can be constructed from the magazines ; otherwise, it will return false.

Each letter in the magazine string can only be used once in your ransom note.

Note:
You may assume that both strings contain only lowercase letters.

```
canConstruct("a", "b") -> false

canConstruct("aa", "ab") -> false

canConstruct("aa", "aab") -> true
```

22. 387-First Unique Character in a String

Given a string, find the first non-repeating character in it and return it's index. If it doesn't exist, return -1.

Examples:

```
s = "leetcode"

return 0.



s = "loveleetcode",

return 2.
```

Note: You may assume the string contain only lowercase letters.

23. 389-Find the Difference

Given two strings ***s*** and ***t*** which consist of only lowercase letters.

String ***t*** is generated by random shuffling string ***s*** and then add one more letter at a random position.

Find the letter that was added in ***t***.

Example:

```
Input:

s = "abcd"

t = "abcde"



Output:

e



Explanation:

'e' is the letter that was added.
```

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 ($n < 2^{31}$).

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.

25. 409-Longest Palindrome

Given a string which consists of lowercase or uppercase letters, find the length of the longest palindromes that can be built with those letters.

This is case sensitive, for example "Aa" is not considered a palindrome here.

Note:
Assume the length of given string will not exceed 1,010.

Example:

```
Input:

"abccccdd"

Output:

7

Explanation:
```

One longest palindrome that can be built is "dccaccd", whose length is 7.

26. 414-Third Maximum Number

Given a **non-empty** array of integers, return the **third** maximum number in this array. If it does not exist, return the maximum number. The time complexity must be in O(n).

Example 1:

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

Output: 1

Explanation: The third maximum is 1.
```

Example 2:

```
Input: [1, 2]

Output: 2

Explanation: The third maximum does not exist, so the maximum (2) is returned instead.
```

Example 3:

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

Output: 1

Explanation: Note that the third maximum here means the third maximum distinct number.
Both numbers with value 2 are both considered as second maximum.
```

27. 434-Number of Segments in a String

Count the number of segments in a string, where a segment is defined to be a contiguous sequence of non-space characters.

Please note that the string does not contain any **non-printable** characters.

Example:

Input: "Hello, my name is John"

Output: 5

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".

29. 441-Arranging Coins

You have a total of *n* coins that you want to form in a staircase shape, where every *k*-th row must have exactly *k* coins.

Given *n*, find the total number of **full** staircase rows that can be formed.

n is a non-negative integer and fits within the range of a 32-bit signed integer.

Example 1:

```
n = 5



The coins can form the following rows:

¤

¤ ¤

¤ ¤



Because the 3rd row is incomplete, we return 2.
```

Example 2:

```
n = 8



The coins can form the following rows:

¤

¤ ¤

¤ ¤ ¤

¤ ¤



Because the 4th row is incomplete, we return 3.
```

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 $1 \le a[i] \le n$ (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 g_i, which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s_j. If $s_j >= g_i$, 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:
$0 \leq x, y < 2^{31}$.

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 [3,10^4] 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 $a^2 + b^2 = c$.

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.

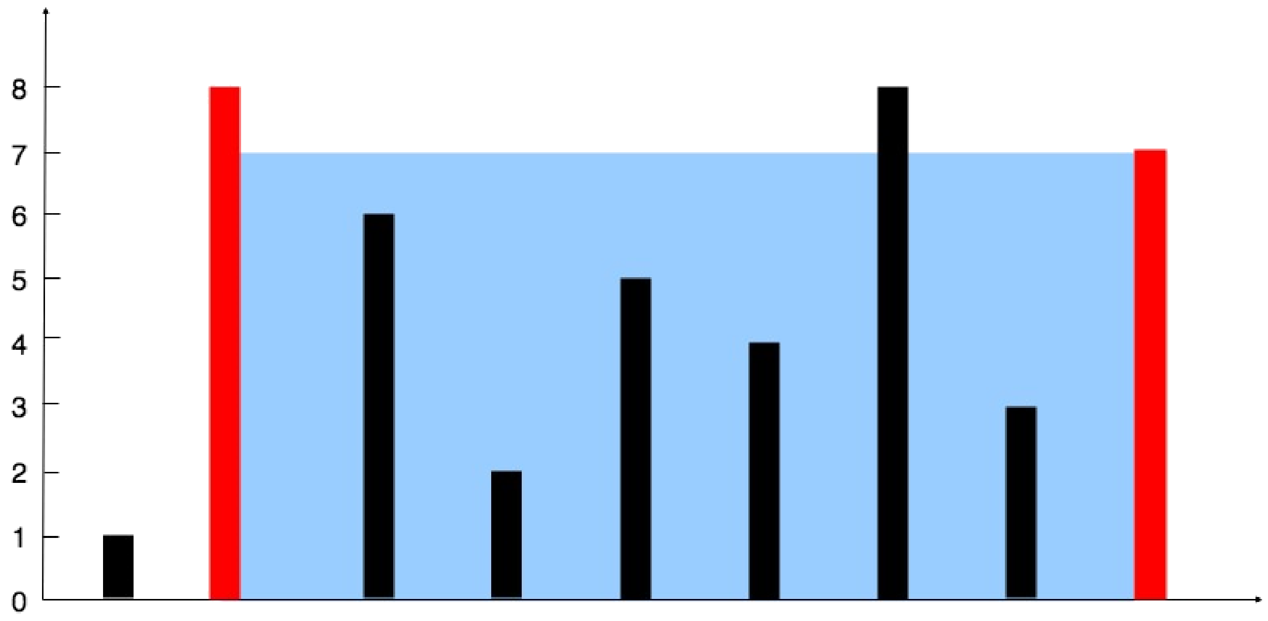
Follow up:

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 inputs 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:

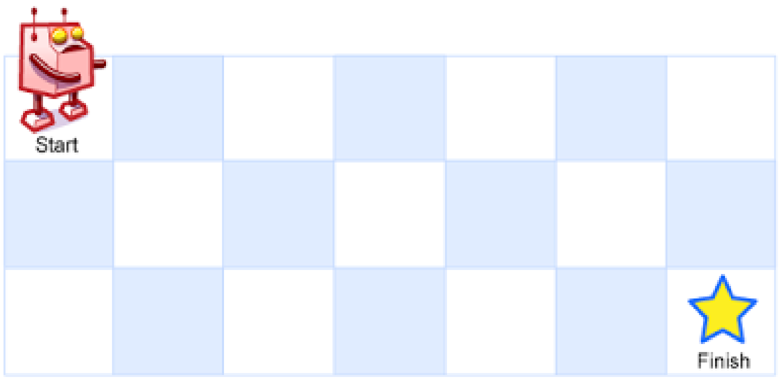
- -100.0 < x < 100.0
- n is a 32-bit signed integer, within the range [−231, 231 − 1]

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 outputs [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: inputs = ["RecentCounter","ping","ping","ping","ping"], inputs = [[],[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