目录

[451. Sort Characters By Frequency 2](#_Toc21270873)

[452. Minimum Number of Arrows to Burst Balloons 4](#_Toc21270874)

[453. Minimum Moves to Equal Array Elements 5](#_Toc21270875)

[454. 4Sum II 6](#_Toc21270876)

[455. Assign Cookies 7](#_Toc21270877)

[456. 132 Pattern 8](#_Toc21270878)

[457. Circular Array Loop 9](#_Toc21270879)

[458. Poor Pigs 11](#_Toc21270880)

[459. Repeated Substring Pattern 12](#_Toc21270881)

[460. LFU Cache 13](#_Toc21270882)

[461. Hamming Distance 14](#_Toc21270883)

[462. Minimum Moves to Equal Array Elements II 15](#_Toc21270884)

[463. Island Perimeter 16](#_Toc21270885)

[464. Can I Win 17](#_Toc21270886)

[466. Count The Repetitions 18](#_Toc21270887)

[467. Unique Substrings in Wraparound String 19](#_Toc21270888)

[468. Validate IP Address 20](#_Toc21270889)

[470. Implement Rand10() Using Rand7() 22](#_Toc21270890)

[472. Concatenated Words 23](#_Toc21270891)

[473. Matchsticks to Square 24](#_Toc21270892)

[474. Ones and Zeroes 25](#_Toc21270893)

[475. Heaters 26](#_Toc21270894)

[476. Number Complement 27](#_Toc21270895)

[477. Total Hamming Distance 28](#_Toc21270896)

[478. Generate Random Point in a Circle 29](#_Toc21270897)

[479. Largest Palindrome Product 30](#_Toc21270898)

[480. Sliding Window Median 31](#_Toc21270899)

[481. Magical String 32](#_Toc21270900)

[482. License Key Formatting 33](#_Toc21270901)

[483. Smallest Good Base 34](#_Toc21270902)

[485. Max Consecutive Ones 35](#_Toc21270903)

[486. Predict the Winner 36](#_Toc21270904)

[488. Zuma Game 37](#_Toc21270905)

[491. Increasing Subsequences 39](#_Toc21270906)

[492. Construct the Rectangle 40](#_Toc21270907)

[493. Reverse Pairs 41](#_Toc21270908)

[494. Target Sum 42](#_Toc21270909)

[495. Teemo Attacking 43](#_Toc21270910)

[496. Next Greater Element I 45](#_Toc21270911)

[497. Random Point in Non-overlapping Rectangles 46](#_Toc21270912)

[498. Diagonal Traverse 48](#_Toc21270913)

[500. Keyboard Row 49](#_Toc21270914)

### 451. Sort Characters By Frequency

Medium

Given a string, sort it in decreasing order based on the frequency of characters.

**Example 1:**

**Input:**

"tree"

**Output:**

"eert"

**Explanation:**

'e' appears twice while 'r' and 't' both appear once.

So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.

**Example 2:**

**Input:**

"cccaaa"

**Output:**

"cccaaa

**Explanation:**

Both 'c' and 'a' appear three times, so "aaaccc" is also a valid answer.

Note that "cacaca" is incorrect, as the same characters must be together.

**Example 3:**

**Input:**

"Aabb"

**Output:**

"bbAa"

**Explanation:**

"bbaA" is also a valid answer, but "Aabb" is incorrect.

Note that 'A' and 'a' are treated as two different characters.

class Solution **{**

public**:**

string frequencySort**(**string s**)** **{**

string res**;**

unordered\_map**<**char**,** int**>** m**;**

**for** **(**auto **&**c **:** s**)** m**[**c**]++;**

set**<**pair**<**int**,** char**>>** st**;**

**for** **(**auto **&**i **:** m**)** st**.**insert**({**i**.**second**,** i**.**first**});**

**for** **(**auto it **=** st**.**rbegin**();** it **!=** st**.**rend**();** it**++)**

res **+=** string**(**it**->**first**,** it**->**second**);**

**return** res**;**

**}**

**};**

### 452. Minimum Number of Arrows to Burst Balloons

Medium

There are a number of spherical balloons spread in two-dimensional space. For each balloon, provided input is the start and end coordinates of the horizontal diameter. Since it's horizontal, y-coordinates don't matter and hence the x-coordinates of start and end of the diameter suffice. Start is always smaller than end. There will be at most 104 balloons.

An arrow can be shot up exactly vertically from different points along the x-axis. A balloon with xstart and xend bursts by an arrow shot at x if xstart ≤ x ≤ xend. There is no limit to the number of arrows that can be shot. An arrow once shot keeps travelling up infinitely. The problem is to find the minimum number of arrows that must be shot to burst all balloons.

**Example:**

**Input:**

[[10,16], [2,8], [1,6], [7,12]]

**Output:**

2

**Explanation:**

One way is to shoot one arrow for example at x = 6 (bursting the balloons [2,8] and [1,6]) and another arrow at x = 11 (bursting the other two balloons).

class Solution **{**

public**:**

int findMinArrowShots**(**vector**<**vector**<**int**>>&** points**)** **{**

**if** **(**points**.**empty**())** **return** 0**;**

sort**(**points**.**begin**(),** points**.**end**(),** **[](**const vector**<**int**>** **&**lhs**,** const vector**<**int**>** **&**rhs**)** **{**

**return** lhs**[**0**]** **!=** rhs**[**0**]** **?** lhs**[**0**]** **<** rhs**[**0**]**

**:** lhs**[**1**]** **<** rhs**[**1**];**

**});**

int res **=** 1**,** low **=** INT\_MIN**,** high **=** INT\_MAX**;**

**for** **(**auto **&**pnt **:** points**)** **{**

**if** **(**pnt**[**0**]** **>** high**)** **{**

low **=** pnt**[**0**],** high **=** pnt**[**1**];**

res**++;**

**}**

high **=** min**(**high**,** pnt**[**1**]);**

**}**

**return** res**;**

**}**

**};**

### 453. Minimum Moves to Equal Array Elements

Easy

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]

class Solution **{**

public**:**

int minMoves**(**vector**<**int**>&** nums**)** **{**

long long Min **=** nums**[**0**],** sum **=** 0**;**

**for** **(**const auto **&**i **:** nums**)** **{**

**if** **(**i **<** Min**)** Min **=** i**;**

sum **+=** i**;**

**}**

**return** sum **-** Min**\***nums**.**size**();**

**}**

**};**

### 454. 4Sum II

Medium

Given four lists A, B, C, D of integer values, compute how many tuples (i, j, k, l) there are such that A[i] + B[j] + C[k] + D[l] is zero.

To make problem a bit easier, all A, B, C, D have same length of N where 0 ≤ N ≤ 500. All integers are in the range of -228 to 228 - 1 and the result is guaranteed to be at most 231 - 1.

**Example:**

**Input:**

A = [ 1, 2]

B = [-2,-1]

C = [-1, 2]

D = [ 0, 2]

**Output:**

2

**Explanation:**

The two tuples are:

1. (0, 0, 0, 1) -> A[0] + B[0] + C[0] + D[1] = 1 + (-2) + (-1) + 2 = 0

2. (1, 1, 0, 0) -> A[1] + B[1] + C[0] + D[0] = 2 + (-1) + (-1) + 0 = 0

class Solution **{**

public**:**

int fourSumCount**(**vector**<**int**>&** A**,** vector**<**int**>&** B**,** vector**<**int**>&** C**,** vector**<**int**>&** D**)** **{**

unordered\_map**<**int**,** int**>** m**;**

**for(**auto a **:** A**)** **{**

**for(**auto b **:** B**)** **{**

**++**m**[**a**+**b**];**

**}**

**}**

int res **=** 0**;**

**for(**auto c **:** C**)** **{**

**for(**auto d **:** D**)** **{**

auto it **=** m**.**find**(-**c**-**d**);**

**if** **(**it **!=** m**.**end**())** **{**

res **+=** it**->**second**;**

**}**

**}**

**}**

**return** res**;**

**}**

**};**

### 455. Assign Cookies

Easy

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.

class Solution **{**

public**:**

int findContentChildren**(**vector**<**int**>&** g**,** vector**<**int**>&** s**)** **{**

sort**(**g**.**begin**(),** g**.**end**());**

sort**(**s**.**begin**(),** s**.**end**());**

int i **=** 0**,** j **=** 0**,** n **=** g**.**size**(),** m **=** s**.**size**();**

int res **=** 0**;**

**while** **(**i **<** n **&&** j **<** m**)** **{**

**if** **(**g**[**i**]** **<=** s**[**j**++])** **{**

res**++;**

i**++;**

**}**

**}**

**return** res**;**

**}**

**};**

### 456. 132 Pattern

Medium

Given a sequence of n integers a1, a2, ..., an, a 132 pattern is a subsequence a**i**, a**j**, a**k** such that **i** < **j** < **k** and a**i** < a**k** < a**j**. Design an algorithm that takes a list of n numbers as input and checks whether there is a 132 pattern in the list.

**Note:** n will be less than 15,000.

**Example 1:**

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

**Output:** False

**Explanation:** There is no 132 pattern in the sequence.

**Example 2:**

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

**Output:** True

**Explanation:** There is a 132 pattern in the sequence: [1, 4, 2].

**Example 3:**

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

**Output:** True

**Explanation:** There are three 132 patterns in the sequence: [-1, 3, 2], [-1, 3, 0] and [-1, 2, 0].

class Solution **{**

public**:**

bool find132pattern**(**vector**<**int**>&** nums**)** **{**

int third **=** INT\_MIN**;**

stack**<**int**>** stk**;** //second, all of it is bigger than Third

**for** **(**int i **=** nums**.**size**()** **-** 1**;** i **>=** 0**;** **--**i**)** **{**

**if** **(**nums**[**i**]** **<** third**)** **return** **true;**

**while** **(!**stk**.**empty**()** **&&** nums**[**i**]** **>** stk**.**top**())** **{**

third **=** stk**.**top**();**

stk**.**pop**();**

**}**

stk**.**push**(**nums**[**i**]);**

**}**

**return** **false;**

**}**

**};**

### 457. Circular Array Loop

Medium

You are given a **circular** array nums of positive and negative integers. If a number *k* at an index is positive, then move forward *k* steps. Conversely, if it's negative (-*k*), move backward *k* steps. Since the array is circular, you may assume that the last element's next element is the first element, and the first element's previous element is the last element.

Determine if there is a loop (or a cycle) in nums. A cycle must start and end at the same index and the cycle's length > 1. Furthermore, movements in a cycle must all follow a single direction. In other words, a cycle must not consist of both forward and backward movements.

**Example 1:**

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

**Output:** true

**Explanation:** There is a cycle, from index 0 -> 2 -> 3 -> 0. The cycle's length is 3.

**Example 2:**

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

**Output:** false

**Explanation:** The movement from index 1 -> 1 -> 1 ... is not a cycle, because the cycle's length is 1. By definition the cycle's length must be greater than 1.

**Example 3:**

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

**Output:** false

**Explanation:** The movement from index 1 -> 2 -> 1 -> ... is not a cycle, because movement from index 1 -> 2 is a forward movement, but movement from index 2 -> 1 is a backward movement. All movements in a cycle must follow a single direction.

**Note:**

1. -1000 ≤ nums[i] ≤ 1000
2. nums[i] ≠ 0
3. 1 ≤ nums.length ≤ 5000

### 458. Poor Pigs

Hard

There are 1000 buckets, one and only one of them is poisonous, while the rest are filled with water. They all look identical. If a pig drinks the poison it will die within 15 minutes. What is the minimum amount of pigs you need to figure out which bucket is poisonous within one hour?

Answer this question, and write an algorithm for the general case.

**General case:**

If there are n buckets and a pig drinking poison will die within m minutes, how many pigs (x) you need to figure out the **poisonous** bucket within p minutes? There is exactly one bucket with poison.

**Note:**

1. A pig can be allowed to drink simultaneously on as many buckets as one would like, and the feeding takes no time.
2. After a pig has instantly finished drinking buckets, there has to be a **cool down time** of *m*minutes. During this time, only observation is allowed and no feedings at all.
3. Any given bucket can be sampled an infinite number of times (by an unlimited number of pigs).

### 459. Repeated Substring Pattern

Easy

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.)

class Solution **{**

public**:**

bool repeatedSubstringPattern**(**string s**)** **{**

int n **=** s**.**size**();**

vector**<**int**>** f**(**n**,** 0**);**

**for(**int i **=** 1**,** len **=** 0**;** i **<** n**;)** **{**

**if** **(**s**[**i**]** **==** s**[**len**])** f**[**i**++]** **=** **++**len**;**

**else** **if** **(**len**)** len **=** f**[**len**-**1**];**

**else** f**[**i**++]** **=** 0**;**

**}**

**return** f**[**n**-**1**]** **&&** n **%** **(**n**-**f**[**n**-**1**])** **==** 0**;**

**}**

**};**

class Solution **{**

public**:**

bool repeatedSubstringPattern**(**string s**)** **{**

int n **=** s**.**length**();**

**for** **(**int i **=** 1**;** i **<=** n**/**2**;** i**++)** **{**

**if** **(**n **%** i **!=** 0**)** **continue;**

int cnt **=** n **/** i**;**

string a**,** b **=** s**.**substr**(**0**,** i**);**

**while** **(**cnt**--)** a **+=** b**;**

**if** **(**a **==** s**)** **return** **true;**

**}**

**return** **false;**

**}**

**};**

### 460. LFU Cache

Hard

Design and implement a data structure for [Least Frequently Used (LFU)](https://en.wikipedia.org/wiki/Least_frequently_used) cache. It should support the following operations: get and put.

get(key) - Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.  
put(key, value) - Set or insert the value if the key is not already present. When the cache reaches its capacity, it should invalidate the least frequently used item before inserting a new item. For the purpose of this problem, when there is a tie (i.e., two or more keys that have the same frequency), the least **recently** used key would be evicted.

Note that the number of times an item is used is the number of calls to the get and put functions for that item since it was inserted. This number is set to zero when the item is removed.

**Follow up:**  
Could you do both operations in **O(1)** time complexity?

**Example:**

LFUCache cache = new LFUCache( 2 /\* capacity \*/ );

cache.put(1, 1);

cache.put(2, 2);

cache.get(1); // returns 1

cache.put(3, 3); // evicts key 2

cache.get(2); // returns -1 (not found)

cache.get(3); // returns 3.

cache.put(4, 4); // evicts key 1.

cache.get(1); // returns -1 (not found)

cache.get(3); // returns 3

cache.get(4); // returns 4

### 461. Hamming Distance

Easy

The [Hamming distance](https://en.wikipedia.org/wiki/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.

class Solution **{**

public**:**

int hammingDistance**(**int x**,** int y**)** **{**

//return bitset<32>(x^y).count();

int res **=** 0**;**

**for** **(**int i **=** 0**;** i **<** 32**;** i**++)** **{**

**if** **((**x**&**1**)** **!=** **(**y**&**1**))** res**++;**

x **>>=** 1**;**

y **>>=** 1**;**

**}**

**return** res**;**

**}**

**};**

### 462. Minimum Moves to Equal Array Elements II

Medium

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

You may assume the array's length is at most 10,000.

**Example:**

**Input:**

[1,2,3]

**Output:**

2

**Explanation:**

Only two moves are needed (remember each move increments or decrements one element):

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

class Solution **{**

public**:**

int minMoves2**(**vector**<**int**>&** nums**)** **{**

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

long long med **=** nums**[**nums**.**size**()/**2**],** res **=** 0**;**

**for** **(**const auto **&**i **:** nums**)** res **+=** abs**(**i**-**med**);**

**return** res**;**

**}**

**};**

### 463. Island Perimeter

Easy

You are given a map in form of a two-dimensional integer grid where 1 represents land and 0 represents water.

Grid cells are connected horizontally/vertically (not diagonally). The grid is completely surrounded by water, and there is exactly one island (i.e., one or more connected land cells).

The island doesn't have "lakes" (water inside that isn't connected to the water around the island). One cell is a square with side length 1. The grid is rectangular, width and height don't exceed 100. Determine the perimeter of the island.

**Example:**

**Input:**

[[0,1,0,0],

[1,1,1,0],

[0,1,0,0],

[1,1,0,0]]

**Output:** 16

**Explanation:** The perimeter is the 16 yellow stripes in the image below:



class Solution **{**

public**:**

int islandPerimeter**(**vector**<**vector**<**int**>>&** grid**)** **{**

int n **=** grid**.**size**(),** m **=** grid**[**0**].**size**();**

int res **=** 0**;**

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

**for** **(**int j **=** 0**;** j **<** m**;** j**++)** **{**

**if** **(**grid**[**i**][**j**]** **==** 1**)** **{**

res **+=** 4**;**

**if** **(**i **&&** grid**[**i**-**1**][**j**]** **==** 1**)** res **-=** 2**;**

**if** **(**j **&&** grid**[**i**][**j**-**1**]** **==** 1**)** res **-=** 2**;**

**}**

**}**

**}**

**return** res**;**

**}**

**};**

### 464. Can I Win

Medium

In the "100 game," two players take turns adding, to a running total, any integer from 1..10. The player who first causes the running total to reach or exceed 100 wins.

What if we change the game so that players cannot re-use integers?

For example, two players might take turns drawing from a common pool of numbers of 1..15 without replacement until they reach a total >= 100.

Given an integer maxChoosableInteger and another integer desiredTotal, determine if the first player to move can force a win, assuming both players play optimally.

You can always assume that maxChoosableInteger will not be larger than 20 and desiredTotal will not be larger than 300.

**Example**

**Input:**

maxChoosableInteger = 10

desiredTotal = 11

**Output:**

false

**Explanation:**

No matter which integer the first player choose, the first player will lose.

The first player can choose an integer from 1 up to 10.

If the first player choose 1, the second player can only choose integers from 2 up to 10.

The second player will win by choosing 10 and get a total = 11, which is >= desiredTotal.

Same with other integers chosen by the first player, the second player will always win.

### 466. Count The Repetitions

Hard

Define S = [s,n] as the string S which consists of n connected strings s. For example, ["abc", 3] ="abcabcabc".

On the other hand, we define that string s1 can be obtained from string s2 if we can remove some characters from s2 such that it becomes s1. For example, “abc” can be obtained from “abdbec” based on our definition, but it can not be obtained from “acbbe”.

You are given two non-empty strings s1 and s2 (each at most 100 characters long) and two integers 0 ≤ n1 ≤ 106 and 1 ≤ n2 ≤ 106. Now consider the strings S1 and S2, where S1=[s1,n1] and S2=[s2,n2]. Find the maximum integer M such that [S2,M] can be obtained from S1.

**Example:**

Input:

s1="acb", n1=4

s2="ab", n2=2

Return:

2

### 467. Unique Substrings in Wraparound String

Medium

Consider the string s to be the infinite wraparound string of "abcdefghijklmnopqrstuvwxyz", so s will look like this: "...zabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcd....".

Now we have another string p. Your job is to find out how many unique non-empty substrings of p are present in s. In particular, your input is the string p and you need to output the number of different non-empty substrings of p in the string s.

**Note:** p consists of only lowercase English letters and the size of p might be over 10000.

**Example 1:**

**Input:** "a"

**Output:** 1

**Explanation:** Only the substring "a" of string "a" is in the string s.

**Example 2:**

**Input:** "cac"

**Output:** 2

**Explanation:** There are two substrings "a", "c" of string "cac" in the string s.

**Example 3:**

**Input:** "zab"

**Output:** 6

**Explanation:** There are six substrings "z", "a", "b", "za", "ab", "zab" of string "zab" in the string s.

### 468. Validate IP Address

Medium

Write a function to check whether an input string is a valid IPv4 address or IPv6 address or neither.

**IPv4** addresses are canonically represented in dot-decimal notation, which consists of four decimal numbers, each ranging from 0 to 255, separated by dots ("."), e.g.,172.16.254.1;

Besides, leading zeros in the IPv4 is invalid. For example, the address 172.16.254.01 is invalid.

**IPv6** addresses are represented as eight groups of four hexadecimal digits, each group representing 16 bits. The groups are separated by colons (":"). For example, the address 2001:0db8:85a3:0000:0000:8a2e:0370:7334 is a valid one. Also, we could omit some leading zeros among four hexadecimal digits and some low-case characters in the address to upper-case ones, so 2001:db8:85a3:0:0:8A2E:0370:7334 is also a valid IPv6 address(Omit leading zeros and using upper cases).

However, we don't replace a consecutive group of zero value with a single empty group using two consecutive colons (::) to pursue simplicity. For example, 2001:0db8:85a3::8A2E:0370:7334 is an invalid IPv6 address.

Besides, extra leading zeros in the IPv6 is also invalid. For example, the address 02001:0db8:85a3:0000:0000:8a2e:0370:7334 is invalid.

**Note:** You may assume there is no extra space or special characters in the input string.

**Example 1:**

**Input:** "172.16.254.1"

**Output:** "IPv4"

**Explanation:** This is a valid IPv4 address, return "IPv4".

**Example 2:**

**Input:** "2001:0db8:85a3:0:0:8A2E:0370:7334"

**Output:** "IPv6"

**Explanation:** This is a valid IPv6 address, return "IPv6".

**Example 3:**

**Input:** "256.256.256.256"

**Output:** "Neither"

**Explanation:** This is neither a IPv4 address nor a IPv6 address.

class Solution **{**

public**:**

string validIPAddress**(**string IP**)** **{**

string s**;**

int cnt **=** 0**;**

**if** **(**IP**.**find**(**'.'**)** **!=** string**::**npos**)** **{**

**for** **(**auto **&**c **:** IP**)** **{**

**if** **(**c **==** '.' **&&** **++**cnt**)** c **=** ' '**;**

**else** **if** **(!**isdigit**(**c**))** **return** "Neither"**;**

**}**

**if** **(**cnt **!=** 3**)** **return** "Neither"**;**

stringstream ss**(**IP**);**

**while(**ss **>>** s**)** **{**

**--**cnt**;**

**if** **(**s**.**empty**()** **||** s**.**length**()** **>** 3 **||** stoi**(**s**)** **>** 255**)** **{**

**return** "Neither"**;**

**}** **else** **if** **(**s**[**0**]** **==** '0' **&&** s **!=** "0"**)** **{**

**return** "Neither"**;**

**}**

**}**

**return** cnt **==** **-**1 **?** "IPv4" **:** "Neither"**;**

**}** **else** **{**

**for** **(**auto **&**c **:** IP**)** **{**

**if** **(**c **==** ':' **&&** **++**cnt**)** c **=** ' '**;**

**else** **if** **(!**isalnum**(**c**))** **return** "Neither"**;**

**else** **if** **(**isalpha**(**c**)&&(**isupper**(**c**)?**c **>** 'F'**:**c **>** 'f'**))**

**return** "Neither"**;**

**}**

**if** **(**cnt **!=** 7**)** **return** "Neither"**;**

stringstream ss**(**IP**);**

**while(**ss **>>** s**)** **{**

**--**cnt**;**

**if** **(**s**.**empty**()** **||** s**.**length**()** **>** 4**)** **{**

**return** "Neither"**;**

**}**

**}**

**return** cnt **==** **-**1 **?** "IPv6" **:** "Neither"**;**

**}**

**return** "Neither"**;**

**}**

**};**

### 470. Implement Rand10() Using Rand7()

Medium

Given a function rand7 which generates a uniform random integer in the range 1 to 7, write a function rand10 which generates a uniform random integer in the range 1 to 10.

Do NOT use system's Math.random().

**Example 1:**

**Input:** 1

**Output:** [7]

**Example 2:**

**Input:** 2

**Output:** [8,4]

**Example 3:**

**Input:** 3

**Output:** [8,1,10]

**Note:**

1. rand7 is predefined.
2. Each testcase has one argument: n, the number of times that rand10 is called.

**Follow up:**

1. What is the [expected value](https://en.wikipedia.org/wiki/Expected_value) for the number of calls to rand7() function?
2. Could you minimize the number of calls to rand7()?

// The rand7() API is already defined for you.

// int rand7();

// @return a random integer in the range 1 to 7

class Solution **{**

public**:**

int rand10**()** **{**

**while** **(**1**)** **{**

int num **=** 7 **\*** **(**rand7**()** **-** 1**)** **+** **(**rand7**()** **-** 1**);**

int ret **=** 1 **+** num**/**4**;**

**if** **(**ret **<** 11**)** **return** ret**;**

**}**

**}**

**};**

### 472. Concatenated Words

Hard

Given a list of words (**without duplicates**), please write a program that returns all concatenated words in the given list of words.

A concatenated word is defined as a string that is comprised entirely of at least two shorter words in the given array.

**Example:**

**Input:** ["cat","cats","catsdogcats","dog","dogcatsdog","hippopotamuses","rat","ratcatdogcat"]

**Output:** ["catsdogcats","dogcatsdog","ratcatdogcat"]

**Explanation:** "catsdogcats" can be concatenated by "cats", "dog" and "cats";   
 "dogcatsdog" can be concatenated by "dog", "cats" and "dog";   
"ratcatdogcat" can be concatenated by "rat", "cat", "dog" and "cat".

**Note:**

1. The number of elements of the given array will not exceed 10,000
2. The length sum of elements in the given array will not exceed 600,000.
3. All the input string will only include lower case letters.
4. The returned elements order does not matter.

### 473. Matchsticks to Square

Medium

Remember the story of Little Match Girl? By now, you know exactly what matchsticks the little match girl has, please find out a way you can make one square by using up all those matchsticks. You should not break any stick, but you can link them up, and each matchstick must be used **exactly** one time.

Your input will be several matchsticks the girl has, represented with their stick length. Your output will either be true or false, to represent whether you could make one square using all the matchsticks the little match girl has.

**Example 1:**

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

**Output:** true

**Explanation:** You can form a square with length 2, one side of the square came two sticks with length 1.

**Example 2:**

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

**Output:** false

**Explanation:** You cannot find a way to form a square with all the matchsticks.

**Note:**

1. The length sum of the given matchsticks is in the range of 0 to 10^9.
2. The length of the given matchstick array will not exceed 15.

class Solution **{**

public**:**

bool makesquare**(**vector**<**int**>&** nums**)** **{**

**if** **(**nums**.**size**()** **<** 4**)** **return** **false;**

int sum **=** accumulate**(**nums**.**begin**(),** nums**.**end**(),** 0**);**

**if** **(**sum **%** 4 **!=** 0**)** **return** **false;**

int n **=** nums**.**size**(),** all **=** **(**1 **<<** n**)** **-** 1**,** target **=** sum **/** 4**;**

vector**<**int**>** masks**,** validHalf**(**1 **<<** n**,** **false);**

**for** **(**int i **=** 0**;** i **<=** all**;** **++**i**)** **{**

int curSum **=** 0**;**

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

**if** **((**i **>>** j**)** **&** 1**)** curSum **+=** nums**[**j**];**

**}**

**if** **(**curSum **==** target**)** **{**

**for** **(**auto **&**mask **:** masks**)** **{**

**if** **((**mask **&** i**)** **!=** 0**)** **continue;**

int half **=** mask **|** i**;**

validHalf**[**half**]** **=** **true;**

**if** **(**validHalf**[**all **^** half**])** **return** **true;**

**}**

masks**.**push\_back**(**i**);**

**}**

**}**

**return** **false;**

**}**

**};**

class Solution **{**

public**:**

bool makesquare**(**vector**<**int**>&** nums**)** **{**

**if** **(**nums**.**size**()** **<** 4**)** **return** **false;**

int sum **=** accumulate**(**nums**.**begin**(),** nums**.**end**(),** 0**);**

**if** **(**sum **%** 4 **!=** 0**)** **return** **false;**

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

**return** dfs**(**nums**,** nums**.**size**()-**1**,** vector**<**int**>** **(**4**,** sum**/**4**));**

**}**

private**:**

bool dfs**(**vector**<**int**>** **&**nums**,** int cur**,** vector**<**int**>** lens**)** **{**

**if** **(**cur **<** 0**)** **return** **true;**

**for** **(**int i **=** 0**;** i **<** 4**;** **++**i**)** **{**

**if** **(**lens**[**i**]** **-** nums**[**cur**]** **>=** 0**)** **{**

lens**[**i**]** **-=** nums**[**cur**];**

**if** **(**dfs**(**nums**,** cur**-**1**,** lens**))** **return** **true;**

lens**[**i**]** **+=** nums**[**cur**];**

**}**

**}**

**return** **false;**

**}**

**};**

### 474. Ones and Zeroes

Medium

In the computer world, use restricted resource you have to generate maximum benefit is what we always want to pursue.

For now, suppose you are a dominator of **m** 0s and **n** 1s respectively. On the other hand, there is an array with strings consisting of only 0s and 1s.

Now your task is to find the maximum number of strings that you can form with given **m** 0s and **n** 1s. Each 0 and 1 can be used at most **once**.

**Note:**

1. The given numbers of 0s and 1s will both not exceed 100
2. The size of given string array won't exceed 600.

**Example 1:**

**Input:** Array = {"10", "0001", "111001", "1", "0"}, m = 5, n = 3

**Output:** 4

**Explanation:** This are totally 4 strings can be formed by the using of 5 0s and 3 1s, which are “10,”0001”,”1”,”0”

**Example 2:**

**Input:** Array = {"10", "0", "1"}, m = 1, n = 1

**Output:** 2

**Explanation:** You could form "10", but then you'd have nothing left. Better form "0" and "1".

class Solution **{**

public**:**

int findMaxForm**(**vector**<**string**>&** strs**,** int m**,** int n**)** **{**

vector**<**vector**<**int**>>** dp**(**m**+**1**,**vector**<**int**>(**n**+**1**,** 0**));**

**for** **(**auto **&**s**:** strs**)** **{**

int ones **=** count**(**s**.**begin**(),** s**.**end**(),** '1'**),** zeros **=** s**.**size**()-**ones**;**

**for** **(**int i **=** m**;** i **>=** zeros**;** i**--)**

**for** **(**int j **=** n**;** j **>=** ones**;** j**--)**

dp**[**i**][**j**]** **=** max**(**dp**[**i**][**j**],** dp**[**i**-**zeros**][**j**-**ones**]** **+** 1**);**

**}**

**return** dp**[**m**][**n**];**

**}**

**};**

### 475. Heaters

Easy

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.

class Solution **{**

public**:**

int findRadius**(**vector**<**int**>&** houses**,** vector**<**int**>&** heaters**)** **{**

sort**(**heaters**.**begin**(),** heaters**.**end**());**

int minRadius **=** 0**,** n **=** houses**.**size**();**

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

auto larger **=** lower\_bound**(**heaters**.**begin**(),** heaters**.**end**(),** houses**[**i**]);**

int curRadius **=** INT\_MAX**;**

**if** **(**larger **!=** heaters**.**end**())** curRadius **=** **\***larger **-** houses**[**i**];**

**if** **(**larger **!=** heaters**.**begin**())** **{**

auto smaller **=** larger **-** 1**;**

curRadius **=** min**(**curRadius**,** houses**[**i**]** **-** **\***smaller**);**

**}**

minRadius **=** max**(**minRadius**,** curRadius**);**

**}**

**return** minRadius**;**

**}**

**};**

### 476. Number Complement

Easy

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.

class Solution **{**

public**:**

int findComplement**(**int num**)** **{**

**for(**unsigned i **=** 1**;** i **<=** num**;** i **<<=** 1**)** num **^=** i**;**

**return** num**;**

**}**

**};**

class Solution **{**

public**:**

int findComplement**(**int num**)** **{**

int cnt **=** 0**,** n **=** num**;**

**while** **(**n**)** **{**

n **>>=** 1**;**

cnt**++;**

**}**

**return** pow(2, cnt) **-** num **-** 1**;**

**}**

**};**

### 477. Total Hamming Distance

Medium

The [Hamming distance](https://en.wikipedia.org/wiki/Hamming_distance) between two integers is the number of positions at which the corresponding bits are different.

Now your job is to find the total Hamming distance between all pairs of the given numbers.

**Example:**

**Input:** 4, 14, 2

**Output:** 6

**Explanation:** In binary representation, the 4 is 0100, 14 is 1110, and 2 is 0010 (just

showing the four bits relevant in this case). So the answer will be:

HammingDistance(4, 14) + HammingDistance(4, 2) + HammingDistance(14, 2) = 2 + 2 + 2 = 6.

**Note:**

1. Elements of the given array are in the range of 0 to 10^9
2. Length of the array will not exceed 10^4.

class Solution **{**

public**:**

int totalHammingDistance**(**vector**<**int**>&** nums**)** **{**

int res **=** 0**,** n **=** nums**.**size**();**

**for** **(**int i **=** 0**;** i **<** 32**;** **++**i**)** **{**

int cnt **=** 0**;**

**for** **(**auto **&**num **:** nums**)** **{**

**if** **(**num **&** **(**1 **<<** i**))** **++**cnt**;**

**}**

res **+=** cnt **\*** **(**n **-** cnt**);**

**}**

**return** res**;**

**}**

**};**

### 478. Generate Random Point in a Circle

Medium

110186FavoriteShare

Given the radius and x-y positions of the center of a circle, write a function randPoint which generates a uniform random point in the circle.

Note:

1. input and output values are in [floating-point](https://www.webopedia.com/TERM/F/floating_point_number.html).
2. radius and x-y position of the center of the circle is passed into the class constructor.
3. a point on the circumference of the circle is considered to be in the circle.
4. randPoint returns a size 2 array containing x-position and y-position of the random point, in that order.

**Example 1:**

**Input:**

["Solution","randPoint","randPoint","randPoint"]

[[1,0,0],[],[],[]]

**Output:** [null,[-0.72939,-0.65505],[-0.78502,-0.28626],[-0.83119,-0.19803]]

**Example 2:**

**Input:**

["Solution","randPoint","randPoint","randPoint"]

[[10,5,-7.5],[],[],[]]

**Output:** [null,[11.52438,-8.33273],[2.46992,-16.21705],[11.13430,-12.42337]]

**Explanation of Input Syntax:**

The input is two lists: the subroutines called and their arguments. Solution's constructor has three arguments, the radius, x-position of the center, and y-position of the center of the circle. randPoint has no arguments. Arguments are always wrapped with a list, even if there aren't any.

class Solution **{**

public**:**

Solution**(**double r**,** double x**,** double y**)** **:**r**(**r**),** x**(**x**),** y**(**y**)** **{}**

vector**<**double**>** randPoint**()** **{**

double xx**,** yy**;**

**do** **{**

xx **=** 2**\***rand1**()** **-** 1**;**

yy **=** 2**\***rand1**()** **-** 1**;**

**}** **while** **(**xx**\***xx **+** yy**\***yy **>** 1**);**

**return** vector**<**double**>** **{**r**\***xx **+** x**,** r**\***yy **+** y**};**

**}**

private**:**

double r**,** x**,** y**;**

double rand1**()** **{return** **(**double**)**rand**()** **/** RAND\_MAX**;}**

**};**

class Solution **{**

public**:**

Solution**(**double r**,** double x**,** double y**)** **:**r**(**r**),** x**(**x**),** y**(**y**)** **{}**

vector**<**double**>** randPoint**()** **{**

double theta **=** 2 **\*** PI **\*** rand1**();**

double rr **=** sqrt**(**rand1**());**

**return** vector**<**double**>** **{**x **+** rr**\***r**\***cos**(**theta**),**y **+** rr**\***r**\***sin**(**theta**)};**

**}**

private**:**

const double PI **=** 3.14159265358979732384626433832795**;**

double r**,** x**,** y**;**

double rand1**()** **{return** **(**double**)**rand**()** **/** RAND\_MAX**;}**

**};**

### 479. Largest Palindrome Product

Hard

741247FavoriteShare

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].

### 480. Sliding Window Median

Hard

Median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value. So the median is the mean of the two middle value.

Examples:

[2,3,4] , the median is 3

[2,3], the median is (2 + 3) / 2 = 2.5

Given an array *nums*, there is a sliding window of size *k* which is moving from the very left of the array to the very right. You can only see the *k* numbers in the window. Each time the sliding window moves right by one position. Your job is to output the median array for each window in the original array.

For example,  
Given *nums* = [1,3,-1,-3,5,3,6,7], and *k* = 3.

Window position Median

--------------- -----

[1 3 -1] -3 5 3 6 7 1

1 [3 -1 -3] 5 3 6 7 -1

1 3 [-1 -3 5] 3 6 7 -1

1 3 -1 [-3 5 3] 6 7 3

1 3 -1 -3 [5 3 6] 7 5

1 3 -1 -3 5 [3 6 7] 6

Therefore, return the median sliding window as [1,-1,-1,3,5,6].

**Note:**   
You may assume k is always valid, ie: k is always smaller than input array's size for non-empty array.

### 481. Magical String

Medium

A magical string **S** consists of only '1' and '2' and obeys the following rules:

The string **S** is magical because concatenating the number of contiguous occurrences of characters '1' and '2' generates the string **S** itself.

The first few elements of string **S** is the following: **S** = "1221121221221121122……"

If we group the consecutive '1's and '2's in **S**, it will be:

1 22 11 2 1 22 1 22 11 2 11 22 ......

and the occurrences of '1's or '2's in each group are:

1 2 2 1 1 2 1 2 2 1 2 2 ......

You can see that the occurrence sequence above is the **S** itself.

Given an integer N as input, return the number of '1's in the first N number in the magical string **S**.

**Note:** N will not exceed 100,000.

**Example 1:**

**Input:** 6

**Output:** 3

**Explanation:** The first 6 elements of magical string S is "12211" and it contains three 1's, so return 3.

### 482. License Key Formatting

Easy

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.

class Solution **{**

public**:**

string licenseKeyFormatting**(**string S**,** int K**)** **{**

string res**;**

int cnt **=** 0**;**

**for** **(**auto it **=** S**.**rbegin**();** it **!=** S**.**rend**();** it**++)** **if** **(\***it **!=** '-'**)** **{**

**if** **(**cnt**++** **%** K **==** 0 **&&** cnt **!=** 1**)** res **+=** '-'**;**

res **+=** toupper**(\***it**);**

**}**

reverse**(**res**.**begin**(),** res**.**end**());**

**return** res**;**

**}**

**};**

### 483. Smallest Good Base

Hard

For an integer n, we call k>=2 a ***good base*** of n, if all digits of n base k are 1.

Now given a string representing n, you should return the smallest good base of n in string format.

**Example 1:**

**Input:** "13"

**Output:** "3"

**Explanation:** 13 base 3 is 111.

**Example 2:**

**Input:** "4681"

**Output:** "8"

**Explanation:** 4681 base 8 is 11111.

**Example 3:**

**Input:** "1000000000000000000"

**Output:** "999999999999999999"

**Explanation:** 1000000000000000000 base 999999999999999999 is 11.

**Note:**

1. The range of n is [3, 10^18].
2. The string representing n is always valid and will not have leading zeros.

### 485. Max Consecutive Ones

Easy

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

class Solution **{**

public**:**

int findMaxConsecutiveOnes**(**vector**<**int**>&** nums**)** **{**

int cnt **=** 0**,** res **=** 0**;**

**for** **(**const auto **&**i **:** nums**)** **{**

**if** **(**i**)** res **=** max**(**res**,** **++**cnt**);**

**else** cnt **=** 0**;**

**}**

**return** res**;**

**}**

**};**

### 486. Predict the Winner

Medium

Given an array of scores that are non-negative integers. Player 1 picks one of the numbers from either end of the array followed by the player 2 and then player 1 and so on. Each time a player picks a number, that number will not be available for the next player. This continues until all the scores have been chosen. The player with the maximum score wins.

Given an array of scores, predict whether player 1 is the winner. You can assume each player plays to maximize his score.

**Example 1:**

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

**Output:** False

**Explanation:** Initially, player 1 can choose between 1 and 2.   
If he chooses 2 (or 1), then player 2 can choose from 1 (or 2) and 5. If player 2 chooses 5, then player 1 will be left with 1 (or 2).   
So, final score of player 1 is 1 + 2 = 3, and player 2 is 5.   
Hence, player 1 will never be the winner and you need to return False.

**Example 2:**

**Input:** [1, 5, 233, 7]

**Output:** True

**Explanation:** Player 1 first chooses 1. Then player 2 have to choose between 5 and 7. No matter which number player 2 choose, player 1 can choose 233.  
Finally, player 1 has more score (234) than player 2 (12), so you need to return True representing player1 can win.

**Note:**

1. 1 <= length of the array <= 20.
2. Any scores in the given array are non-negative integers and will not exceed 10,000,000.
3. If the scores of both players are equal, then player 1 is still the winner.

### 488. Zuma Game

Hard

Think about Zuma Game. You have a row of balls on the table, colored red(R), yellow(Y), blue(B), green(G), and white(W). You also have several balls in your hand.

Each time, you may choose a ball in your hand, and insert it into the row (including the leftmost place and rightmost place). Then, if there is a group of 3 or more balls in the same color touching, remove these balls. Keep doing this until no more balls can be removed.

Find the minimal balls you have to insert to remove all the balls on the table. If you cannot remove all the balls, output -1.

**Examples:**

**Input:** "WRRBBW", "RB"

**Output:** -1

**Explanation:** WRRBBW -> WRR[R]BBW -> WBBW -> WBB[B]W -> WW

**Input:** "WWRRBBWW", "WRBRW"

**Output:** 2

**Explanation:** WWRRBBWW -> WWRR[R]BBWW -> WWBBWW -> WWBB[B]WW -> WWWW -> empty

**Input:**"G", "GGGGG"

**Output:** 2

**Explanation:** G -> G[G] -> GG[G] -> empty

**Input:** "RBYYBBRRB", "YRBGB"

**Output:** 3

**Explanation:** RBYYBBRRB -> RBYY[Y]BBRRB -> RBBBRRB -> RRRB -> B -> B[B] -> BB[B] -> empty

**Note:**

1. You may assume that the initial row of balls on the table won’t have any 3 or more consecutive balls with the same color.
2. The number of balls on the table won't exceed 20, and the string represents these balls is called "board" in the input.
3. The number of balls in your hand won't exceed 5, and the string represents these balls is called "hand" in the input.
4. Both input strings will be non-empty and only contain characters 'R','Y','B','G','W'.

### 491. Increasing Subsequences

Medium

Given an integer array, your task is to find all the different possible increasing subsequences of the given array, and the length of an increasing subsequence should be at least 2.

**Example:**

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

**Output:** [[4, 6], [4, 7], [4, 6, 7], [4, 6, 7, 7], [6, 7], [6, 7, 7], [7,7], [4,7,7]]

**Note:**

1. The length of the given array will not exceed 15.
2. The range of integer in the given array is [-100,100].
3. The given array may contain duplicates, and two equal integers should also be considered as a special case of increasing sequence.

class Solution **{**

public**:**

vector**<**vector**<**int**>>** findSubsequences**(**vector**<**int**>&** nums**)** **{**

vector**<**vector**<**int**>>** res**;**

vector**<**int**>** path**;**

dfs**(**res**,** path**,** nums**,** 0**);**

**return** res**;**

**}**

void dfs**(**vector**<**vector**<**int**>>** **&**res**,** vector**<**int**>** **&**path**,** vector**<**int**>&** nums**,** int cur**)** **{**

**if** **(**path**.**size**()** **>** 1**)** res**.**push\_back**(**path**);**

unordered\_set**<**int**>** hash**;**

**for** **(**int i **=** cur**;** i **<** nums**.**size**();** **++**i**)** **{**

**if** **((**path**.**empty**()** **||** nums**[**i**]** **>=** path**.**back**())** **&&** **!**hash**.**count**(**nums**[**i**]))** **{**

path**.**push\_back**(**nums**[**i**]);**

dfs**(**res**,** path**,** nums**,** i **+** 1**);**

path**.**pop\_back**();**

hash**.**insert**(**nums**[**i**]);**

**}**

**}**

**}**

**};**

### 492. Construct the Rectangle

Easy

For a web developer, it is very important to know how to design a web page's size. So, given a specific rectangular web page’s area, your job by now is to design a rectangular web page, whose length L and width W satisfy the following requirements:

1. The area of the rectangular web page you designed must equal to the given target area.

2. The width W should not be larger than the length L, which means L >= W.

3. The difference between length L and width W should be as small as possible.

You need to output the length L and the width W of the web page you designed in sequence.

**Example:**

**Input:** 4

**Output:** [2, 2]

**Explanation:** The target area is 4, and all the possible ways to construct it are [1,4], [2,2], [4,1].

But according to requirement 2, [1,4] is illegal; according to requirement 3, [4,1] is not optimal compared to [2,2]. So the length L is 2, and the width W is 2.

**Note:**

1. The given area won't exceed 10,000,000 and is a positive integer
2. The web page's width and length you designed must be positive integers.

class Solution **{**

public**:**

vector**<**int**>** constructRectangle**(**int area**)** **{**

int W **=** sqrt**(**area**);**

**while** **(**1**)** **{**

**if** **(**area **%** W **==** 0**)** **return** **{**area**/**W**,** W**};**

**else** W**--;**

**}**

**return** **{**area**,** 1**};**

**}**

**};**

### 493. Reverse Pairs

Hard

Given an array nums, we call (i, j) an ***important reverse pair*** if i < j and nums[i] > 2\*nums[j].

You need to return the number of important reverse pairs in the given array.

**Example1:**

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

**Output**: 2

**Example2:**

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

**Output**: 3

**Note:**

1. The length of the given array will not exceed 50,000.
2. All the numbers in the input array are in the range of 32-bit integer.

### 494. Target Sum

Medium

You are given a list of non-negative integers, a1, a2, ..., an, and a target, S. Now you have 2 symbols + and -. For each integer, you should choose one from + and - as its new symbol.

Find out how many ways to assign symbols to make sum of integers equal to target S.

**Example 1:**

**Input:** nums is [1, 1, 1, 1, 1], S is 3.

**Output:** 5

**Explanation:**

-1+1+1+1+1 = 3

+1-1+1+1+1 = 3

+1+1-1+1+1 = 3

+1+1+1-1+1 = 3

+1+1+1+1-1 = 3

There are 5 ways to assign symbols to make the sum of nums be target 3.

**Note:**

1. The length of the given array is positive and will not exceed 20.
2. The sum of elements in the given array will not exceed 1000.
3. Your output answer is guaranteed to be fitted in a 32-bit integer.

class Solution **{**

public**:**

int findTargetSumWays**(**vector**<**int**>&** nums**,** int S**)** **{**

int sum **=** 0**,** n **=** nums**.**size**();**

**for** **(**auto **&**i **:** nums**)** sum **+=** i**;**

**if** **((**sum**-**S**)** **%** 2 **!=** 0 **||** S **>** sum**)** **return** 0**;**

int newS **=** **(**sum **+** S**)** **/** 2**;**

vector**<**int**>** dp**(**newS **+** 1**,** 0**);**

dp**[**0**]** **=** 1**;**

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

**for** **(**int j **=** newS**;** j **>=** nums**[**i**];** **--**j**)** **{**

dp**[**j**]** **+=** dp**[**j **-** nums**[**i**]];**

**}**

**}**

**return** dp**[**newS**];**

**}**

**};**

class Solution **{**

public**:**

int findTargetSumWays**(**vector**<**int**>&** nums**,** int S**)** **{**

**return** dfs**(**nums**,** 0**,** S**);**

**}**

private**:**

int dfs**(**vector**<**int**>&** nums**,** int cur**,** long S**)** **{**

**if** **(**cur **==** nums**.**size**())** **return** S **==** 0 **?** 1 **:** 0**;**

**return** dfs**(**nums**,** cur**+**1**,** S**+**nums**[**cur**])**

**+** dfs**(**nums**,** cur**+**1**,** S**-**nums**[**cur**]);**

**}**

**};**

### 495. Teemo Attacking

Medium

In LOL world, there is a hero called Teemo and his attacking can make his enemy Ashe be in poisoned condition. Now, given the Teemo's attacking **ascending** time series towards Ashe and the poisoning time duration per Teemo's attacking, you need to output the total time that Ashe is in poisoned condition.

You may assume that Teemo attacks at the very beginning of a specific time point, and makes Ashe be in poisoned condition immediately.

**Example 1:**

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

**Output:** 4

**Explanation:** At time point 1, Teemo starts attacking Ashe and makes Ashe be poisoned immediately.

This poisoned status will last 2 seconds until the end of time point 2.

And at time point 4, Teemo attacks Ashe again, and causes Ashe to be in poisoned status for another 2 seconds.

So you finally need to output 4.

**Example 2:**

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

**Output:** 3

**Explanation:** At time point 1, Teemo starts attacking Ashe and makes Ashe be poisoned.

This poisoned status will last 2 seconds until the end of time point 2.

However, at the beginning of time point 2, Teemo attacks Ashe again who is already in poisoned status.

Since the poisoned status won't add up together, though the second poisoning attack will still work at time point 2, it will stop at the end of time point 3.

So you finally need to output 3.

**Note:**

1. You may assume the length of given time series array won't exceed 10000.
2. You may assume the numbers in the Teemo's attacking time series and his poisoning time duration per attacking are non-negative integers, which won't exceed 10,000,000.

class Solution **{**

public**:**

int findPoisonedDuration**(**vector**<**int**>&** timeSeries**,** int duration**)** **{**

int endTime **=** **-**1**,** res **=** 0**;**

**for** **(**auto **&**i **:** timeSeries**)** **{**

**if** **(**i **>** endTime**)** **{**

res **+=** duration**;**

endTime **=** i **+** duration**;**

**}**

**else** **if** **(**i**+**duration **>** endTime**){**

res **+=** i **+** duration **-** endTime**;**

endTime **=** i **+** duration**;**

**}**

**}**

**return** res**;**

**}**

**};**

### 496. Next Greater Element I

Easy

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.

class Solution **{**

public**:**

vector**<**int**>** nextGreaterElement**(**vector**<**int**>&** nums1**,** vector**<**int**>&** nums2**)** **{**

unordered\_map**<**int**,** int**>** m**;**

stack**<**int**>** stk**;**

**for** **(**auto **&**i **:** nums2**)** **{**

**while** **(!**stk**.**empty**()** **&&** stk**.**top**()** **<** i**)** **{**

m**[**stk**.**top**()]** **=** i**;**

stk**.**pop**();**

**}**

stk**.**push**(**i**);**

**}**

**for** **(**auto **&**i **:** nums1**)** **{**

**if** **(**m**.**count**(**i**))** i **=** m**[**i**];**

**else** i **=** **-**1**;**

**}**

**return** nums1**;**

**}**

**};**

### 497. Random Point in Non-overlapping Rectangles

Medium

Given a list of **non-overlapping** axis-aligned rectangles rects, write a function pick which randomly and uniformily picks an **integer point** in the space covered by the rectangles.

Note:

1. An **integer point** is a point that has integer coordinates.
2. A point on the perimeter of a rectangle is **included** in the space covered by the rectangles.
3. ith rectangle = rects[i] = [x1,y1,x2,y2], where [x1, y1] are the integer coordinates of the bottom-left corner, and [x2, y2] are the integer coordinates of the top-right corner.
4. length and width of each rectangle does not exceed 2000.
5. 1 <= rects.length <= 100
6. pick return a point as an array of integer coordinates [p\_x, p\_y]
7. pick is called at most 10000 times.

**Example 1:**

**Input:**

["Solution","pick","pick","pick"]

[[[[1,1,5,5]]],[],[],[]]

**Output:**

[null,[4,1],[4,1],[3,3]]

**Example 2:**

**Input:**

["Solution","pick","pick","pick","pick","pick"]

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

**Output:**

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

**Explanation of Input Syntax:**

The input is two lists: the subroutines called and their arguments. Solution's constructor has one argument, the array of rectangles rects. pick has no arguments. Arguments are always wrapped with a list, even if there aren't any.

### 498. Diagonal Traverse

Medium

Given a matrix of M x N elements (M rows, N columns), return all elements of the matrix in diagonal order as shown in the below image.

**Example:**

**Input:**

[

[ 1, 2, 3 ],

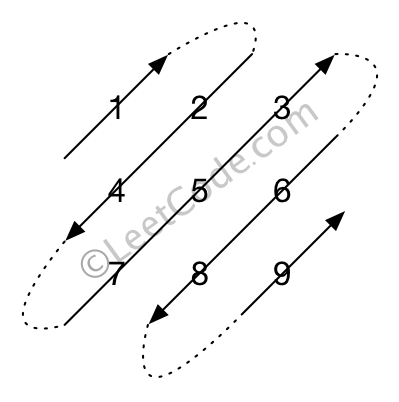
[ 4, 5, 6 ],

[ 7, 8, 9 ]

]

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

**Explanation:**



**Note:**

The total number of elements of the given matrix will not exceed 10,000.

class Solution **{**

public**:**

vector**<**int**>** findDiagonalOrder**(**vector**<**vector**<**int**>>&** matrix**)** **{**

vector**<**int**>** res**;**

**if** **(**matrix**.**empty**())** **return** res**;**

int m **=** matrix**.**size**(),** n **=** matrix**[**0**].**size**(),** flag **=** 0**;**

**for** **(**int k **=** 0**;** k **<** m**+**n**-**1**;** k**++)** **{**

int st **=** max**(**0**,** k**+**1**-**n**),** ed **=** min**(**m**-**1**,** k**);**

**if** **(**flag**)** **for** **(**int i **=** st**;** i **<=** ed**;** i**++)** **{**

res**.**push\_back**(**matrix**[**i**][**k**-**i**]);**

**}**

**else** **for** **(**int i **=** ed**;** i **>=** st**;** i**--)** **{**

res**.**push\_back**(**matrix**[**i**][**k**-**i**]);**

**}**

flag **^=** 1**;**

**}**

**return** res**;**

**}**

**};**

### 500. Keyboard Row

Easy

Given a List of words, return the words that can be typed using letters of **alphabet** on only one row's of American keyboard like the image below.



**Example:**

**Input:** ["Hello", "Alaska", "Dad", "Peace"]

**Output:** ["Alaska", "Dad"]

**Note:**

1. You may use one character in the keyboard more than once.
2. You may assume the input string will only contain letters of alphabet.

class Solution **{**

public**:**

vector**<**string**>** findWords**(**vector**<**string**>&** words**)** **{**

vector**<**int**>** vect**{**1**,**2**,**2**,**1**,**0**,**1**,**1**,**1**,**0**,**1**,**1**,**1**,**2**,**2**,**

0**,**0**,**0**,**0**,**1**,**0**,**0**,**2**,**0**,**2**,**0**,**2**};**

vector**<**string**>** res**;**

**for** **(**auto **&**s **:** words**)** **{**

bool ok **=** **true;**

int r **=** vect**[**tolower**(**s**[**0**])-**'a'**];**

**for** **(**auto c **:** s**)** **{**

**if** **(**vect**[**tolower**(**c**)-**'a'**]** **!=** r**)** **{**

ok **=** **false;**

**break;**

**}**

**}**

**if** **(**ok**)** res**.**push\_back**(**s**);**

**}**

**return** res**;**

**}**

**};**