目录

[352. Data Stream as Disjoint Intervals 2](#_Toc21283930)

[354. Russian Doll Envelopes 3](#_Toc21283931)

[355. Design Twitter 4](#_Toc21283932)

[357. Count Numbers with Unique Digits 6](#_Toc21283933)

[363. Max Sum of Rectangle No Larger Than K 7](#_Toc21283934)

[365. Water and Jug Problem 8](#_Toc21283935)

[367. Valid Perfect Square 9](#_Toc21283936)

[368. Largest Divisible Subset 11](#_Toc21283937)

[371. Sum of Two Integers 13](#_Toc21283938)

[372. Super Pow 14](#_Toc21283939)

[373. Find K Pairs with Smallest Sums 16](#_Toc21283940)

[374. Guess Number Higher or Lower 17](#_Toc21283941)

[375. Guess Number Higher or Lower II 19](#_Toc21283942)

[376. Wiggle Subsequence 20](#_Toc21283943)

[377. Combination Sum IV 22](#_Toc21283944)

[378. Kth Smallest Element in a Sorted Matrix 23](#_Toc21283945)

[380. Insert Delete GetRandom O(1) 25](#_Toc21283946)

[381. Insert Delete GetRandom O(1) - Duplicates allowed 27](#_Toc21283947)

[382. Linked List Random Node 28](#_Toc21283948)

[383. Ransom Note 29](#_Toc21283949)

[384. Shuffle an Array 30](#_Toc21283950)

[385. Mini Parser 31](#_Toc21283951)

[386. Lexicographical Numbers 32](#_Toc21283952)

[387. First Unique Character in a String 33](#_Toc21283953)

[388. Longest Absolute File Path 34](#_Toc21283954)

[389. Find the Difference 36](#_Toc21283955)

[390. Elimination Game 37](#_Toc21283956)

[391. Perfect Rectangle 38](#_Toc21283957)

[392. Is Subsequence 40](#_Toc21283958)

[393. UTF-8 Validation 42](#_Toc21283959)

[394. Decode String 44](#_Toc21283960)

[395. Longest Substring with At Least K Repeating Characters 46](#_Toc21283961)

[396. Rotate Function 47](#_Toc21283962)

[397. Integer Replacement 48](#_Toc21283963)

[398. Random Pick Index 50](#_Toc21283964)

[399. Evaluate Division 52](#_Toc21283965)

[400. Nth Digit 53](#_Toc21283966)

### 352. Data Stream as Disjoint Intervals

Hard

Given a data stream input of non-negative integers a1, a2, ..., an, ..., summarize the numbers seen so far as a list of disjoint intervals.

For example, suppose the integers from the data stream are 1, 3, 7, 2, 6, ..., then the summary will be:

[1, 1]

[1, 1], [3, 3]

[1, 1], [3, 3], [7, 7]

[1, 3], [7, 7]

[1, 3], [6, 7]

**Follow up:**

What if there are lots of merges and the number of disjoint intervals are small compared to the data stream's size?

### 354. Russian Doll Envelopes

Hard

You have a number of envelopes with widths and heights given as a pair of integers (w, h). One envelope can fit into another if and only if both the width and height of one envelope is greater than the width and height of the other envelope.

What is the maximum number of envelopes can you Russian doll? (put one inside other)

**Note:**  
Rotation is not allowed.

**Example:**

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

**Output:** 3

**Explanation: T**he maximum number of envelopes you can Russian doll is 3 ([2,3] => [5,4] => [6,7])

class Solution **{**

public**:**

int maxEnvelopes**(**vector**<**vector**<**int**>>&** envelopes**)** **{**

sort**(**envelopes**.**begin**(),** envelopes**.**end**(),** **[](**vector**<**int**>** **&**a**,** vector**<**int**>** **&**b**){**

**return** a**[**0**]** **<** b**[**0**]** **||** **(**a**[**0**]** **==** b**[**0**]** **&&** a**[**1**]** **>** b**[**1**]);**

**});**

vector**<**int**>** dp**;**

**for** **(**auto **&**v **:** envelopes**)** **{**

auto iter **=** lower\_bound**(**dp**.**begin**(),** dp**.**end**(),** v**[**1**]);**

**if** **(**iter **==** dp**.**end**())** dp**.**push\_back**(**v**[**1**]);**

**else** **if** **(**v**[**1**]** **<** **\***iter**)** **\***iter **=** v**[**1**];**

**}**

**return** dp**.**size**();**

**}**

**};**

### 355. Design Twitter

Medium

583148FavoriteShare

Design a simplified version of Twitter where users can post tweets, follow/unfollow another user and is able to see the 10 most recent tweets in the user's news feed. Your design should support the following methods:

1. **postTweet(userId, tweetId)**: Compose a new tweet.
2. **getNewsFeed(userId)**: Retrieve the 10 most recent tweet ids in the user's news feed. Each item in the news feed must be posted by users who the user followed or by the user herself. Tweets must be ordered from most recent to least recent.
3. **follow(followerId, followeeId)**: Follower follows a followee.
4. **unfollow(followerId, followeeId)**: Follower unfollows a followee.

**Example:**

Twitter twitter = new Twitter();

// User 1 posts a new tweet (id = 5).

twitter.postTweet(1, 5);

// User 1's news feed should return a list with 1 tweet id -> [5].

twitter.getNewsFeed(1);

// User 1 follows user 2.

twitter.follow(1, 2);

// User 2 posts a new tweet (id = 6).

twitter.postTweet(2, 6);

// User 1's news feed should return a list with 2 tweet ids -> [6, 5].

// Tweet id 6 should precede tweet id 5 because it is posted after tweet id 5.

twitter.getNewsFeed(1);

// User 1 unfollows user 2.

twitter.unfollow(1, 2);

// User 1's news feed should return a list with 1 tweet id -> [5],

// since user 1 is no longer following user 2.

twitter.getNewsFeed(1);

### 357. Count Numbers with Unique Digits

Medium

Given a **non-negative** integer n, count all numbers with unique digits, x, where 0 ≤ x < 10n.

**Example:**

**Input:** 2

**Output:** 91

**Explanation:** The answer should be the total numbers in the range of 0 ≤ x < 100,

  excluding 11,22,33,44,55,66,77,88,99

### 363. Max Sum of Rectangle No Larger Than K

Hard

Given a non-empty 2D matrix *matrix* and an integer *k*, find the max sum of a rectangle in the *matrix* such that its sum is no larger than *k*.

**Example:**

**Input:** matrix = [[1,0,1],[0,-2,3]], k = 2

**Output:** 2

**Explanation:** Because the sum of rectangle [[0, 1], [-2, 3]] is 2,

  and 2 is the max number no larger than k (k = 2).

**Note:**

1. The rectangle inside the matrix must have an area > 0.
2. What if the number of rows is much larger than the number of columns?

### 365. Water and Jug Problem

Medium

You are given two jugs with capacities *x* and *y* litres. There is an infinite amount of water supply available. You need to determine whether it is possible to measure exactly *z* litres using these two jugs.

If *z* liters of water is measurable, you must have *z* liters of water contained within **one or both buckets** by the end.

Operations allowed:

* Fill any of the jugs completely with water.
* Empty any of the jugs.
* Pour water from one jug into another till the other jug is completely full or the first jug itself is empty.

**Example 1:** (From the famous [*"Die Hard"* example](https://www.youtube.com/watch?v=BVtQNK_ZUJg))

Input: x = 3, y = 5, z = 4

Output: True

**Example 2:**

Input: x = 2, y = 6, z = 5

Output: False

class Solution **{**

public**:**

bool canMeasureWater**(**int x**,** int y**,** int z**)** **{**

**return** z **==** 0 **||** **(**z **-** x **<=** y **&&** z **%** gcd**(**x**,** y**)** **==** 0**);**

**}**

private**:**

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

**return** y **==** 0 **?** x **:** gcd**(**y**,** x **%** y**);**

**}**

**};**

### 367. Valid Perfect Square

Easy

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

class Solution **{**

public**:**

bool isPerfectSquare**(**int num**)** **{**

long r **=** num**;**

**while** **(**r**\***r **>** num**)**

r **=** **(**r **+** num**/**r**)** **/** 2**;**

**return** r**\***r **==** num**;**

**}**

**};**

class Solution **{**

public**:**

bool isPerfectSquare**(**int num**)** **{**

long long l **=** 0**,** r **=** num**;**

**while** **(**l **<=** r**)** **{**

auto mid **=** l **+** **(**r**-**l**)/**2**;**

auto sqmid **=** mid **\*** mid**;**

**if** **(**sqmid **==** num**)** **return** **true;**

**else** **if** **(**sqmid **<** num**)** l **=** mid **+** 1**;**

**else** r **=** mid **-** 1**;**

**}**

**return** **false;**

**}**

**};**

### 368. Largest Divisible Subset

Medium

Given a set of **distinct** positive integers, find the largest subset such that every pair (Si, Sj) of elements in this subset satisfies:

Si % Sj = 0 or Sj % Si = 0.

If there are multiple solutions, return any subset is fine.

**Example 1:**

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

**Output:** [1,2] (of course, [1,3] will also be ok)

**Example 2:**

**Input:** [1,2,4,8]

**Output:** [1,2,4,8]

class Solution **{**

public**:**

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

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

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

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

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

vector**<**pair**<**int**,** int**>>** dp**(**n**);**

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

dp**[**i**]** **=** **{**1**,** **-**1**};**

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

**if** **(**nums**[**i**]** **%** nums**[**j**]** **==** 0**)** **{**

**if** **(**dp**[**j**].**first**+**1 **>** dp**[**i**].**first**)** **{**

dp**[**i**]** **=** **{**dp**[**j**].**first**+**1**,** j**};**

**}**

**}**

**}**

**if** **(**dp**[**i**].**first **>** dp**[**p**].**first**)** p **=** i**;**

**}**

**while** **(**p **!=** **-**1**)** **{**

res**.**push\_back**(**nums**[**p**]);**

p **=** dp**[**p**].**second**;**

**}**

**return** res**;**

**}**

**};**

### 371. Sum of Two Integers

Easy

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

**Example 1:**

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

**Output:** 3

**Example 2:**

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

**Output:** 1

class Solution **{**

public**:**

int getSum**(**int a**,** int b**)** **{**

**while** **(**b **!=** 0**)** **{**

int carry **=** a **&** b**;**

a **=** a **^** b**;**

b **=** **(**carry **&** 0x7fffffff**)** **<<** 1**;**

**}**

**return** a**;**

**}**

**};**

### 372. Super Pow

Medium

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

**Example 1:**

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

**Output:** 8

**Example 2:**

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

**Output:** 1024

class Solution **{**

public**:**

int superPow**(**int a**,** vector**<**int**>&** b**)** **{**

int pre **=** 1**;**

a **%=** 1337**;**

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

int cur **=** POW**(**a**,** b**[**i**]);**

pre **=** pre**\***cur **%** 1337**;**

a **=** POW**(**a**,** 10**);**

**}**

**return** pre**;**

**}**

private**:**

int POW**(**int x**,** int n**)** **{**

int ret **=** 1**;**

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

ret **=** ret**\***x **%** 1337**;**

**}**

**return** ret**;**

**}**

**};**

### 373. Find K Pairs with Smallest Sums

Medium

You are given two integer arrays **nums1** and **nums2** sorted in ascending order and an integer **k**.

Define a pair **(u,v)** which consists of one element from the first array and one element from the second array.

Find the k pairs **(u1,v1),(u2,v2) ...(uk,vk)** with the smallest sums.

**Example 1:**

**Input:** nums1 = [1,7,11], nums2 = [2,4,6], k = 3

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

**Explanation:** The first 3 pairs are returned from the sequence:

  [1,2],[1,4],[1,6],[7,2],[7,4],[11,2],[7,6],[11,4],[11,6]

**Example 2:**

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

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

**Explanation:** The first 2 pairs are returned from the sequence:

  [1,1],[1,1],[1,2],[2,1],[1,2],[2,2],[1,3],[1,3],[2,3]

**Example 3:**

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

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

**Explanation:** All possible pairs are returned from the sequence: [1,3],[2,3]

### 374. Guess Number Higher or Lower

Easy

We are playing the Guess Game. The game is as follows:

I pick a number from **1** to ***n***. You have to guess which number I picked.

Every time you guess wrong, I'll tell you whether the number is higher or lower.

You call a pre-defined API guess(int num) which returns 3 possible results (-1, 1, or 0):

-1 : My number is lower

1 : My number is higher

0 : Congrats! You got it!

**Example :**

**Input:** n = 10, pick = 6

**Output:** 6

class Solution **{**

public**:**

int guessNumber**(**int n**)** **{**

int l **=** 1**,** r **=** n**;**

**while** **(**l **<=** n**)** **{**

int mid **=** l **+** **(**r**-**l**)/**2**;**

**switch(**guess**(**mid**))** **{**

**case** 0**:** **return** mid**;** **break;**

**case** **-**1**:** r **=** mid**-**1**;** **break;**

**case** 1**:** l **=** mid**+**1**;** **break;**

**}**

**}**

**return** **-**1**;**

**}**

**};**

### 375. Guess Number Higher or Lower II

Medium

We are playing the Guess Game. The game is as follows:

I pick a number from **1** to **n**. You have to guess which number I picked.

Every time you guess wrong, I'll tell you whether the number I picked is higher or lower.

However, when you guess a particular number x, and you guess wrong, you pay **$x**. You win the game when you guess the number I picked.

**Example:**

n = 10, I pick 8.

First round: You guess 5, I tell you that it's higher. You pay $5.

Second round: You guess 7, I tell you that it's higher. You pay $7.

Third round: You guess 9, I tell you that it's lower. You pay $9.

Game over. 8 is the number I picked.

You end up paying $5 + $7 + $9 = $21.

Given a particular **n ≥ 1**, find out how much money you need to have to guarantee a **win**.

### 376. Wiggle Subsequence

Medium

A sequence of numbers is called a **wiggle sequence** if the differences between successive numbers strictly alternate between positive and negative. The first difference (if one exists) may be either positive or negative. A sequence with fewer than two elements is trivially a wiggle sequence.

For example, [1,7,4,9,2,5] is a wiggle sequence because the differences (6,-3,5,-7,3) are alternately positive and negative. In contrast, [1,4,7,2,5] and [1,7,4,5,5] are not wiggle sequences, the first because its first two differences are positive and the second because its last difference is zero.

Given a sequence of integers, return the length of the longest subsequence that is a wiggle sequence. A subsequence is obtained by deleting some number of elements (eventually, also zero) from the original sequence, leaving the remaining elements in their original order.

**Example 1:**

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

**Output:** 6

**Explanation:** The entire sequence is a wiggle sequence.

**Example 2:**

**Input:** [1,17,5,10,13,15,10,5,16,8]

**Output:** 7

**Explanation:** There are several subsequences that achieve this length. One is [1,17,10,13,10,16,8].

**Example 3:**

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

**Output:** 2

**Follow up:**  
Can you do it in O(*n*) time?

class Solution **{**

public**:**

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

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

**if** **(**n **<** 2**)** **return** n**;**

int down **=** 1**,** up **=** 1**;**

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

**if** **(**nums**[**i**]** **>** nums**[**i**-**1**])**

up **=** down **+** 1**;**

**else** **if** **(**nums**[**i**]** **<** nums**[**i**-**1**])**

down **=** up **+** 1**;**

**}**

**return** max**(**down**,** up**);**

**}**

**};**

### 377. Combination Sum IV

Medium

Given an integer array with all positive numbers and no duplicates, find the number of possible combinations that add up to a positive integer target.

**Example:**

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

***target*** = 4

The possible combination ways are:

(1, 1, 1, 1)

(1, 1, 2)

(1, 2, 1)

(1, 3)

(2, 1, 1)

(2, 2)

(3, 1)

Note that different sequences are counted as different combinations.

Therefore the output is ***7***.

**Follow up:**  
What if negative numbers are allowed in the given array?  
How does it change the problem?  
What limitation we need to add to the question to allow negative numbers?

**Credits:**  
Special thanks to [@pbrother](https://leetcode.com/pbrother/) for adding this problem and creating all test cases.

### 378. Kth Smallest Element in a Sorted Matrix

Medium

147692FavoriteShare

Given a *n* x *n* matrix where each of the rows and columns are sorted in ascending order, find the kth smallest element in the matrix.

Note that it is the kth smallest element in the sorted order, not the kth distinct element.

**Example:**

matrix = [

[ 1, 5, 9],

[10, 11, 13],

[12, 13, 15]

],

k = 8,

return 13.

**Note:**   
You may assume k is always valid, 1 ≤ k ≤ n2.

class Solution **{**

public**:**

struct node**{**

int x**,** y**,** value**;**

node**(**int x**,** int y**,** int value**):**x**(**x**),** y**(**y**),** value**(**value**){}**

bool **operator** **<** **(**const node **&**rhs**)** const**{**

**return** value **>** rhs**.**value**;**

**}**

**};**

int kthSmallest**(**vector**<**vector**<**int**>>&** matrix**,** int k**)** **{**

const int dx**[]** **=** **{**1**,** 0**};**

const int dy**[]** **=** **{**0**,** 1**};**

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

priority\_queue**<**node**>** pq**;**

pq**.**push**({**0**,** 0**,** matrix**[**0**][**0**]});**

matrix**[**0**][**0**]** **=** INT\_MIN**;**

**while(--**k**)** **{**

node t **=** pq**.**top**();**

pq**.**pop**();**

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

int x **=** t**.**x**+**dx**[**i**],** y **=** t**.**y**+**dy**[**i**];**

**if** **(**x **<** n **&&** y **<** n **&&** matrix**[**x**][**y**]** **!=** INT\_MIN**)** **{**

pq**.**push**(**node**{**x**,** y**,** matrix**[**x**][**y**]});**

matrix**[**x**][**y**]** **=** INT\_MIN**;**

**}**

**}**

**}**

**return** pq**.**top**().**value**;**

**}**

**};**

### 380. Insert Delete GetRandom O(1)

Medium

Design a data structure that supports all following operations in *average* **O(1)** time.

1. insert(val): Inserts an item val to the set if not already present.
2. remove(val): Removes an item val from the set if present.
3. getRandom: Returns a random element from current set of elements. Each element must have the **same probability** of being returned.

**Example:**

// Init an empty set.

RandomizedSet randomSet = new RandomizedSet();

// Inserts 1 to the set. Returns true as 1 was inserted successfully.

randomSet.insert(1);

// Returns false as 2 does not exist in the set.

randomSet.remove(2);

// Inserts 2 to the set, returns true. Set now contains [1,2].

randomSet.insert(2);

// getRandom should return either 1 or 2 randomly.

randomSet.getRandom();

// Removes 1 from the set, returns true. Set now contains [2].

randomSet.remove(1);

// 2 was already in the set, so return false.

randomSet.insert(2);

// Since 2 is the only number in the set, getRandom always return 2.

randomSet.getRandom();

class RandomizedSet **{**

public**:**

/\*\* Initialize your data structure here. \*/

RandomizedSet**():** sz**(**0**)** **{}**

/\*\* Inserts a value to the set. Returns true if the set did not already contain the specified element. \*/

bool insert**(**int val**)** **{**

**if** **(**m**.**count**(**val**))** **return** **false;**

m**[**val**]** **=** sz**++;**

nums**.**push\_back**(**val**);**

**return** **true;**

**}**

/\*\* Removes a value from the set. Returns true if the set contained the specified element. \*/

bool remove**(**int val**)** **{**

**if** **(!**m**.**count**(**val**))** **return** **false;**

int pos **=** m**[**val**];**

m**[**nums**[--**sz**]]** **=** pos**;**

nums**[**pos**]** **=** nums**[**sz**];**

m**.**erase**(**val**);**

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

**return** **true;**

**}**

/\*\* Get a random element from the set. \*/

int getRandom**()** **{**

**return** nums**[**rand**()** **%** sz**];**

**}**

private**:**

unordered\_map**<**int**,** int**>** m**;** // <number, pos>

int sz**;** // the size of nums

vector**<**int**>** nums**;**

**};**

/\*\*

\* Your RandomizedSet object will be instantiated and called as such:

\* RandomizedSet\* obj = new RandomizedSet();

\* bool param\_1 = obj->insert(val);

\* bool param\_2 = obj->remove(val);

\* int param\_3 = obj->getRandom();

\*/

### 381. Insert Delete GetRandom O(1) - Duplicates allowed

Hard

Design a data structure that supports all following operations in *average* **O(1)** time.

**Note: Duplicate elements are allowed.**

1. insert(val): Inserts an item val to the collection.
2. remove(val): Removes an item val from the collection if present.
3. getRandom: Returns a random element from current collection of elements. The probability of each element being returned is **linearly related** to the number of same value the collection contains.

**Example:**

// Init an empty collection.

RandomizedCollection collection = new RandomizedCollection();

// Inserts 1 to the collection. Returns true as the collection did not contain 1.

collection.insert(1);

// Inserts another 1 to the collection. Returns false as the collection contained 1. Collection now contains [1,1].

collection.insert(1);

// Inserts 2 to the collection, returns true. Collection now contains [1,1,2].

collection.insert(2);

// getRandom should return 1 with the probability 2/3, and returns 2 with the probability 1/3.

collection.getRandom();

// Removes 1 from the collection, returns true. Collection now contains [1,2].

collection.remove(1);

// getRandom should return 1 and 2 both equally likely.

collection.getRandom();

### 382. Linked List Random Node

Medium

Given a singly linked list, return a random node's value from the linked list. Each node must have the **same probability** of being chosen.

**Follow up:**  
What if the linked list is extremely large and its length is unknown to you? Could you solve this efficiently without using extra space?

**Example:**

// Init a singly linked list [1,2,3].

ListNode head = new ListNode(1);

head.next = new ListNode(2);

head.next.next = new ListNode(3);

Solution solution = new Solution(head);

// getRandom() should return either 1, 2, or 3 randomly. Each element should have equal probability of returning.

solution.getRandom();

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution **{**

public**:**

/\*\* **@param** head The linked list's head.

Note that the head is guaranteed to be not null, so it contains at least one node. \*/

Solution**(**ListNode**\*** head**)** **:** head**(**head**)** **{}**

/\*\* Returns a random node's value. \*/

int getRandom**()** **{**

//在此我扩展为k个随机取样(水塘抽样问题)

int k **=** 1**,** cnt **=** 0**;**

vector**<**int**>** ret**;**

ListNode **\***cur **=** head**;**

**while** **(**cur **!=** **nullptr)** **{**

**if** **(**cnt**++** **<** k**)** ret**.**push\_back**(**cur**->**val**);**

**else** **{**

int t **=** rand**()** **%** cnt**;**

**if** **(**t **<** k**)** ret**[**t**]** **=** cur**->**val**;**

**}**

cur **=** cur**->**next**;**

**}**

**return** ret**[**0**];**

**}**

private**:**

ListNode **\***head**;**

**};**

/\*\*

\* Your Solution object will be instantiated and called as such:

\* Solution\* obj = new Solution(head);

\* int param\_1 = obj->getRandom();

\*/

### 383. Ransom Note

Easy

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

class Solution **{**

public**:**

bool canConstruct**(**string ransomNote**,** string magazine**)** **{**

vector**<**int**>** vec**(**256**,** 0**);**

**for** **(**const auto **&**c **:** magazine**)** vec**[**c**]++;**

**for** **(**const auto **&**c **:** ransomNote**)** **{**

**if** **(--**vec**[**c**]** **<** 0**)** **return** **false;**

**}**

**return** **true;**

**}**

**};**

### 384. Shuffle an Array

Medium

Shuffle a set of numbers without duplicates.

**Example:**

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

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

Solution solution = new Solution(nums);

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

solution.shuffle();

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

solution.reset();

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

solution.shuffle();

### 385. Mini Parser

Medium

Given a nested list of integers represented as a string, implement a parser to deserialize it.

Each element is either an integer, or a list -- whose elements may also be integers or other lists.

**Note:** You may assume that the string is well-formed:

* String is non-empty.
* String does not contain white spaces.
* String contains only digits 0-9, [, - ,, ].

**Example 1:**

Given s = "324",

You should return a NestedInteger object which contains a single integer 324.

**Example 2:**

Given s = "[123,[456,[789]]]",

Return a NestedInteger object containing a nested list with 2 elements:

1. An integer containing value 123.

2. A nested list containing two elements:

i. An integer containing value 456.

ii. A nested list with one element:

a. An integer containing value 789.

### 386. Lexicographical Numbers

Medium

Given an integer *n*, return 1 - *n* in lexicographical order.

For example, given 13, return: [1,10,11,12,13,2,3,4,5,6,7,8,9].

Please optimize your algorithm to use less time and space. The input size may be as large as 5,000,000.

class Solution **{**

public**:**

vector**<**int**>** lexicalOrder**(**int n**)** **{**

dfs**(**0**,** n**);**

**return** res**;**

**}**

private**:**

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

void dfs**(**int i**,** int n**)** **{**

**if** **(**i **>** n**)** **return;**

**for** **(**int j **=** **(**i **==** 0 **?** 1 **:** 0**);** j **<=** 9**;** **++**j**)** **{**

int k **=** i**\***10 **+** j**;**

**if** **(**k **<=** n**)** **{**

res**.**push\_back**(**k**);**

dfs**(**k**,** n**);**

**}**

**}**

**}**

**};**

### 387. First Unique Character in a String

Easy

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.

class Solution **{**

public**:**

int firstUniqChar**(**string s**)** **{**

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

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

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

**if** **(**m**[**s**[**i**]]** **==** 1**)** **return** i**;**

**}**

**return** **-**1**;**

**}**

**};**

### 388. Longest Absolute File Path

Medium

Suppose we abstract our file system by a string in the following manner:

The string "dir\n\tsubdir1\n\tsubdir2\n\t\tfile.ext" represents:

dir

subdir1

subdir2

file.ext

The directory dir contains an empty sub-directory subdir1 and a sub-directory subdir2 containing a file file.ext.

The string "dir\n\tsubdir1\n\t\tfile1.ext\n\t\tsubsubdir1\n\tsubdir2\n\t\tsubsubdir2\n\t\t\tfile2.ext" represents:

dir

subdir1

file1.ext

subsubdir1

subdir2

subsubdir2

file2.ext

The directory dir contains two sub-directories subdir1 and subdir2. subdir1 contains a file file1.ext and an empty second-level sub-directory subsubdir1. subdir2 contains a second-level sub-directory subsubdir2 containing a file file2.ext.

We are interested in finding the longest (number of characters) absolute path to a file within our file system. For example, in the second example above, the longest absolute path is "dir/subdir2/subsubdir2/file2.ext", and its length is 32 (not including the double quotes).

Given a string representing the file system in the above format, return the length of the longest absolute path to file in the abstracted file system. If there is no file in the system, return 0.

**Note:**

* The name of a file contains at least a . and an extension.
* The name of a directory or sub-directory will not contain a ..

Time complexity required: O(n) where n is the size of the input string.

Notice that a/aa/aaa/file1.txt is not the longest file path, if there is another path aaaaaaaaaaaaaaaaaaaaa/sth.png.

class Solution **{**

public**:**

int lengthLongestPath**(**string input**)** **{**

input **+=** "\n"**;**

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

stk**.**push**(**0**);**

int res **=** 0**,** left **=** 0**,** pre **=** **-**1**;**

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

auto pos0 **=** input**.**find**(**'\n'**,** left**);**

auto pos1 **=** input**.**find**(**'.'**,** left**);**

**if** **(**pos0 **==** string**::**npos**)** **break;**

int cur **=** 0**;**

**while** **(**input**[**left**]** **==** '\t'**)** **{**

cur**++;**

left **+=** 1**;**

**}**

**for** **(**int i **=** cur**;** i **<=** pre**;** i**++)** stk**.**pop**();**

stk**.**push**((**int**)**pos0**-**left**+**stk**.**top**()+**1**);**

**if** **(**pos1 **!=** string**::**npos **&&** pos1 **<** pos0**)**

res **=** max**(**res**,** stk**.**top**()-**1**);**

pre **=** cur**;**

left **=** pos0 **+** 1**;**

**}**

**return** res**;**

**}**

**};**

### 389. Find the Difference

Easy

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.

class Solution **{**

public**:**

char findTheDifference**(**string s**,** string t**)** **{**

char res **=** 0**;**

**for** **(**auto **&**c **:** s**)** res **^=** c**;**

**for** **(**auto **&**c **:** t**)** res **^=** c**;**

**return** res**;**

**}**

**};**

### 390. Elimination Game

Medium

There is a list of sorted integers from 1 to *n*. Starting from left to right, remove the first number and every other number afterward until you reach the end of the list.

Repeat the previous step again, but this time from right to left, remove the right most number and every other number from the remaining numbers.

We keep repeating the steps again, alternating left to right and right to left, until a single number remains.

Find the last number that remains starting with a list of length *n*.

**Example:**

Input:

n = 9,

1 2 3 4 5 6 7 8 9

2 4 6 8

2 6

6

Output:

6

class Solution **{**

public**:**

int lastRemaining**(**int n**)** **{**

int A **=** 1**,** B **=** 0**;**

int flag **=** 0**;**

**while** **(**n **!=** 1**)** **{**

**if** **(**n **%** 2 **==** 0 **&&** flag**)** B **-=** A**;**

A **\*=** 2**;**

n **/=** 2**;**

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

**}**

**return** A **+** B**;**

**}**

**};**

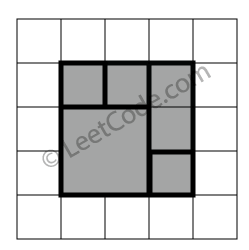
### 391. Perfect Rectangle

Hard

Given N axis-aligned rectangles where N > 0, determine if they all together form an exact cover of a rectangular region.

Each rectangle is represented as a bottom-left point and a top-right point. For example, a unit square is represented as [1,1,2,2]. (coordinate of bottom-left point is (1, 1) and top-right point is (2, 2)).

**Example 1:**

rectangles = [

[1,1,3,3],

[3,1,4,2],

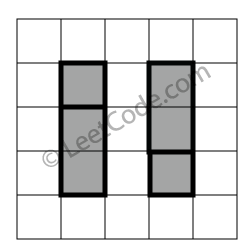
[3,2,4,4],

[1,3,2,4],

[2,3,3,4]

]

Return true. All 5 rectangles together form an exact cover of a rectangular region.

**Example 2:**

rectangles = [

[1,1,2,3],

[1,3,2,4],

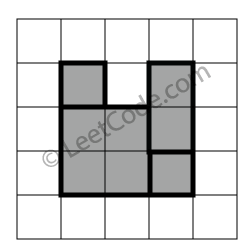
[3,1,4,2],

[3,2,4,4]

]

Return false. Because there is a gap between the two rectangular regions.

**Example 3:**

rectangles = [

[1,1,3,3],

[3,1,4,2],

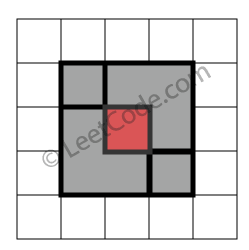
[1,3,2,4],

[3,2,4,4]

]

Return false. Because there is a gap in the top center.

**Example 4:**

rectangles = [

[1,1,3,3],

[3,1,4,2],

[1,3,2,4],

[2,2,4,4]

]

Return false. Because two of the rectangles overlap with each other.

### 392. Is Subsequence

Easy

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

You may assume that there is only lower case English letters in both **s** and **t**. **t** is potentially a very long (length ~= 500,000) string, and **s** is a short string (<=100).

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

**Example 1:**  
**s** = "abc", **t** = "ahbgdc"

Return true.

**Example 2:**  
**s** = "axc", **t** = "ahbgdc"

Return false.

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

**Credits:**  
Special thanks to [@pbrother](https://leetcode.com/pbrother/) for adding this problem and creating all test cases.

class Solution **{**

public**:**

bool isSubsequence**(**string s**,** string t**)** **{**

int n **=** s**.**length**(),** m **=** t**.**length**(),** j **=** 0**;**

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

**while** **(**j **<** m **&&** t**[**j**]** **!=** s**[**i**])** j**++;**

**if** **(**j**++** **==** m**)** **return** **false;**

**if** **(**i **==** n**-**1**)** **return** **true;**

**}**

**return** **true;**

**}**

**};**

### 393. UTF-8 Validation

Medium

A character in UTF8 can be from **1 to 4 bytes** long, subjected to the following rules:

1. For 1-byte character, the first bit is a 0, followed by its unicode code.
2. For n-bytes character, the first n-bits are all one's, the n+1 bit is 0, followed by n-1 bytes with most significant 2 bits being 10.

This is how the UTF-8 encoding would work:

Char. number range | UTF-8 octet sequence

(hexadecimal) | (binary)

--------------------+---------------------------------------------

0000 0000-0000 007F | 0xxxxxxx

0000 0080-0000 07FF | 110xxxxx 10xxxxxx

0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx

0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Given an array of integers representing the data, return whether it is a valid utf-8 encoding.

**Note:**  
The input is an array of integers. Only the **least significant 8 bits** of each integer is used to store the data. This means each integer represents only 1 byte of data.

**Example 1:**

data = [197, 130, 1], which represents the octet sequence: **11000101 10000010 00000001**.

Return **true**.

It is a valid utf-8 encoding for a 2-bytes character followed by a 1-byte character.

**Example 2:**

data = [235, 140, 4], which represented the octet sequence: **11101011 10001100 00000100**.

Return **false**.

The first 3 bits are all one's and the 4th bit is 0 means it is a 3-bytes character.

The next byte is a continuation byte which starts with 10 and that's correct.

But the second continuation byte does not start with 10, so it is invalid.

class Solution **{**

public**:**

bool validUtf8**(**vector**<**int**>&** data**)** **{**

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

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

**if** **(**cnt **==** 0**)** **{**

**if** **((**c **>>** 5**)** **==** 0b110**)** cnt **=** 1**;**

**else** **if** **((**c **>>** 4**)** **==** 0b1110**)** cnt **=** 2**;**

**else** **if** **((**c **>>** 3**)** **==** 0b11110**)** cnt **=** 3**;**

**else** **if** **((**c **>>** 7**)** **==** 0b1**)** **return** **false;**

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

**if** **((**c **>>** 6**)** **!=** 0b10**)** **return** **false;**

cnt**--;**

**}**

**}**

**return** cnt **==** 0**;**

**}**

**};**

### 394. Decode String

Medium

Given an encoded string, return its decoded string.

The encoding rule is: k[encoded\_string], where the *encoded\_string* inside the square brackets is being repeated exactly *k* times. Note that *k* is guaranteed to be a positive integer.

You may assume that the input string is always valid; No extra white spaces, square brackets are well-formed, etc.

Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, *k*. For example, there won't be input like 3a or 2[4].

**Examples:**

s = "3[a]2[bc]", return "aaabcbc".

s = "3[a2[c]]", return "accaccacc".

s = "2[abc]3[cd]ef", return "abcabccdcdcdef".

class Solution **{**

public**:**

string decodeString**(**string s**)** **{**

stack**<**string**>** chars**;**

stack**<**int**>** nums**;**

string res**;**

int num **=** 0**;**

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

**if** **(**isdigit**(**c**))** num **=** num**\***10 **+** **(**c**-**'0'**);**

**else** **if** **(**isalpha**(**c**))** res**.**push\_back**(**c**);**

**else** **if** **(**c **==** '['**)** **{**

chars**.**push**(**res**);**

nums**.**push**(**num**);**

res **=** ""**;**

num **=** 0**;**

**}**

**else** **if** **(**c **==** ']'**)** **{**

string tmp **=** res**;**

int k **=** nums**.**top**();**

nums**.**pop**();**

res **=** chars**.**top**();**

chars**.**pop**();**

**while** **(**k**--)** res **+=** tmp**;**

**}**

**}**

**return** res**;**

**}**

**};**

### 395. Longest Substring with At Least K Repeating Characters

Medium

Find the length of the longest substring ***T*** of a given string (consists of lowercase letters only) such that every character in ***T*** appears no less than *k* times.

**Example 1:**

Input:

s = "aaabb", k = 3

Output:

3

The longest substring is "aaa", as 'a' is repeated 3 times.

**Example 2:**

Input:

s = "ababbc", k = 2

Output:

5

The longest substring is "ababb", as 'a' is repeated 2 times and 'b' is repeated 3 times.

class Solution **{**

public**:**

int longestSubstring**(**string s**,** int k**)** **{**

int cnt**[**256**]** **=** **{**0**};**

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

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

**for** **(**auto **&**c **:** s**)** **if** **(**cnt**[**c**]** **<** k**)** **{**

c **=** ' '**;**

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

**}**

**if** **(**ok**)** **return** s**.**length**();**

stringstream ss**(**s**);**

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

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

res **=** max**(**res**,** longestSubstring**(**s**,** k**));**

**}**

**return** res**;**

**}**

**};**

### 396. Rotate Function

Medium

Given an array of integers A and let *n* to be its length.

Assume Bk to be an array obtained by rotating the array A *k* positions clock-wise, we define a "rotation function" F on A as follow:

F(k) = 0 \* Bk[0] + 1 \* Bk[1] + ... + (n-1) \* Bk[n-1].

Calculate the maximum value of F(0), F(1), ..., F(n-1).

**Note:**  
*n* is guaranteed to be less than 105.

**Example:**

A = [4, 3, 2, 6]

F(0) = (0 \* 4) + (1 \* 3) + (2 \* 2) + (3 \* 6) = 0 + 3 + 4 + 18 = 25

F(1) = (0 \* 6) + (1 \* 4) + (2 \* 3) + (3 \* 2) = 0 + 4 + 6 + 6 = 16

F(2) = (0 \* 2) + (1 \* 6) + (2 \* 4) + (3 \* 3) = 0 + 6 + 8 + 9 = 23

F(3) = (0 \* 3) + (1 \* 2) + (2 \* 6) + (3 \* 4) = 0 + 2 + 12 + 12 = 26

So the maximum value of F(0), F(1), F(2), F(3) is F(3) = 26.

class Solution **{**

public**:**

int maxRotateFunction**(**vector**<**int**>&** A**)** **{**

long long n **=** A**.**size**(),** sum **=** 0**,** res **=** 0**;**

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

sum **+=** A**[**i**];**

res **+=** i **\*** A**[**i**];**

**}**

long long temp **=** res**;**

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

temp **+=** n**\***A**[**i**]** **-** sum**;**

res **=** max**(**temp**,** res**);**

**}**

**return** res**;**

**}**

**};**

### 397. Integer Replacement

Medium

Given a positive integer *n* and you can do operations as follow:

1. If *n* is even, replace *n* with *n*/2.
2. If *n* is odd, you can replace *n* with either *n* + 1 or *n* - 1.

What is the minimum number of replacements needed for *n* to become 1?

**Example 1:**

**Input:**

8

**Output:**

3

**Explanation:**

8 -> 4 -> 2 -> 1

**Example 2:**

**Input:**

7

**Output:**

4

**Explanation:**

7 -> 8 -> 4 -> 2 -> 1

or

7 -> 6 -> 3 -> 2 -> 1

class Solution **{**

public**:**

int integerReplacement**(**int n**)** **{**

**if** **(**n **==** 1**)** **return** 0**;**

**else** **if** **(**n **==** INT\_MAX**)** **return** 32**;**

**if** **(**m**.**find**(**n**)** **!=** m**.**end**())** **return** m**[**n**];**

**if** **(**n **%** 2 **==** 0**)** **{**

**return** m**[**n**]** **=** 1 **+** integerReplacement**(**n**/**2**);**

**}**

int a **=** 1 **+** integerReplacement**(**n**-**1**);**

int b **=** 1 **+** integerReplacement**(**n**+**1**);**

**return** m**[**n**]** **=** min**(**a**,** b**);**

**}**

private**:**

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

**};**

### 398. Random Pick Index

Medium

Given an array of integers with possible duplicates, randomly output the index of a given target number. You can assume that the given target number must exist in the array.

**Note:**  
The array size can be very large. Solution that uses too much extra space will not pass the judge.

**Example:**

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

Solution solution = new Solution(nums);

// pick(3) should return either index 2, 3, or 4 randomly. Each index should have equal probability of returning.

solution.pick(3);

// pick(1) should return 0. Since in the array only nums[0] is equal to 1.

solution.pick(1);

class Solution **{**

public**:**

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

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

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

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

**}**

**}**

int pick**(**int target**)** **{**

queue**<**int**>** **&**q **=** m**[**target**];**

int ret **=** q**.**front**();**

q**.**pop**();**

q**.**push**(**ret**);**

**return** ret**;**

**}**

private**:**

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

**};**

/\*\*

\* Your Solution object will be instantiated and called as such:

\* Solution\* obj = new Solution(nums);

\* int param\_1 = obj->pick(target);

\*/

### 399. Evaluate Division

Medium

Equations are given in the format A / B = k, where A and B are variables represented as strings, and k is a real number (floating point number). Given some queries, return the answers. If the answer does not exist, return -1.0.

**Example:**  
Given a / b = 2.0, b / c = 3.0.  
queries are: a / c = ?, b / a = ?, a / e = ?, a / a = ?, x / x = ? .  
return [6.0, 0.5, -1.0, 1.0, -1.0 ].

The input is: vector<pair<string, string>> equations, vector<double>& values, vector<pair<string, string>> queries , where equations.size() == values.size(), and the values are positive. This represents the equations. Return vector<double>.

According to the example above:

equations = [ ["a", "b"], ["b", "c"] ],

values = [2.0, 3.0],

queries = [ ["a", "c"], ["b", "a"], ["a", "e"], ["a", "a"], ["x", "x"] ].

The input is always valid. You may assume that evaluating the queries will result in no division by zero and there is no contradiction.

### 400. Nth Digit

Medium

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

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

**Example 1:**

**Input:**

3

**Output:**

3

**Example 2:**

**Input:**

11

**Output:**

0

**Explanation:**

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

class Solution **{**

public**:**

int findNthDigit**(**int n**)** **{**

int k **=** 1**;**

// k位数第n个

**while** **(**n **>** 9**\***pow**(**10**,** k**-**1**)\***k**)** **{**

n **-=** 9**\***pow**(**10**,** k**-**1**)\***k**;**

k**++;**

**}**

int m **=** **(**n**-**1**)/**k**,** num **=** pow**(**10**,** k**-**1**)+**m**;**

n **-=** m**\***k**;**

// num的第n位, 即从右往左数第k+1-n位

**return** **(**num **/** **(**int**)**pow**(**10**,** k**-**n**))** **%** 10**;**

**}**

**};**