Data Structures & Algorithms

INterviewbit

Bhattacharya, Mrinal

2020

Array

**Min Steps in Infinite Grid**

You are in an infinite 2D grid where you can move in any of the 8 directions :

(x,y) to

(x+1, y),

(x - 1, y),

(x, y+1),

(x, y-1),

(x-1, y-1),

(x+1,y+1),

(x-1,y+1),

(x+1,y-1)

You are given a sequence of points and the order in which you need to cover the points. Give the minimum number of steps in which you can achieve it. You start from the first point.

Input :

Given two integer arrays A and B, where A[i] is x coordinate and B[i] is y coordinate of ith point respectively.

Output :

Return an Integer, i.e minimum number of steps.

Example :

Input : [(0, 0), (1, 1), (1, 2)]

Output : 2 It takes 1 step to move from (0, 0) to (1, 1). It takes one more step to move from (1, 1) to (1, 2)

**Add One To Number**

Given a non-negative number represented as an array of digits,

add 1 to the number ( increment the number represented by the digits ).

The digits are stored such that the most significant digit is at the head of the list.

Example:

If the vector has [1, 2, 3]

the returned vector should be [1, 2, 4]

as 123 + 1 = 124.



**Max Sum Contiguous Subarray**

Find the contiguous subarray within an array, A of length N which has the largest sum.

Input 1:

A = [1, 2, 3, 4, -10]

Output 1:

10

Explanation 1:

The subarray [1, 2, 3, 4] has the maximum possible sum of 10.

Input 2:

A = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

Output 2:

6

Explanation 2:

The subarray [4,-1,2,1] has the maximum possible sum of 6.



**Maximum Absolute Difference**

You are given an array of N integers, A1, A2 ,…, AN. Return maximum value of f(i, j) for all 1 ≤ i, j ≤ N.  
f(i, j) is defined as |A[i] - A[j]| + |i - j|, where |x| denotes absolute value of x.

A=[1, 3, -1]

f(1, 1) = f(2, 2) = f(3, 3) = 0

f(1, 2) = f(2, 1) = |1 - 3| + |1 - 2| = 3

f(1, 3) = f(3, 1) = |1 - (-1)| + |1 - 3| = 4

f(2, 3) = f(3, 2) = |3 - (-1)| + |2 - 3| = 5

So, we return 5.



**Repeat and Missing Number Array**

You are given a read only array of n integers from 1 to n.

Each integer appears exactly once except A which appears twice and B which is missing.

Return A and B.

Input:[3 1 2 5 3]

Output:[3, 4]

A = 3, B = 4



TODO:

**Flip**

**Max Non Negative SubArray**

**Triplets with Sum between given range**

**Spiral Order Matrix II**

Given an integer A, generate a square matrix filled with elements from 1 to A2 in spiral order.

Input 1:

A = 3

Output 1:

[ [ 1, 2, 3 ],

[ 8, 9, 4 ],

[ 7, 6, 5 ] ]

Input 2:

4

Output 2:

[ [1, 2, 3, 4],

[12, 13, 14, 5],

[11, 16, 15, 6],

[10, 9, 8, 7] ]



**Pascal Triangle**

Given numRows, generate the first numRows of Pascal’s triangle.

Pascal’s triangle : To generate A[C] in row R, sum up A’[C] and A’[C-1] from previous row R - 1.

Given numRows = 5,

Return

[

[1],

[1,1],

[1,2,1],

[1,3,3,1],

[1,4,6,4,1]

]



**Kth Row of Pascal's Triangle**

Given an index k, return the kth row of the Pascal’s triangle.

Pascal’s triangle:To generate A[C] in row R, sum up A’[C] and A’[C-1] from previous row R - 1.

Example:

Input : k = 3

Return : [1,3,3,1]



**Anti Diagonals**

Give a N\*N square matrix, return an array of its anti-diagonals. Look at the example for more details.

Input:

1 2 3

4 5 6

7 8 9

Return the following :

[

[1],

[2, 4],

[3, 5, 7],

[6, 8],

[9]

]



**Noble Integer**

Given an integer array, find if an integer p exists in the array such that the number of integers greater than p in the array equals to p  
If such an integer is found return 1 else return -1.



**Largest Number**

Given a list of non negative integers, arrange them such that they form the largest number.

For example:

Given [3, 30, 34, 5, 9], the largest formed number is 9534330.

Note: The result may be very large, so you need to return a string instead of an integer.



**Wave Array**

Given an array of integers, sort the array into a wave like array and return it,   
In other words, arrange the elements into a sequence such that a1 >= a2 <= a3 >= a4 <= a5.....

Example

Given [1, 2, 3, 4]

One possible answer : [2, 1, 4, 3]

Another possible answer : [4, 1, 3, 2]



**Max Distance**

Given an array A of integers, find the maximum of j - i subjected to the constraint of A[i] <= A[j].

If there is no solution possible, return -1.

Example :

A : [3 5 4 2]

Output : 2 for the pair (3, 4)



**Maximum Unsorted Subarray**

You are given an array (zero indexed) of N non-negative integers, A0, A1 ,…, AN-1.  
Find the minimum sub array Al, Al+1 ,…, Ar so if we sort(in ascending order) that sub array, then the whole array should get sorted.  
If A is already sorted, output -1.

Example :

Input 1:

A = [1, 3, 2, 4, 5]

Return: [1, 2]

Input 2:

A = [1, 2, 3, 4, 5]

Return: [-1]

In the above example(Input 1), if we sort the subarray A1, A2, then whole array A should get sorted.



**Hotel Bookings Possible**

A hotel manager has to process N advance bookings of rooms for the next season. His hotel has K rooms. Bookings contain an arrival date and a departure date. He wants to find out whether there are enough rooms in the hotel to satisfy the demand. Write a program that solves this problem in time O(N log N) .

Input:

First list for arrival time of booking.

Second list for departure time of booking.

Third is K which denotes count of rooms.

Output:

A boolean which tells whether its possible to make a booking.

Return 0/1 for C programs.

O -> No there are not enough rooms for N booking.

1 -> Yes there are enough rooms for N booking.

Example :

Input :

Arrivals : [1 3 5]

Departures : [2 6 8]

K : 1

Return : False / 0

At day = 5, there are 2 guests in the hotel. But I have only one room.



**Find Duplicate in Array**

Given a read only array of n + 1 integers between 1 and n, find one number that repeats in linear time using less than O(n) space and traversing the stream sequentially O(1) times.

Sample Input:

[3 4 1 4 1]

Sample Output:

1

If there are multiple possible answers ( like in the sample case above ), output any one.

If there is no duplicate, output -1



**Maximum Consecutive Gap**

Given an unsorted array, find the maximum difference between the successive elements in its sorted form.

Try to solve it in linear time/space.

Example :

Input : [1, 10, 5]

Output : 5

Return 0 if the array contains less than 2 elements.

You may assume that all the elements in the array are non-negative integers and fit in the 32-bit signed integer range.

You may also assume that the difference will not overflow.



**MAXSPPROD**

You are given an array A containing N integers. The special product of each ith integer in this array is defined as the product of the following:<ul>

 LeftSpecialValue: For an index i, it is defined as the index j such that A[j]>A[i] (i>j). If multiple A[j]’s are present in multiple positions, the LeftSpecialValue is the maximum value of j.

 RightSpecialValue: For an index i, it is defined as the index j such that A[j]>A[i] (j>i). If multiple A[j]s are present in multiple positions, the RightSpecialValue is the minimum value of j.

Write a program to find the maximum special product of any integer in the array.

Input: You will receive array of integers as argument to function.

Return: Maximum special product of any integer in the array modulo 1000000007.

Note: If j does not exist, the LeftSpecialValue and RightSpecialValue are considered to be 0.

Constraints 1 <= N <= 10^5 1 <= A[i] <= 10^9



**Rotate Matrix**

You are given an n x n 2D matrix representing an image.

Rotate the image by 90 degrees (clockwise).

You need to do this in place.

Note that if you end up using an additional array, you will only receive partial score.

Example:

If the array is

[

[1, 2],

[3, 4]

]

Then the rotated array becomes:

[

[3, 1],

[4, 2]

]



**Next Permutation**

Implement the next permutation, which rearranges numbers into the numerically next greater permutation of numbers for a given array A of size N.

If such arrangement is not possible, it must be rearranged as the lowest possible order i.e., sorted in an ascending order.

Input 1:

A = [1, 2, 3]

Output 1:

[1, 3, 2]

Input 2:

A = [3, 2, 1]

Output 2:

[1, 2, 3]

Input 3:

A = [1, 1, 5]

Output 3:

[1, 5, 1]

Input 4:

A = [20, 50, 113]

Output 4:

[20, 113, 50]



**Find Permutation**

Given a positive integer n and a string s consisting only of letters D or I, you have to find any permutation of first n positive integer that satisfy the given input string.

D means the next number is smaller, while I means the next number is greater.

Example :

Input 1:

n = 3

s = ID

Return: [1, 3, 2]



**Merge Intervals**

Given a set of non-overlapping intervals, insert a new interval into the intervals (merge if necessary).

You may assume that the intervals were initially sorted according to their start times.

Given intervals [1,3],[6,9] insert and merge [2,5] would result in [1,5],[6,9].

Example 2:

Given [1,2],[3,5],[6,7],[8,10],[12,16], insert and merge [4,9] would result in [1,2],[3,10],[12,16].

This is because the new interval [4,9] overlaps with [3,5],[6,7],[8,10].

Make sure the returned intervals are also sorted.



**Merge Overlapping Intervals**

Given a collection of intervals, merge all overlapping intervals.

For example:

Given [1,3],[2,6],[8,10],[15,18],

return [1,6],[8,10],[15,18].

Make sure the returned intervals are sorted.



**Set Matrix Zeros**

Given a matrix, A of size M x N of 0s and 1s. If an element is 0, set its entire row and column to 0.

Input 1:

[ [1, 0, 1],

[1, 1, 1],

[1, 1, 1] ]

Output 1:

[ [0, 0, 0],

[1, 0, 1],

[1, 0, 1] ]

Input 2:

[ [1, 0, 1],

[1, 1, 1],

[1, 0, 1] ]

Output 2:

[ [0, 0, 0],

[1, 0, 1],

[0, 0, 0] ]



**First Missing Integer**

Given an unsorted integer array, find the first missing positive integer.

Example:

Given [1,2,0] return 3,

[3,4,-1,1] return 2,

[-8, -7, -6] returns 1

Your algorithm should run in O(n) time and use constant space.



**Repeat and Missing Number Array**

You are given a read only array of n integers from 1 to n.

Each integer appears exactly once except A which appears twice and B which is missing.

Return A and B.

Input:[3 1 2 5 3]

Output:[3, 4]

A = 3, B = 4



**Find Duplicate in Array**

Given a read only array of n + 1 integers between 1 and n, find one number that repeats in linear time using less than O(n) space and traversing the stream sequentially O(1) times.

Input [3 4 1 4 1]

Output 1



**N/3 Repeat Number**

You’re given a read only array of n integers. Find out if any integer occurs more than n/3 times in the array in linear time and constant additional space.

If so, return the integer. If not, return -1.

If there are multiple solutions, return any one.

Input : [1 2 3 1 1]

Output : 1

1 occurs 3 times which is more than 5/3 times.



**Spiral Order Matrix I**

Given a matrix of m \* n elements (m rows, n columns), return all elements of the matrix in spiral order.

Example:

Given the following matrix:

[

[ 1, 2, 3 ],

[ 4, 5, 6 ],

[ 7, 8, 9 ]

]

You should return

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



Math

**All Factors**

Given a number N, find all factors of N.

**Example:**

N = 6

factors = {1, 2, 3, 6}



**Binary Representation**

Given a number N >= 0, find its representation in binary.

**Example:**

if N = 6,

binary form = 110



**Prime Numbers**

Given a number N, find all prime numbers upto N ( N included ).

**Example:**

if N = 7,

all primes till 7 = {2, 3, 5, 7}



**Verify Prime**

Given a number N, verify if N is prime or not.

Return 1 if N is prime, else return 0.

**Example :**

Input : 7

Output : True



**Prime Sum**

Given an even number ( greater than 2 ), return two prime numbers whose sum will be equal to given number.

**NOTE** A solution will always exist. read [Goldbach’s conjecture](https://en.wikipedia.org/wiki/Goldbach%27s_conjecture)

**Example:**

Input : 4

Output: 2 + 2 = 4



**Sum of pairwise Hamming Distance**

For example,

HammingDistance(2, 7) = 2, as only the *first* and the *third* bit differs in the binary representation of **2** (010) and **7** (111).

Given an array of N non-negative integers, find the sum of hamming distances of all pairs of integers in the array.  
Return the answer modulo 1000000007.

**Example**

Let f(x, y) be the hamming distance defined above.

A=[2, 4, 6]

We return,

f(2, 2) + f(2, 4) + f(2, 6) +

f(4, 2) + f(4, 4) + f(4, 6) +

f(6, 2) + f(6, 4) + f(6, 6) =

0 + 2 + 1

2 + 0 + 1

1 + 1 + 0 = 8



**Power Of Two Integers**

Given a positive integer which fits in a 32 bit signed integer, find if it can be expressed as A^P where P > 1 and A > 0. A and P both should be integers.

**Example**

Input : 4

Output : True

as 2^2 = 4.



**FizzBuzz**

Given a positive integer A, return an array of strings with all the integers from 1 to N.   
But for multiples of **3** the array should have “Fizz” instead of the number.   
For the multiples of **5**, the array should have “Buzz” instead of the number.   
For numbers which are multiple of **3** and **5** both, the array should have “FizzBuzz” instead of the number.

Look at the example for more details.

**Example**

A = 5

Return: [1 2 Fizz 4 Buzz]



**Excel Column Number**

Given a column title as appears in an Excel sheet, return its corresponding column number.

**Example:**

A -> 1

B -> 2

C -> 3

...

Z -> 26

AA -> 27

AB -> 28



**Excel Column Title**

Given a positive integer, return its corresponding column title as appear in an Excel sheet.

For example:

1 -> A

2 -> B

3 -> C

...

26 -> Z

27 -> AA

28 -> AB



**Palindrome Integer**

Determine whether an integer is a palindrome. Do this without extra space.

A palindrome integer is an integer x for which reverse(x) = x where reverse(x) is x with its digit reversed.  
Negative numbers are not palindromic.

**Example :**

Input : 12121

Output : True

Input : 123

Output : False



**Reverse integer**

Reverse digits of an integer.

**Example1:**

x = 123,

return 321  
**Example2:**

x = -123,

return -321

Return 0 if the result overflows and does not fit in a 32 bit signed integer



**Numbers of length N and value less than K**

Given a set of digits (A) in sorted order, find how many numbers of length B are possible whose value is less than number C.

**NOTE:** All numbers can only have digits from the given set.

**Examples:**

Input:

0 1 5

1

2

Output:

2 (0 and 1 are possible)

Input:

0 1 2 5

2

21

Output:

5 (10, 11, 12, 15, 20 are possible)



**Rearrange Array**

Rearrange a given array so that Arr[i] becomes Arr[Arr[i]] with O(1) extra space.

**Example:**

Input : [1, 0]

Return : [0, 1]



**Grid Unique Paths**

A robot is located at the top-left corner of an **A x B grid** (marked ‘Start’ in the diagram below).



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

How many possible unique paths are there?

Input : A = 2, B = 2

Output : 2

2 possible routes : (0, 0) -> (0, 1) -> (1, 1)

OR : (0, 0) -> (1, 0) -> (1, 1)



**Grid Unique Paths**

Given 2 non negative integers m and n, find gcd(m, n)

GCD of 2 integers m and n is defined as the greatest integer g such that g is a divisor of both m and n.  
Both m and n fit in a 32 bit signed integer.

**Example**

m : 6

n : 9

GCD(m, n) : 3



**Trailing Zeros in Factorial**

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

Note: Your solution should be in logarithmic time complexity.

Example :

n = 5

n! = 120

Number of trailing zeros = 1

So, return 1



**Largest Coprime Divisor**

You are given two positive numbers **A** and **B**. You need to find the maximum valued integer **X** such that:

* **X** divides **A** i.e. **A** % **X** = **0**
* **X** and **B** are co-prime i.e. gcd(**X**, **B**) = **1**

For example,

A = 30

B = 12

We return

X = 5



TODO

Sorted Permutation Rank with Repeats

Sorted Permutation Rank

City Tour

Binary Search

**Count Element Occurence**

Given a sorted array of integers, find the number of occurrences of a given target value.  
Your algorithm’s runtime complexity must be in the order of O(log n).  
If the target is not found in the array, return 0

\*\*Example : \*\*  
Given [5, 7, 7, 8, 8, 10] and target value 8,  
return 2.



**Rotated Array**

Suppose a sorted array A is rotated at some pivot unknown to you beforehand.

(i.e., 0 1 2 4 5 6 7 might become 4 5 6 7 0 1 2).

Find the minimum element.

The array will not contain duplicates.



**Matrix Median**

Given a matrix of integers **A** of size **N x M** in which each row is sorted.

Find an return the overall median of the matrix **A**.

**Note:** No extra memory is allowed.

**Note:** Rows are numbered from top to bottom and columns are numbered from left to right.

Input Format

The first and only argument given is the integer matrix A.

Output Format

Return the overall median of the matrix A.

Constraints

1 <= N, M <= 10^5

1 <= N\*M <= 10^6

1 <= A[i] <= 10^9

N\*M is odd

For Example

Input 1:

A = [ [1, 3, 5],

[2, 6, 9],

[3, 6, 9] ]

Output 1:

5

Explanation 1:

A = [1, 2, 3, 3, 5, 6, 6, 9, 9]

Median is 5. So, we return 5.

Input 2:

A = [ [5, 17, 100] ]

Output 2:

17 ``` Matrix=



**Square Root of Integer**

Given an integar **A**.

Compute and return the **square root of A**.

If **A** is not a perfect square, return **floor(sqrt(A))**.

Input 1:

A = 11

Output 1:

3

Input 2:

A = 9

Output 2:

3



**Painter's Partition Problem**

Given 2 integers **A** and **B** and an array of integars **C** of size **N**.

Element **C[i]** represents length of **ith** board.

You have to paint all **N** boards **[C0, C1, C2, C3 … CN-1]**. There are **A** painters available and each of them takes **B** units of time to paint 1 unit of board.

Calculate and return minimum time required to paint all boards under the constraints that **any painter will only paint contiguous sections of board.**

* 2 painters cannot share a board to paint. That is to say, a board  
  cannot be painted partially by one painter, and partially by another.
* A painter will only paint contiguous boards. Which means a  
  configuration where painter 1 paints board 1 and 3 but not 2 is  
  invalid.

Input 1:

A = 2

B = 5

C = [1, 10]

Output 1:

50

Explanation 1:

Possibility 1:- same painter paints both blocks, time taken = 55units

Possibility 2:- Painter 1 paints block 1, painter 2 paints block 2, time take = max(5, 50) = 50

There are no other distinct ways to paint boards.

ans = 50%10000003

Input 2:

A = 10

B = 1

C = [1, 8, 11, 3]

Output 2:

11



**Allocate Books**

Given an array of integers **A** of size **N** and an integer **B**.

College library has **N** bags,the **ith** book has **A[i]** number of pages.

You have to allocate books to **B** number of students so that maximum number of pages alloted to a student is minimum.

A book will be allocated to exactly one student.

Each student has to be allocated at least one book.

Allotment should be in contiguous order, for example: A student cannot be allocated book 1 and book 3, skipping book 2.

Calculate and return that minimum possible number.

**NOTE:** Return -1 if a valid assignment is not possible.

Input 1:

A = [12, 34, 67, 90]

B = 2

Output 1:

113

Explanation 1:

There are 2 number of students. Books can be distributed in following fashion :

1) [12] and [34, 67, 90]

Max number of pages is allocated to student 2 with 34 + 67 + 90 = 191 pages

2) [12, 34] and [67, 90]

Max number of pages is allocated to student 2 with 67 + 90 = 157 pages

3) [12, 34, 67] and [90]

Max number of pages is allocated to student 1 with 12 + 34 + 67 = 113 pages

Of the 3 cases, Option 3 has the minimum pages = 113.

Input 2:

A = [5, 17, 100, 11]

B = 4

Output 2:

100



**Matrix Search**

Given a matrix of integers **A** of size **N x M** and an integer **B**.

Write an efficient algorithm that searches for integar **B** in matrix **A**.

This matrix **A** has the following properties:

1. Integers in each row are sorted from left to right.
2. The **first integer** of each row is greater than or equal to the **last integer** of the previous row.

Return **1** if **B** is present in **A**, else return **0**.

**Note:** Rows are numbered from top to bottom and columns are numbered from left to right.

Input 1:

A =

[ [1, 3, 5, 7],

[10, 11, 16, 20],

[23, 30, 34, 50] ]

B = 3

Output 1:

1

Input 2:

A = [ [5, 17, 100, 111]

[119, 120, 127, 131] ]

B = 3

Output 2:

0



**Search for a Range**

Given a sorted array of integers **A**(0 based index) of size **N**, find the starting and ending position of a given integar **B** in array **A**.

Your algorithm’s runtime complexity must be in the order of O(log n).

Return an array of size 2, such that first element = starting position of B in A and second element = ending position of B in A, if B is not found in A return [-1, -1].

Input 1:

A = [5, 7, 7, 8, 8, 10]

B = 8

Output 1:

[3, 4]

Explanation 1:

First occurence of 8 in A is at index 3

Second occurence of 8 in A is at index 4

ans = [3, 4]

Input 2:

A = [5, 17, 100, 111]

B = 3

Output 2:

[-1, -1]



**Sorted Insert Position**

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

You may assume no duplicates in the array.

Here are few examples.

[1,3,5,6], 5 → 2

[1,3,5,6], 2 → 1

[1,3,5,6], 7 → 4

[1,3,5,6], 0 → 0



**Implement Power Function**

Implement pow(x, n) % d.

In other words, given x, n and d,

find (xn % d)

Note that remainders on division cannot be negative.   
In other words, make sure the answer you return is non negative.

Input : x = 2, n = 3, d = 3

Output : 2

2^3 % 3 = 8 % 3 = 2.



**Rotated Sorted Array Search**

Given an array of integers **A** of size **N** and an integer **B**.

array **A** is rotated at some pivot unknown to you beforehand.

(i.e., 0 1 2 4 5 6 7 might become 4 5 6 7 0 1 2 ).

You are given a target value **B** to search. If found in the array, return its index, otherwise return **-1**.

You may assume no duplicate exists in the array.

**NOTE:-** Array **A** was sorted in non-decreasing order before rotation.

Input 1:

A = [4, 5, 6, 7, 0, 1, 2, 3]

B = 4

Output 1:

0

Explanation 1:

Target 4 is found at index 0 in A.

Input 2:

A = [5, 17, 100, 3]

B = 6

Output 2:

-1



**Median of Array**

There are two sorted arrays A and B of size m and n respectively.

Find the median of the two sorted arrays ( The median of the array formed by merging both the arrays ).

The overall run time complexity should be O(log (m+n)).

Sample Input

A : [1 4 5]

B : [2 3]

Sample Output

3



String

**Palindrome String**

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

Example:

"A man, a plan, a canal: Panama" is a palindrome.

"race a car" is not a palindrome.

Return 0 / 1 ( 0 for false, 1 for true ) for this problem



**Longest Common Prefix**

Given the array of strings **A**,   
you need to find the longest string S which is the prefix of **ALL** the strings in the array.

Longest common prefix for a pair of strings **S1** and **S2** is the longest string **S** which is the prefix of both **S1**   
and **S2**.

For Example, longest common prefix of "abcdefgh" and "abcefgh" is "abc".

Input 1:

A = ["abcdefgh", "aefghijk", "abcefgh"]

Output 1:

"a"

Explanation 1:

Longest common prefix of all the strings is "a".

Input 2:

A = ["abab", "ab", "abcd"];

Output 2:

"ab"

Explanation 2:

Longest common prefix of all the strings is "ab".



**Count And Say**

The count-and-say sequence is the sequence of integers beginning as follows:

1, 11, 21, 1211, 111221, ...

1 is read off as **one 1** or 11.  
11 is read off as **two 1s** or 21.

21 is read off as **one 2, then one 1** or 1211.

Given an integer n, generate the nth sequence.

Note: The sequence of integers will be represented as a string.

Example:

if n = 2,  
the sequence is 11.



**Amazing Subarrays**

You are given a string **S**, and you have to find all the **amazing substrings** of **S**.

Amazing Substring is one that starts with a **vowel** (a, e, i, o, u, A, E, I, O, U).

Input

Only argument given is string S.

Output

Return a single integer X mod 10003, here X is number of Amazing Substrings in given string.

Example

Input

ABEC

Output

6

Explanation

Amazing substrings of given string are :

1. A

2. AB

3. ABE

4. ABEC

5. E

6. EC

here number of substrings are 6 and 6 % 10003 = 6.



**Stringoholics**

You are given an array **A** consisting of strings made up of the letters ‘a’ and ‘b’ only.   
Each string goes through a number of operations, where:

1. At time 1, you circularly rotate each string by 1 letter.

2. At time 2, you circularly rotate the new rotated strings by 2 letters.

3. At time 3, you circularly rotate the new rotated strings by 3 letters.

4. At time i, you circularly rotate the new rotated strings by i % length(string) letters.

Eg: String is "abaa"

1. At time 1, string is "baaa", as 1 letter is circularly rotated to the back

2. At time 2, string is "aaba", as 2 letters of the string "baaa" is circularly rotated to the back

3. At time 3, string is "aaab", as 3 letters of the string "aaba" is circularly rotated to the back

4. At time 4, string is again "aaab", as 4 letters of the string "aaab" is circularly rotated to the back

5. At time 5, string is "aaba", as 1 letters of the string "aaab" is circularly rotated to the back

After some units of time, a string becomes equal to it’s original self.   
Once a string becomes equal to itself, it’s letters start to rotate from the first letter again (**process resets**). So, if a string takes **t** time to get back to the original, at time t+1 one letter will be rotated and the string will be it’s original self at 2**t** time.   
You have to find the minimum time, where maximum number of strings are equal to their original self.   
As this time can be very large, give the answer modulo 109+7.

Example:

Input

A: [a,ababa,aba]

Output

4

String 'a' is it's original self at time 1, 2, 3 and 4.

String 'ababa' is it's original self only at time 4. (ababa => babaa => baaba => babaa => ababa)

String 'aba' is it's original self at time 2 and 4. (aba => baa => aba)

Hence, 3 strings are their original self at time 4.

**Minimum Characters required to make a String Palindromic**

Given an string **A**. The only operation allowed is to insert characters in the beginning of the string.

Find how many minimum characters are needed to be inserted to make the string a palindrome string.

Input Format

The only argument given is string A.

Output Format

Return the minimum characters that are needed to be inserted to make the string a palindrome string.

For Example

Input 1:

A = "ABC"

Output 1:

2

Explanation 1:

Insert 'B' at beginning, string becomes: "BABC".

Insert 'C' at beginning, string becomes: "CBABC".

Input 2:

A = "AACECAAAA"

Output 2:

2

Explanation 2:

Insert 'A' at beginning, string becomes: "AAACECAAAA".

Insert 'A' at beginning, string becomes: "AAAACECAAAA".



Longest Palindromic Substring

Given a string S, find the longest palindromic substring in S.

Substring of string S:

S[i...j] where 0 <= i <= j < len(S)

Palindrome string:

A string which reads the same backwards. More formally, S is palindrome if reverse(S) = S.

**Incase of conflict**, return the substring which occurs first ( with the least starting index ).

Example :

Input : "aaaabaaa"

Output : "aaabaaa"



**Implement StrStr**

Implement strStr().

strstr - locate a substring ( needle ) in a string ( haystack ).

Try not to use standard library string functions for this question.

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

NOTE:

Good clarification questions:

1. What should be the return value if the needle is empty?
2. What if both haystack and needle are empty?

For the purpose of this problem, assume that the return value should be -1 in both cases.



**Compare Version Numbers**

Compare two version numbers version1 and version2.

* If version1 > version2 return 1,
* If version1 < version2 return -1,
* otherwise return 0.

You may assume that the version strings are non-empty and contain only digits and the . character.  
The . character does not represent a decimal point and is used to separate number sequences.  
For instance, 2.5 is not "two and a half" or "half way to version three", it is the fifth second-level revision of the second first-level revision.

Here is an example of version numbers ordering:

0.1 < 1.1 < 1.2 < 1.13 < 1.13.4



**Atoi**

Implement atoi to convert a string to an integer.

Example :

Input : "9 2704"

Output : 9

Note: There might be multiple corner cases here. Clarify all your doubts using “See Expected Output”.

Questions:

**Q1.** Does string contain whitespace characters before the number?  
**A.** Yes

**Q2.** Can the string have garbage characters after the number?  
**A.** Yes. Ignore it.

**Q3.** If no numeric character is found before encountering garbage characters, what should I do?  
**A.** Return 0.

**Q4.** What if the integer overflows?  
**A.** Return INT\_MAX if the number is positive, INT\_MIN otherwise.



**Valid Number**

Validate if a given string is numeric.

Examples:

* "0" => true
* " 0.1 " => true
* "abc" => false
* "1 a" => false
* "2e10" => true

Return 0 / 1 ( 0 for false, 1 for true ) for this problem

Clarify the question using “See Expected Output”

* Is 1u ( which may be a representation for unsigned integers valid?  
  *For this problem, no.*
* Is 0.1e10 valid?  
  *Yes*
* -01.1e-10?  
  *Yes*
* Hexadecimal numbers like 0xFF?  
  Not for the purpose of this problem
* 3. (. not followed by a digit)?  
  *No*
* Can exponent have decimal numbers? 3e0.1?  
  *Not for this problem.*
* Is 1f ( floating point number with f as prefix ) valid?  
  *Not for this problem.*
* How about 1000LL or 1000L ( C++ representation for long and long long numbers )?  
  Not for this problem.
* How about integers preceded by 00 or 0? like 008?  
  *Yes for this problem*



**Valid Ip Addresses**

Given a string containing only digits, restore it by returning all possible valid IP address combinations.

A valid IP address must be in the form of A.B.C.D, where A,B,C and D are numbers from 0-255. The numbers cannot be 0 prefixed unless they are 0.

Example:

Given “25525511135”,

return [“255.255.11.135”, “255.255.111.35”]. (Make sure the returned strings are sorted in order)







**Length of Last Word**

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

If the last word does not exist, return 0.

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

Example:

Given s = "Hello World",

return 5 as length("World") = 5.



**Reverse the String**

Given a string **A**.

Return the string **A** after reversing the string word by word.

NOTE:

1. A sequence of non-space characters constitutes a word.
2. Your reversed string should not contain leading or trailing spaces, even if it is present in the input string.
3. If there are multiple spaces between words, reduce them to a single space in the reversed string.

Input 1:

A = "the sky is blue"

Output 1:

"blue is sky the"

Input 2:

A = "this is ib"

Output 2:

"ib is this"



**Roman To Integer**

Given a string **A** representing a roman numeral.  
Convert **A** into integer.

**A** is guaranteed to be within the range from **1** to **3999**.

**NOTE**: Read more   
details about roman numerals at [Roman Numeric System](https://en.wikipedia.org/wiki/Roman_numerals#Roman_numeric_system)

Input 1:

A = "XIV"

Output 1:

14

Input 2:

A = "XX"

Output 2:

20



**Integer To Roman**

Given an integer **A**, convert it to a roman numeral, and return a string corresponding to its roman numeral version

Input 1:

A = 5

Output 1:

"V"

Input 2:

A = 14

Output 2:

"XIV"



**Add Binary Strings**

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

Example:

a = "100"

b = "11"

Return a + b = “111”.



**Power of 2**

Find if Given number is power of 2 or not.   
More specifically, find if given number can be expressed as 2^k where k >= 1.

Input : 128

Output : 1



**Multiply Strings**

Given two numbers represented as strings, return multiplication of the numbers as a string.

**Note:** The numbers can be arbitrarily large and are non-negative.  
**Note2:** Your answer should not have leading zeroes. For example, 00 is not a valid answer.

For example,   
given strings "12", "10", your answer should be “120”.



**Justified Text**

Given an array of words and a length L, format the text such that each line has exactly L characters and is fully *(left and right)* justified.  
You should pack your words in a greedy approach; that is, pack as many words as you can in each line.

Pad extra spaces ‘ ‘ when necessary so that each line has exactly L characters.  
Extra spaces between words should be distributed as evenly as possible.  
If the number of spaces on a line do not divide evenly between words, the empty slots on the left will be assigned more spaces than the slots on the right.  
For the last line of text, it should be left justified and no extra space is inserted between words.

Your program should return a list of strings, where each string represents a single line.

Example:

words: ["This", "is", "an", "example", "of", "text", "justification."]

L: 16.

Return the formatted lines as:

[

"This is an",

"example of text",

"justification. "

]



**Zigzag String**

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

P.......A........H.......N

..A..P....L....S....I...I....G

....Y.........I........R

And then read line by line: PAHNAPLSIIGYIR  
Write the code that will take a string and make this conversion given a number of rows:

string convert(string text, int nRows);

convert("PAYPALISHIRING", 3) should return "PAHNAPLSIIGYIR"

\*\*Example 2 : \*\*  
ABCD, 2 can be written as

A....C

...B....D

and hence the answer would be ACBD.



**Pretty Json**

Given a string **A** representating json object. Return an array of string denoting json object with proper indentaion.

Rules for proper indentaion:

* Every inner brace should increase one indentation to the following lines.
* Every close brace should decrease one indentation to the same line and the following lines.
* The indents can be increased with an additional ‘\t’

Note:

1. [] and {} are only acceptable braces in this case.
2. Assume for this problem that space characters can be done away with.

Input Format

The only argument given is the integer array A.

Output Format

Return a list of strings, where each entry corresponds to a single line. The strings should not have "\n" character in them.

For Example

Input 1:

A = "{A:"B",C:{D:"E",F:{G:"H",I:"J"}}}"

Output 1:

{

A:"B",

C:

{

D:"E",

F:

{

G:"H",

I:"J"

}

}

}

Input 2:

A = ["foo", {"bar":["baz",null,1.0,2]}]

Output 2:

[

"foo",

{

"bar":

[

"baz",

null,

1.0,

2

]

}

]



Bit Manipulation

**Min XOR value**

Given an integer array **A** of **N** integers, find the pair of integers in the array which have minimum XOR value. Report the minimum XOR value.

Example Input 1:

A = [0, 2, 5, 7]

Example Output 1:

2

Explanation:

0 xor 2 = 2

Example Input 2:

A = [0, 4, 7, 9]

Example Output 2:

3



**Single Number**

Given an array of integers, every element appears twice except for one. Find that single one.

Example Input 1:

A = [1, 2, 2, 3, 1]

Example Output 1:

3

Explanation:

3 occurs only once

Example Input 2:

A = [1, 2, 2]

Example Output 2:

1



**Single Number II**

Given an array of integers, every element appears thrice except for one which occurs once.

Find that element which does not appear thrice.

Example Input 1:

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

Example Output 1:

4

Explanation:

4 occur exactly once

Example Input 2:

A = [0, 0, 0, 1]

Example Output 2:

1



**Number of 1 Bits**

Write a function that takes an unsigned integer and returns the number of 1 bits it has.

Example:

The 32-bit integer 11 has binary representation

00000000000000000000000000001011

**Reverse Bits**

Reverse the bits of an 32 bit unsigned integer A. Example Input 1:

A = 0

Example Output 1:

0

Explanation 1:

00000000000000000000000000000000

=> 00000000000000000000000000000000

Example Input 2:

A = 3

Example Output 2:

3221225472

Explanation 2:

00000000000000000000000000000011

=> 11000000000000000000000000000000



**Divide Integers**

Divide two integers without using multiplication, division and mod operator.

Return the floor of the result of the division.

Example:

5 / 2 = 2



**Different Bits Sum Pairwise**

We define f(X, Y) as number of different corresponding bits in binary representation of X and Y. For example, f(2, 7) = 2, since binary representation of 2 and 7 are 010 and 111, respectively. The first and the third bit differ, so f(2, 7) = 2.

You are given an array of N positive integers, A1, A2 ,…, AN. Find sum of f(Ai, Aj) for all pairs (i, j) such that 1 ≤ i, j ≤ N. Return the answer modulo 109+7.

For example,

A=[1, 3, 5]

We return

f(1, 1) + f(1, 3) + f(1, 5) +

f(3, 1) + f(3, 3) + f(3, 5) +

f(5, 1) + f(5, 3) + f(5, 5) =

0 + 1 + 1 +

1 + 0 + 2 +

1 + 2 + 0 = 8



**Kth Bit Manipulation**



Two Pointer

**Merge Two Sorted Lists II**

Given two sorted integer arrays A and B, merge B into A as one sorted array.

**Note**: You have to modify the array A to contain the merge of A and B. Do not output anything in your code.  
**TIP**: C users, please malloc the result into a new array and return the result.

If the number of elements initialized in A and B are m and n respectively, the resulting size of array A after your code is executed should be m + n

**Example :**

Input :

A : [1 5 8]

B : [6 9]

Modified A : [1 5 6 8 9]



**Intersection Of Sorted Arrays**

Find the intersection of two sorted arrays.  
**OR in other words,**  
Given 2 sorted arrays, find all the elements which occur in both the arrays.

**Example :**

Input :

A : [1 2 3 3 4 5 6]

B : [3 3 5]

Output : [3 3 5]

Input :

A : [1 2 3 3 4 5 6]

B : [3 5]

Output : [3 5]



**3 Sum**

Given an array S of n integers, find three integers in S such that the sum is closest to a given number, target.   
Return the sum of the three integers.

*Assume that there will only be one solution*

**Example:**   
given array S = {-1 2 1 -4},   
and target = 1.

The sum that is closest to the target is 2. (-1 + 2 + 1 = 2)



**3 Sum Zero**

Given an array S of n integers, are there elements a, b, c in S such that a + b + c = 0?   
Find all unique triplets in the array which gives the sum of zero.

**Note:**

Elements in a triplet (a,b,c) must be in non-descending order. (ie, a ≤ b ≤ c)  
The solution set must not contain duplicate triplets.

For example, given array S = {-1 0 1 2 -1 -4},

A solution set is:  
(-1, 0, 1)  
(-1, -1, 2)



**Diffk**

Given an array ‘A’ of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[i] - A[j] = k, i != j.

**Example:**

Input :

A : [1 3 5]

k : 4

Output : YES

as 5 - 1 = 4

Return 0 / 1 ( 0 for false, 1 for true ) for this problem



**Remove Duplicates from Sorted Array**

Remove duplicates from Sorted Array  
Given a sorted array, remove the duplicates in place such that each element appears only once and return the new length.

**Note that even though we want you to return the new length, make sure to change the original array as well in place**

Do not allocate extra space for another array, you must do this in place with constant memory.

**Example:**   
Given input array A = [1,1,2],  
Your function should return length = 2, and A is now [1,2].



**Remove Duplicates from Sorted Array II**

**Remove Duplicates from Sorted Array**

Given a sorted array, remove the duplicates in place such that each element can appear atmost twice and return the new length.

Do not allocate extra space for another array, you must do this in place with constant memory.

**Note that even though we want you to return the new length, make sure to change the original array as well in place**

For example,  
Given input array A = [1,1,1,2],

Your function should return length = 3, and A is now [1,1,2].



**Remove Element from Array**

Given an array and a value, remove all the instances of that value in the array.   
Also return the number of elements left in the array after the operation.  
It does not matter what is left beyond the expected length.

**Example:**  
If array A is [4, 1, 1, 2, 1, 3]  
and value elem is 1,   
then new length is 3, and A is now [4, 2, 3]



**Sort by Color**

Given an array with n objects colored red, white or blue,   
sort them so that objects of the same color are adjacent, with the colors in the order red, white and blue.

Here, we will use the integers 0, 1, and 2 to represent the color red, white, and blue respectively.

*Note: Using library sort function is not allowed.*

**Example :**

Input : [0 1 2 0 1 2]

Modify array so that it becomes : [0 0 1 1 2 2]



**Max Continuous Series of 1s**

You are given with an array of 1s and 0s. And you are given with an integer M, which signifies number of flips allowed.  
Find the position of zeros which when flipped will produce maximum continuous series of 1s.

**For this problem, return the indices of maximum continuous series of 1s in order.**

**Example:**

Input :

Array = {1 1 0 1 1 0 0 1 1 1 }

M = 1

Output :

[0, 1, 2, 3, 4]



**Array 3 Pointers**

You are given 3 arrays A, B and C. All 3 of the arrays are sorted.

Find i, j, k such that :  
max(abs(A[i] - B[j]), abs(B[j] - C[k]), abs(C[k] - A[i])) is minimized.  
Return the minimum max(abs(A[i] - B[j]), abs(B[j] - C[k]), abs(C[k] - A[i]))

\*\*abs(x) is absolute value of x and is implemented in the following manner : \*\*

if (x < 0) return -x;

else return x;

**Example :**

Input :

A : [1, 4, 10]

B : [2, 15, 20]

C : [10, 12]

Output : 5

With 10 from A, 15 from B and 10 from C.



**Container With Most Water**

Given **n non-negative integers a1, a2, ..., an,**  
where each represents a point at coordinate (i, ai).  
'n' vertical lines are drawn such that the two endpoints of line i is at (i, ai) and (i, 0).

Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Your program should return an integer which corresponds to the maximum area of water that can be contained ( Yes, we know maximum area instead of maximum volume sounds weird. But this is 2D plane we are working with for simplicity ).

**Note:** You may not slant the container.

**Example :**

Input : [1, 5, 4, 3]

Output : 6

Explanation : 5 and 3 are distance 2 apart. So size of the base = 2. Height of container = min(5, 3) = 3.

So total area = 3 \* 2 = 6



TODO:

**Counting Triangles**

**Minimize the absolute difference**

Linked List

**Intersection of Linked Lists**

Write a program to find the node at which the intersection of two singly linked lists begins.

For example, the following two linked lists:

A: a1 → a2

↘

c1 → c2 → c3

↗

B: b1 → b2 → b3



**Reverse Linked List**

Reverse a linked list. Do it in-place and in one-pass.

For example:  
Given 1->2->3->4->5->NULL,

return 5->4->3->2->1->NULL.



**Palindrome List**

Given a singly linked list, determine if its a palindrome. Return 1 or 0 denoting if its a palindrome or not, respectively.

**Notes**:

* Expected solution is linear in time and constant in space.

For example,

List 1-->2-->1 is a palindrome.

List 1-->2-->3 is not a palindrome.



**Remove Duplicates from Sorted List**

Given a sorted linked list, delete all duplicates such that each element appear only once.

For example,  
Given 1->1->2, return 1->2.  
Given 1->1->2->3->3, return 1->2->3.



**Remove Duplicates from Sorted List II**

Given a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list.

For example,  
Given 1->2->3->3->4->4->5, return 1->2->5.  
Given 1->1->1->2->3, return 2->3.



**Merge Two Sorted Lists**

Merge two sorted linked lists and return it as a new list.   
The new list should be made by splicing together the nodes of the first two lists, and should also be sorted.

For example, given following linked lists :

5 -> 8 -> 20

4 -> 11 -> 15

The merged list should be :

4 -> 5 -> 8 -> 11 -> 15 -> 20



**Remove Nth Node from List End**

Given a linked list, remove the nth node from the end of list and return its head.

For example,  
Given linked list: 1->2->3->4->5, and n = 2.  
After removing the second node from the end, the linked list becomes 1->2->3->5.

**Note:**

* If n is greater than the size of the list, remove the first node of the list.



**Rotate List**

Given a list, rotate the list to the right by k places, where k is non-negative.

For example:

Given 1->2->3->4->5->NULL and k = 2,  
return 4->5->1->2->3->NULL.



**Reverse Link List II**

Reverse a linked list from position m to n. Do it in-place and in one-pass.

For example:  
Given 1->2->3->4->5->NULL, m = 2 and n = 4,

return 1->4->3->2->5->NULL.



**Reorder List**

Given a singly linked list

L: L0 → L1 → … → Ln-1 → Ln,

reorder it to:

L0 → Ln → L1 → Ln-1 → L2 → Ln-2 → …

You must do this in-place without altering the nodes’ values.

For example,  
Given {1,2,3,4}, reorder it to {1,4,2,3}.



**K reverse linked list**

Given a singly linked list and an integer *K*, reverses the nodes of the

list *K* at a time and returns modified linked list.

**NOTE :** The length of the list is divisible by K

**Example :**

Given linked list 1 -> 2 -> 3 -> 4 -> 5 -> 6 and K=2,

You should return 2 -> 1 -> 4 -> 3 -> 6 -> 5



**Swap List Nodes in pairs**

Given a linked list, swap every two adjacent nodes and return its head.

For example,  
Given 1->2->3->4, you should return the list as 2->1->4->3.

Your algorithm should use only constant space. You may not modify the values in the list, only nodes itself can be changed.



**Add Two Numbers as Lists**

You are given two linked lists representing two non-negative numbers. The digits are stored in **reverse order** and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)  
Output: 7 -> 0 -> 8

342 + 465 = 807

Make sure there are no trailing zeros in the output list  
So, 7 -> 0 -> 8 -> 0 is not a valid response even though the value is still 807.



**List Cycle**

Given a linked list, return the node where the cycle begins. If there is no cycle, return null.

Try solving it using constant additional space.

**Example :**

Input :

\_\_\_\_\_\_

| |

\/ |

1 -> 2 -> 3 -> 4

Return the node corresponding to node 3.



**Partition List**

Given a linked list and a value x, partition it such that all nodes less than x come before nodes greater than or equal to x.

You should preserve the original relative order of the nodes in each of the two partitions.

For example,  
Given 1->4->3->2->5->2 and x = 3,  
return 1->2->2->4->3-

**Insertion Sort List**

Sort a linked list using insertion sort.

We have explained Insertion Sort at Slide 7 of [Arrays](http://www.interviewbit.com/courses/programming/topics/arrays/)

[Insertion Sort Wiki](http://en.wikipedia.org/wiki/Insertion_sort#Algorithm) has some details on Insertion Sort as well.

**Example :**

Input : 1 -> 3 -> 2

Return 1 -> 2 -> 3



**Sort List**

Sort a linked list in O(n log n) time using constant space complexity.

Example :

Input : 1 -> 5 -> 4 -> 3

Returned list : 1 -> 3 -> 4 -> 5



Stack

**Generate all Parentheses**

Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

The brackets must close in the correct order, "()" and "()[]{}" are all valid but "(]" and "([)]" are not.

Return 0 / 1 ( 0 for false, 1 for true ) for this problem



**Reverse String**

Given a string S, reverse the string using stack.

**Example :**

Input : "abc"

Return "cba"



**Simplify Directory Path**

Given a string **A** representing an absolute path for a file (Unix-style).

Return the string A after simplifying the absolute path.

**Note**:

1. Absolute path always begin with **’/’** ( root directory ).
2. Path will not have whitespace characters.
3. Input 1:
4. A = "/home/"
5. Output 1:
6. "/home"
7. Input 2:
8. A = "/a/./b/../../c/"
9. Output 2:
10. "/c"



**Redundant Braces**

Given a string **A** denoting an expression. It contains the following operators **’+’, ‘-‘, ‘\*’, ‘/’**.

Chech whether **A** has redundant braces or not.

Return **1** if **A** has redundant braces, else return **0**.

Input 1:

A = "((a + b))"

Output 1:

1

Explanation 1:

((a + b)) has redundant braces so answer will be 1.

Input 2:

A = "(a + (a + b))"

Output 2:

0

Explanation 2:

(a + (a + b)) doesn't have have any redundant braces so answer will be 0.



**Nearest Smaller Element**

Given an array, find the **nearest** smaller element G[i] for every element A[i] in the array such that the element has an **index smaller than i**.

More formally,

G[i] for an element A[i] = an element A[j] such that

j is maximum possible AND

j < i AND

A[j] < A[i]

Elements for which no smaller element exist, consider next smaller element as -1.

Input 1:

A = [4, 5, 2, 10, 8]

Output 1:

G = [-1, 4, -1, 2, 2]

Explaination 1:

index 1: No element less than 4 in left of 4, G[1] = -1

index 2: A[1] is only element less than A[2], G[2] = A[1]

index 3: No element less than 2 in left of 2, G[3] = -1

index 4: A[3] is nearest element which is less than A[4], G[4] = A[3]

index 4: A[3] is nearest element which is less than A[5], G[5] = A[3]

Input 2:

A = [3, 2, 1]

Output 2:

[-1, -1, -1]

Explaination 2:

index 1: No element less than 3 in left of 3, G[1] = -1

index 2: No element less than 2 in left of 2, G[2] = -1

index 3: No element less than 1 in left of 1, G[3] = -1



**Largest Rectangle in Histogram**

Given an array of integers **A** of size **N**. **A** represents a histogram i.e **A[i]** denotes height of   
the **ith** histogram’s bar. Width of each bar is **1**.



Above is a histogram where width of each bar is 1, given height = [2,1,5,6,2,3].



The largest rectangle is shown in the shaded area, which has area = 10 unit.

Find the area of largest rectangle in the histogram.

**Input Format**

The only argument given is the integer array A.

**Output Format**

Return the area of largest rectangle in the histogram.

**For Example**

Input 1:

A = [2, 1, 5, 6, 2, 3]

Output 1:

10

Explanation 1:

The largest rectangle is shown in the shaded area, which has area = 10 unit.



**Sliding Window Maximum**

Given an array of integers **A**. There is a sliding window of size **B** which   
is moving from the very left of the array to the very right.   
You can only see the w numbers in the window. Each time the sliding window moves   
rightwards by one position. You have to find the maximum for each window.   
The following example will give you more clarity.

The array **A** is [1 3 -1 -3 5 3 6 7], and **B** is 3.

|  |  |
| --- | --- |
| **Window position** | **Max** |
| ———————————- | ————————- |
| [1 3 -1] -3 5 3 6 7 | 3 |
| 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 | 5 |
| 1 3 -1 -3 [5 3 6] 7 | 6 |
| 1 3 -1 -3 5 [3 6 7] | 7 |

Return an array **C**, where **C[i]** is the maximum value of from **A[i]** to **A[i+B-1]**.

**Note**: If **B** > length of the array, return 1 element with the max of the array.

**Input Format**

The first argument given is the integer array A.

The second argument given is the integer B.

**Output Format**

Return an array C, where C[i] is the maximum value of from A[i] to A[i+B-1]

**For Example**

Input 1:

A = [1, 3, -1, -3, 5, 3, 6, 7]

B = 3

Output 1:

C = [3, 3, 5, 5, 6, 7]



**Evaluate Expression**

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are +, -, \*, /. Each operand may be an integer or another expression.

**Input Format**

The only argument given is character array A.

**Output Format**

Return the value of arithmetic expression formed using reverse Polish Notation.

**For Example**

Input 1:

A = ["2", "1", "+", "3", "\*"]

Output 1:

9

Explaination 1:

starting from backside:

\*: ( )\*( )

3: ()\*(3)

+: ( () + () )\*(3)

1: ( () + (1) )\*(3)

2: ( (2) + (1) )\*(3)

((2)+(1))\*(3) = 9

Input 2:

A = ["4", "13", "5", "/", "+"]

Output 2:

6

Explaination 2:

+: ()+()

/: ()+(() / ())

5: ()+(() / (5))

1: ()+((13) / (5))

4: (4)+((13) / (5))

(4)+((13) / (5)) = 6



**Rain Water Trapped**

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

**Input Format**

The only argument given is integer array A.

**Output Format**

Return the total water it is able to trap after raining..

**For Example**

Input 1:

A = [0,1,0,2,1,0,1,3,2,1,2,1]

Output 1:

6

Explaination 1: <img src="http://i.imgur.com/0qkUFco.png">

In this case, 6 units of rain water (blue section) are being trapped.



**Min Stack**

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

* **push(x)** – Push element x onto stack.
* **pop()** – Removes the element on top of the stack.
* **top()** – Get the top element.
* **getMin()** – Retrieve the minimum element in the stack.

Note that all the operations have to be constant time operations.

Questions to ask the interviewer :

Q: What should getMin() do on empty stack?

A: In this case, return -1.

Q: What should pop do on empty stack?

A: In this case, nothing.

Q: What should top() do on empty stack?

A: In this case, return -1



Hashing

**Colorful Number**

For Given Number N find if its **COLORFUL** number or not

**Return 0/1**

**COLORFUL number:**

A number can be broken into different contiguous sub-subsequence parts.

Suppose, a number 3245 can be broken into parts like 3 2 4 5 32 24 45 324 245.

And this number is a COLORFUL number, since product of every digit of a contiguous subsequence is different

**Example:**

N = 23

2 3 23

2 -> 2

3 -> 3

23 -> 6

this number is a COLORFUL number since product of every digit of a sub-sequence are different.

Output : 1



**Largest Continuous Sequence Zero Sum**

Find the largest continuous sequence in a array which sums to zero.

**Example:**

Input: {1 ,2 ,-2 ,4 ,-4}

Output: {2 ,-2 ,4 ,-4}



**2 Sum**

Given an array of integers, find two numbers such that they add up to a specific target number.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 < index2. Please note that your returned answers (both index1 and index2 ) are not zero-based.   
Put both these numbers in order in an array and return the array from your function ( Looking at the function signature will make things clearer ). Note that, if no pair exists, return empty list.

If multiple solutions exist, output the one where index2 is minimum. If there are multiple solutions with the minimum index2, choose the one with minimum index1 out of them.

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

Output: index1 = 1, index2 = 2



**4 Sum**

Given an array S of n integers, are there elements a, b, c, and d in S such that a + b + c + d = target? Find all unique quadruplets in the array which gives the sum of target.

**Note:**

* Elements in a quadruplet (a,b,c,d) must be in non-descending order. (ie, a ≤ b ≤ c ≤ d)
* The solution set must not contain duplicate quadruplets.

**Example :**   
Given array S = {1 0 -1 0 -2 2}, and target = 0  
A solution set is:

(-2, -1, 1, 2)

(-2, 0, 0, 2)

(-1, 0, 0, 1)



**Anagrams**

Given an array of strings, return all groups of strings that are anagrams. Represent a group by a list of integers representing the index in the original list. Look at the sample case for clarification.

**Anagram :** a word, phrase, or name formed by rearranging the letters of another, such as 'spar', formed from 'rasp'

**Note:** All inputs will be in lower-case.

**Example :**

Input : cat dog god tca

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

cat and tca are anagrams which correspond to index 1 and 4.   
dog and god are another set of anagrams which correspond to index 2 and 3.  
The indices are 1 based ( the first element has index 1 instead of index 0).

**Ordering of the result :** You should not change the relative ordering of the words / phrases within the group. Within a group containing A[i] and A[j], A[i] comes before A[j] if i < j.



**Equal**

Given an array A of integers, find the index of values that satisfy A + B = C + D, where A,B,C & D are integers values in the array

**Note:**

1) Return the indices `A1 B1 C1 D1`, so that

A[A1] + A[B1] = A[C1] + A[D1]

A1 < B1, C1 < D1

A1 < C1, B1 != D1, B1 != C1

2) If there are more than one solutions,

then return the tuple of values which are lexicographical smallest.

Assume we have two solutions

S1 : A1 B1 C1 D1 ( these are values of indices int the array )

S2 : A2 B2 C2 D2

S1 is lexicographically smaller than S2 iff

A1 < A2 OR

A1 = A2 AND B1 < B2 OR

A1 = A2 AND B1 = B2 AND C1 < C2 OR

A1 = A2 AND B1 = B2 AND C1 = C2 AND D1 < D2

**Example:**

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

Output: [0, 2, 3, 5] (O index)

If no solution is possible, return an empty list.



**Copy List**

A linked list is given such that each node contains an additional random pointer which could point to any node in the list or NULL.

Return a deep copy of the list.

**Example**

Given list

1 -> 2 -> 3

with random pointers going from

1 -> 3

2 -> 1

3 -> 1

You should return a deep copy of the list. The returned answer should not contain the same node as the original list, but a copy of them. The pointers in the returned list should not link to any node in the original input list.



**Longest Substring Without Repeat**

Given a string,   
find the length of the longest substring without repeating characters.

**Example:**

The longest substring without repeating letters for "abcabcbb" is "abc", which the length is 3.

For "bbbbb" the longest substring is "b", with the length of 1.



**Window String**

Given a string S and a string T, find the minimum window in S which will contain all the characters in T in linear time complexity.  
Note that when the count of a character C in T is N, then the count of C in minimum window in S should be at least N.

**Example :**

S = "ADOBECODEBANC"

T = "ABC"

Minimum window is "BANC"



**Fraction**

Given two integers representing the numerator and denominator of a fraction, return the fraction in string format.

If the fractional part is repeating, enclose the repeating part in parentheses.

**Example :**

Given numerator = 1, denominator = 2, return "0.5"

Given numerator = 2, denominator = 1, return "2"

Given numerator = 2, denominator = 3, return "0.(6)"



**Points on the Straight Line**

Given n points on a 2D plane, find the maximum number of points that lie on the same straight line.

Sample Input :

(1, 1)

(2, 2)

Sample Output :

2

You will be given 2 arrays X and Y. Each point is represented by (X[i], Y[i])



**Substring Concatenation**

You are given a string, S, and a list of words, L, that are all of the same length.

Find all starting indices of substring(s) in S that is a concatenation of each word in L exactly once and without any intervening characters.

**Example :**

S: "barfoothefoobarman"

L: ["foo", "bar"]

You should return the indices: [0,9].



TODO:

valid-sudoku

diffk-ii

Backtracking

**Reverse Link List Recursion**

Reverse a linked list using recursion.

**Example :**  
Given 1->2->3->4->5->NULL,

return 5->4->3->2->1->NULL.



**Subset**

Given a set of distinct integers, S, return all possible subsets.

**Note:**

* Elements in a subset must be in non-descending order.
* The solution set must not contain duplicate subsets.
* Also, the subsets should be sorted in ascending ( lexicographic ) order.
* The list is not necessarily sorted.

**Example :**

If S = [1,2,3], a solution is:

[

[],

[1],

[1, 2],

[1, 2, 3],

[1, 3],

[2],

[2, 3],

[3],

]



**Combinations**

Given two integers n and k, return all possible combinations of k numbers out of 1 2 3 ... n.

Make sure the combinations are **sorted**.

To elaborate,

1. Within every entry, elements should be sorted. [1, 4] is a valid entry while [4, 1] is not.
2. Entries should be sorted within themselves.

**Example :**  
If n = 4 and k = 2, a solution is:

[

[1,2],

[1,3],

[1,4],

[2,3],

[2,4],

[3,4],

]

**public** ArrayList<ArrayList<Integer>> combine(**final int** A, **final int** B) {  
 **final** ArrayList<ArrayList<Integer>> ans = **new** ArrayList<>();  
 **if** (A < B) {  
 **return** ans;  
 }  
  
 **final** ArrayList<Integer> temp = **new** ArrayList<>();  
 **this**.combineHelper(A, B, ans, temp, 1);  
 **return** ans;  
}  
  
**private void** combineHelper(**final int** a, **final int** b, **final** ArrayList<ArrayList<Integer>> ans,  
 **final** ArrayList<Integer> temp, **final int** start) {  
 *// this block is just for performance improvement, we can remove this if block* **if** (temp.size() > b) {  
 **return**;  
 }  
 **if** ((temp.size() == b)) {  
 ans.add(**new** ArrayList<>(temp));  
 }  
 **for** (**int** i = start; i <= a; i++) {  
 **if** (temp.contains(i)) {  
 **continue**;  
 }  
 *// Choose* temp.add(i);  
 *// Explore* **this**.combineHelper(a, b, ans, temp, i + 1);  
 *// Un-choose* temp.remove(temp.size() - 1);  
 }  
}

**Combination Sum**

Given a set of candidate numbers (C) and a target number (T), find all unique combinations in C where the candidate numbers sums to T.

The same repeated number may be chosen from C unlimited number of times.

**Note:**

* All numbers (including target) will be positive integers.
* Elements in a combination (a1, a2, … , ak) must be in non-descending order. (ie, a1 ≤ a2 ≤ … ≤ ak).
* The combinations themselves must be sorted in ascending order.
* CombinationA > CombinationB iff (a1 > b1) OR (a1 = b1 AND a2 > b2) OR … (a1 = b1 AND a2 = b2 AND … ai = bi AND ai+1 > bi+1)
* The solution set must not contain duplicate combinations.

**Example**,   
Given candidate set 2,3,6,7 and target 7,  
A solution set is:

[2, 2, 3]

[7]



**Combination Sum II**

Given a collection of candidate numbers (C) and a target number (T), find all unique combinations in C where the candidate numbers sums to T.

Each number in C may only be used **once** in the combination.

**Note:**

* All numbers (including target) will be positive integers.
* Elements in a combination (a1, a2, … , ak) must be in non-descending order. (ie, a1 ≤ a2 ≤ … ≤ ak).
* The solution set must not contain duplicate combinations.

**Example :**

Given candidate set 10,1,2,7,6,1,5 and target 8,

A solution set is:

[1, 7]

[1, 2, 5]

[2, 6]

[1, 1, 6]



**Subsets II**

Given a collection of integers that might contain duplicates, S, return all possible subsets.

**Note:**

* Elements in a subset must be in non-descending order.
* The solution set must not contain duplicate subsets.
* The subsets must be sorted lexicographically.

**Example :**  
If S = [1,2,2], the solution is:

[

[],

[1],

[1,2],

[1,2,2],

[2],

[2, 2]

]



**Letter Phone**

Given a digit string, return all possible letter combinations that the number could represent.

A mapping of digit to letters (just like on the telephone buttons) is given below.



The digit 0 maps to 0 itself.  
The digit 1 maps to 1 itself.

Input: Digit string "23"

Output: ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"].



**Palindrome Partitioning**

Given a string s, partition s such that every string of the partition is a palindrome.

Return all possible palindrome partitioning of s.

For example, given s = "aab",  
Return

[

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

["aa","b"],

]

**Ordering the results in the answer :**

Entry i will come before Entry j if :

* len(Entryi[0]) < len(Entryj[0]) OR
* (len(Entryi[0]) == len(Entryj[0]) AND len(Entryi[1]) < len(Entryj[1])) OR  
  \*  
  \*  
  \*
* (len(Entryi[0]) == len(Entryj[0]) AND … len(Entryi[k] < len(Entryj[k]))

In the given example,  
["a", "a", "b"] comes before ["aa", "b"] because len("a") < len("aa")



**Generate all Parentheses II**

Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses of length 2\*n.

For example, given n = 3, a solution set is:

"((()))", "(()())", "(())()", "()(())", "()()()"



**Permutations**

Given a collection of numbers, return all possible permutations.

**Example:**

[1,2,3] will have the following permutations:

[1,2,3]

[1,3,2]

[2,1,3]

[2,3,1]

[3,1,2]

[3,2,1]



**Gray Code**

The gray code is a binary numeral system where two successive values differ in only one bit.

Given a non-negative integer n representing the total number of bits in the code, print the sequence of gray code. A gray code sequence must begin with 0.

For example, given n = 2, return [0,1,3,2]. Its gray code sequence is:

00 - 0

01 - 1

11 - 3

10 - 2

There might be multiple gray code sequences possible for a given n.  
Return any such sequence.



**NQueens**

The n-queens puzzle is the problem of placing n queens on an n×n chessboard such that no two queens attack each other.



Given an integer n, return all distinct solutions to the n-queens puzzle.

Each solution contains a distinct board configuration of the n-queens’ placement, where 'Q' and '.' both indicate a queen and an empty space respectively.

For example,  
There exist two distinct solutions to the 4-queens puzzle:

[

[".Q..", // Solution 1

"...Q",

"Q...",

"..Q."],

["..Q.", // Solution 2

"Q...",

"...Q",

".Q.."]

]



**Sudoku**

Write a program to solve a Sudoku puzzle by filling the empty cells.  
Empty cells are indicated by the character '.'   
You may assume that there will be only one unique solution.



A sudoku puzzle,



and its solution numbers marked in red.

**Example :**

For the above given diagrams, the corresponding input to your program will be

[[53..7....], [6..195...], [.98....6.], [8...6...3], [4..8.3..1], [7...2...6], [.6....28.], [...419..5], [....8..79]]

and we would expect your program to modify the above array of array of characters to

[[534678912], [672195348], [198342567], [859761423], [426853791], [713924856], [961537284], [287419635], [345286179]]



Heaps and Maps

**N max pair combinations**

Given two arrays A & B of size N each.  
Find the maximum N elements from the sum combinations (Ai + Bj) formed from elements in array A and B.

For example if A = [1,2], B = [3,4], then possible pair sums can be 1+3 = 4 , 1+4=5 , 2+3=5 , 2+4=6  
and maximum 2 elements are 6, 5

**Example:**

N = 4  
a[]={1,4,2,3}  
b[]={2,5,1,6}

Maximum 4 elements of combinations sum are

10 (4+6),

9 (3+6),

9 (4+5),

8 (2+6)



**Magician and Chocolates**

Given **N** bags, each bag contains **Ai** chocolates. There is a kid and a magician. In one unit of time, kid chooses a random bag **i**, eats **Ai** chocolates, then the magician fills the ith bag with [*floor*](http://mathworld.wolfram.com/FloorFunction.html)(**Ai/2**) chocolates.

Given **Ai** for 1 <= *i* <= N, find the maximum number of chocolates kid can eat in **K** units of time.

For example,

K = 3

N = 2

A = 6 5

Return: 14

At t = 1 kid eats 6 chocolates from bag 0, and the bag gets filled by 3 chocolates  
At t = 2 kid eats 5 chocolates from bag 1, and the bag gets filled by 2 chocolates  
At t = 3 kid eats 3 chocolates from bag 0, and the bag gets filled by 1 chocolate  
so, total number of chocolates eaten: 6 + 5 + 3 = 14

**Note**: Return your answer modulo 10^9+7



**Merge K Sorted Lists**

Merge k sorted linked lists and return it as one sorted list.

**Example :**

1 -> 10 -> 20

4 -> 11 -> 13

3 -> 8 -> 9

will result in

1 -> 3 -> 4 -> 8 -> 9 -> 10 -> 11 -> 13 -> 20



**Distinct Numbers in Window**

You are given an array of N integers, A1, A2 ,…, AN and an integer K. Return the of count of distinct numbers in all windows of size K.

Formally, return an array of size N-K+1 where i’th element in this array contains number of distinct elements in sequence Ai, Ai+1 ,…, Ai+k-1.

**Note**:

If K > N, return empty array.

A[i] is a signed integer

For example,

A=[1, 2, 1, 3, 4, 3] and K = 3

All windows of size K are

[1, 2, 1]

[2, 1, 3]

[1, 3, 4]

[3, 4, 3]

So, we return an array [2, 3, 3, 2].



**LRU Cache**

Design and implement a data structure for LRU (Least Recently Used) cache. It should support the following operations: get and set.

1. get(key) - Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.
2. set(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 recently used item before inserting the new item.

The LRU Cache will be initialized with an integer corresponding to its capacity. Capacity indicates the maximum number of unique keys it can hold at a time.

**Definition of “least recently used”** : An access to an item is defined as a get or a set operation of the item. “Least recently used” item is the one with the oldest access time.

**NOTE:** If you are using any global variables, make sure to clear them in the constructor.

**Example :**

Input :

capacity = 2

set(1, 10)

set(5, 12)

get(5) returns 12

get(1) returns 10

get(10) returns -1

set(6, 14) this pushes out key = 5 as LRU is full.

get(5) returns -1



Todo:

**Ways to form Max Heap**

Tree Data Structure

**Next Greater Number BST**

Given a BST node, return the node which has value just greater than the given node.

**Example:**

Given the tree

100

/ \

98 102

/ \

96 99

\

97

Given 97, you should return the node corresponding to 98 as thats the value just greater than 97 in the tree.  
If there are no successor in the tree ( the value is the largest in the tree, return NULL).



**Hotel Reviews**

Given a set of reviews provided by the customers for different hotels and a string containing “Good Words”, you need to sort the reviews in descending order according to their “Goodness Value” (Higher goodness value first). We define the “Goodness Value” of a string as the number of “Good Words” in that string.

Note: Sorting should be stable. If review i and review j have the same “Goodness Value” then their original order would be preserved.

You are expected to use Trie in an Interview for such problems

**Constraints:**

1. 1 <= No.of reviews <= 200

2. 1 <= No. of words in a review <= 1000

3. 1 <= Length of an individual review <= 10,000

4. 1 <= Number of Good Words <= 10,000

5. 1 <= Length of an individual Good Word <= 4

6. All the alphabets are lower case (a - z)

**Input:**

S : A string S containing "Good Words" separated by "\_" character. (See example below)

R : A vector of strings containing Hotel Reviews. Review strings are also separated by "\_" character.

**Output:**

A vector V of integer which contain the original indexes of the reviews in the sorted order of reviews.

V[i] = k means the review R[k] comes at i-th position in the sorted order. (See example below)

In simple words, V[i]=Original index of the review which comes at i-th position in the sorted order. (Indexing is 0 based)

**Example:**

Input:

S = "cool\_ice\_wifi"

R = ["water\_is\_cool", "cold\_ice\_drink", "cool\_wifi\_speed"]

Output:

ans = [2, 0, 1]

Here, sorted reviews are ["cool\_wifi\_speed", "water\_is\_cool", "cold\_ice\_drink"]





**Balanced Binary Tree**

Given a binary tree, determine if it is height-balanced.

**Height-balanced binary tree** : is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

Return 0 / 1 ( 0 for false, 1 for true ) for this problem

**Example :**

Input :

1

/ \

2 3

Return : True or 1

Input 2 :

3

/

2

/

1

Return : False or 0

Because for the root node, left subtree has depth 2 and right subtree has depth 0.

Difference = 2 > 1.



**Identical Binary Trees**

Given two binary trees, write a function to check if they are equal or not.

Two binary trees are considered equal if they are structurally identical and the nodes have the same value.

Return 0 / 1 ( 0 for false, 1 for true ) for this problem

**Example :**

Input :

1 1

/ \ / \

2 3 2 3

Output :

1 or True



**Symmetric Binary Tree**

Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

**Example :**

1

/ \

2 2

/ \ / \

3 4 4 3

The above binary tree is symmetric.   
But the following is not:

1

/ \

2 2

\ \

3 3

Return 0 / 1 ( 0 for false, 1 for true ) for this problem



**Inorder Traversal of Cartesian Tree**

Given an inorder traversal of a cartesian tree, construct the tree.

**Cartesian tree** : is a heap ordered binary tree, where the root is greater than all the elements in the subtree.

**Note:** You may assume that duplicates do not exist in the tree.

**Example :**

Input : [1 2 3]

Return :

3

/

2

/

1



**Sorted Array To Balanced BST**

Given an array where elements are sorted in ascending order, convert it to a height balanced BST.

**Balanced tree :** a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

**Example :**

Given A : [1, 2, 3]

A height balanced BST :

2

/ \

1 3



**Binary Tree From Inorder And Postorder**

Given inorder and postorder traversal of a tree, construct the binary tree.

**Note:** You may assume that duplicates do not exist in the tree.

**Example :**

Input :

Inorder : [2, 1, 3]

Postorder : [2, 3, 1]

Return :

1

/ \

2 3



**Construct Binary Tree From Inorder And Preorder**

Given preorder and inorder traversal of a tree, construct the binary tree.

**Note:** You may assume that duplicates do not exist in the tree.

**Example :**

Input :

Preorder : [1, 2, 3]

Inorder : [2, 1, 3]

Return :

1

/ \

2 3



**Kth Smallest Element In Tree**

Given a binary search tree, write a function to find the kth smallest element in the tree.

**Example :**

Input :

2

/ \

1 3

and k = 2

Return : 2

As 2 is the second smallest element in the tree.



**2-Sum Binary Tree**

Given a binary search tree T, where each node contains a positive integer, and an integer K, you have to find whether or not there exist two different nodes A and B such that A.value + B.value = K.

Return 1 to denote that two such nodes exist. Return 0, otherwise.

**Notes**

* Your solution should run in linear time and not take memory more than O(height of T).
* Assume all values in BST are distinct.

**Example :**

Input 1:

T : 10

/ \

9 20

K = 19

Return: 1

Input 2:

T: 10

/ \

9 20

K = 40

Return: 0



**BST Iterator**

Implement an iterator over a binary search tree (BST). Your iterator will be initialized with the root node of a BST.

The first call to next() will return the smallest number in BST. Calling next() again will return the next smallest number in the BST, and so on.

**Note:** next() and hasNext() should run in average O(1) time and uses O(h) memory, where h is the height of the tree.  
Try to optimize the additional space complexity apart from the amortized time complexity.



**Recover Binary Search Tree**

Two elements of a binary search tree (BST) are swapped by mistake.  
Tell us the 2 values swapping which the tree will be restored.

**Note:**  
A solution using O(n) space is pretty straight forward. Could you devise a constant space solution?

**Example :**

Input :

1

/ \

2 3

Output :

[1, 2]

Explanation : Swapping 1 and 2 will change the BST to be

2

/ \

1 3

which is a valid BST



**Invert the Binary Tree**

Given a binary tree, invert the binary tree and return it.   
Look at the example for more details.

**Example :**   
Given binary tree

1

/ \

2 3

/ \ / \

4 5 6 7

invert and return

1

/ \

3 2

/ \ / \

7 6 5 4



**Inorder Traversal**

Given a binary tree, return the inorder traversal of its nodes’ values.

**Example :**  
Given binary tree

1

\

2

/

3

return [1,3,2].



**Postorder Traversal**

Given a binary tree, return the postorder traversal of its nodes’ values.

**Example :**

Given binary tree

1

\

2

/

3

return [3,2,1].



**Preorder Traversal**

Given a binary tree, return the preorder traversal of its nodes’ values.

**Example :**  
Given binary tree

1

\

2

/

3

return [1,2,3].



**ZigZag Level Order Traversal BT**

Given a binary tree, return the zigzag level order traversal of its nodes’ values. (ie, from left to right, then right to left for the next level and alternate between).

**Example :**   
Given binary tree

3

/ \

9 20

/ \

15 7

return

[

[3],

[20, 9],

[15, 7]

]



**Populate Next Right Pointers Tree**

Given a binary tree

struct TreeLinkNode {

TreeLinkNode \*left;

TreeLinkNode \*right;

TreeLinkNode \*next;

}

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.

Initially, all next pointers are set to NULL.

**Note:**

* You may only use constant extra space.

**Example :**

Given the following binary tree,

1

/ \

2 3

/ \ / \

4 5 6 7

After calling your function, the tree should look like:

1 -> NULL

/ \

2 -> 3 -> NULL

/ \ / \

4->5->6->7 -> NULL



**Path Sum**

Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.

**Example :**

Given the below binary tree and sum = 22,

5

/ \

4 8

/ / \

11 13 4

/ \ \

7 2 1

return true, as there exist a root-to-leaf path 5->4->11->2 which sum is 22.

**Return 0 / 1 ( 0 for false, 1 for true ) for this problem**



**Root to Leaf Paths With Sum**

Given a binary tree and a sum, find all root-to-leaf paths where each path’s sum equals the given sum.

For example:  
Given the below binary tree and sum = 22,

5

/ \

4 8

/ / \

11 13 4

/ \ / \

7 2 5 1

return

[

[5,4,11,2],

[5,8,4,5]

]



**Max Depth of Binary Tree**

Given a binary tree, find its maximum depth.

The maximum depth of a binary tree is the number of nodes along the longest path from the root node down to the farthest leaf node.

**NOTE :** The path has to end on a leaf node.

**Example :**

1

/

2

max depth = 2.



**Min Depth of Binary Tree**

Given a binary tree, find its minimum depth.

The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.

**NOTE :** The path has to end on a leaf node.

**Example :**

1

/

2

min depth = 2.



**Sum Root to Leaf Numbers**

Given a binary tree containing digits from 0-9 only, each root-to-leaf path could represent a number.

An example is the root-to-leaf path 1->2->3 which represents the number 123.

Find the total sum of all root-to-leaf numbers % 1003.

**Example :**

1

/ \

2 3

The root-to-leaf path 1->2 represents the number 12.  
The root-to-leaf path 1->3 represents the number 13.

Return the sum = (12 + 13) % 1003 = 25 % 1003 = 25.



**Shortest Unique Prefix**

Find shortest unique prefix to represent each word in the list.

**Example:**

Input: [zebra, dog, duck, dove]

Output: {z, dog, du, dov}

where we can see that

zebra = z

dog = dog

duck = du

dove = dov



**Least Common Ancestor**

Find the lowest common ancestor in an unordered binary tree given two values in the tree.

**Lowest common ancestor :** the lowest common ancestor (LCA) of two nodes v and w in a tree or directed acyclic graph (DAG) is the lowest (i.e. deepest) node that has both v and w as descendants.

**Example :**

\_\_\_\_\_\_\_3\_\_\_\_\_\_

/ \

\_\_\_5\_\_ \_\_\_1\_\_

/ \ / \

6 \_2\_ 0 8

/ \

7 4

For the above tree, the LCA of nodes 5 and 1 is 3.

**LCA** = Lowest common ancestor

Please note that LCA for nodes 5 and 4 is 5.

* You are given 2 values. Find the lowest common ancestor of the two nodes represented by val1 and val2
* No guarantee that val1 and val2 exist in the tree. If one value doesn’t exist in the tree then return -1.
* There are no duplicate values.
* You can use extra memory, helper functions, and can modify the node struct but, you can’t add a parent pointer.



**Flatten Binary Tree to Linked List**

Given a binary tree, flatten it to a linked list in-place.

**Example :**  
Given0

1

/ \

2 5

/ \ \

3 4 6

The flattened tree should look like:

1

\

2

\

3

\

4

\

5

\

6

Note that the left child of all nodes should be NULL.



**Order of People Heights**

You are given the following :

* A positive number N
* Heights : A list of heights of N persons standing in a queue
* Infronts : A list of numbers corresponding to each person (P) that gives the **number of persons** who are **taller** than P and standing in front of P

You need to return list of actual order of persons’s height

**Consider that heights will be unique**

**Example**

Input :

Heights: 5 3 2 6 1 4

InFronts: 0 1 2 0 3 2

Output :

actual order is: 5 3 2 1 6 4

So, you can see that for the person with height 5, there is no one taller than him who is in front of him, and hence Infronts has 0 for him.

For person with height 3, there is 1 person ( Height : 5 ) in front of him who is taller than him.

You can do similar inference for other people in the list.



**Morris Traversal**

**Using Morris Traversal, we can traverse the tree without using stack and recursion.**

****

Greedy Algorithms

**Highest Product**

Given an array **A**, of **N** integers A.

Return the highest product possible by multiplying 3 numbers from the array.

**NOTE:** Solution will fit in a 32-bit signed integer.

**Input Format:**

The first and the only argument is an integer array A.

**Output Format:**

Return the highest possible product.

**Constraints:**

1 <= N <= 5e5

**Example:**

Input 1:

A = [1, 2, 3, 4]

Output 1:

24

Explanation 1:

2 \* 3 \* 4 = 24

Input 2:

A = [0, -1, 3, 100, 70, 50]

Output 2:

350000

Explanation 2:

70 \* 50 \* 100 = 350000



**Bulbs**

**N** light bulbs are connected by a wire.

Each bulb has a switch associated with it, however due to faulty wiring, a switch also changes the state of all the bulbs to the right of current bulb.

Given an initial state of all bulbs, find the minimum number of switches you have to press to turn on all the bulbs.

You can press the same switch multiple times.

**Note :** 0 represents the bulb is off and 1 represents the bulb is on.

**Input Format:**

The first and the only argument contains an integer array A, of size N.

**Output Format:**

Return an integer representing the minimum number of switches required.

**Constraints:**

1 <= N <= 5e5  
0 <= A[i] <= 1

**Example:**

Input 1:

A = [1]

Output 1: 0

Explanation 1:

There is no need to turn any switches as all the bulbs are already on.

Input 2:

A = [0 1 0 1]

Output 2:

4

Explanation 2:

press switch 0 : [1 0 1 0]

press switch 1 : [1 1 0 1]

press switch 2 : [1 1 1 0]

press switch 3 : [1 1 1 1]



**Distribute Candy**

There are **N** children standing in a line. Each child is assigned a rating value.

You are giving candies to these children subjected to the following requirements:

1. Each child must have at least one candy.

2. Children with a higher rating get more candies than their neighbors.

What is the minimum candies you must give?

**Input Format:**

The first and the only argument contains N integers in an array A.

**Output Format:**

Return an integer, representing the minimum candies to be given.

**Example:**

Input 1:

A = [1, 2]

Output 1:

3

Explanation 1:

The candidate with 1 rating gets 1 candy and candidate with rating cannot get 1 candy as 1 is its neighbor.

So rating 2 candidate gets 2 candies. In total, 2 + 1 = 3 candies need to be given out.

Input 2:

A = [1, 5, 2, 1]

Output 2:

7

Explanation 2:

Candies given = [1, 3, 2, 1]



**Assign Mice to Holes**

There are N Mice and N holes are placed in a straight line.   
Each hole can accomodate only 1 mouse.   
A mouse can stay at his position, move one step right from x to x + 1, or move one step left from x to x − 1. Any of these moves consumes 1 minute.  
Assign mice to holes so that the time when the last mouse gets inside a hole is minimized.

**Example:**

positions of mice are:4 -4 2

positions of holes are:4 0 5

Assign mouse at position x=4 to hole at position x=4 : Time taken is 0 minutes

Assign mouse at position x=-4 to hole at position x=0 : Time taken is 4 minutes

Assign mouse at position x=2 to hole at position x=5 : Time taken is 3 minutes

After 4 minutes all of the mice are in the holes.

Since, there is no combination possible where the last mouse's time is less than 4,

answer = 4.

**Input:**

A : list of positions of mice

B : list of positions of holes

**Output:**

single integer value



**Majority Element**

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

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

**Example :**

Input : [2, 1, 2]

Return : 2 which occurs 2 times which is greater than 3/2.



**Seats**

There is a row of seats. Assume that it contains N seats adjacent to each other. There is a group of people who are already seated in that row randomly. i.e. some are sitting together & some are scattered.

An occupied seat is marked with a character 'x' and an unoccupied seat is marked with a dot ('.')

Now your target is to make the whole group sit together i.e. next to each other, without having any vacant seat between them in such a way that the total number of hops or jumps to move them should be minimum.

**Return minimum value % MOD where MOD = 10000003**

**Example**

Here is the row having 15 seats represented by the String (0, 1, 2, 3, ......... , 14) -

. . . . x . . x x . . . x . .

Now to make them sit together one of approaches is -

. . . . . . x x x x . . . . .

Following are the steps to achieve this -

1 - Move the person sitting at 4th index to 6th index -

Number of jumps by him = (6 - 4) = 2

2 - Bring the person sitting at 12th index to 9th index -

Number of jumps by him = (12 - 9) = 3

So now the total number of jumps made =

( 2 + 3 ) % MOD =

5 which is the minimum possible jumps to make them seat together.

There are also other ways to make them sit together but the number of jumps will exceed 5 and that will not be minimum.

For example bring them all towards the starting of the row i.e. start placing them from index 0.

In that case the total number of jumps will be

( 4 + 6 + 6 + 9 )%MOD

= 25 which is very costly and not an optimized way to do this movement



**Gas Station**

Given two integer arrays **A** and **B** of size **N**.  
There are **N** gas stations along a circular route, where the amount of gas at station **i** is **A[i]**.

You have a car with an unlimited gas tank and it costs **B[i]** of gas to travel from station **i**   
to its next station **(i+1)**. You begin the journey with an empty tank at one of the gas stations.

Return the minimum starting gas station’s index if you can travel around the circuit once, otherwise return -1.

You can only travel in one direction. **i to i+1, i+2, … n-1, 0, 1, 2..** Completing the circuit means starting at **i** and   
ending up at **i** again.

**Input Format**

The first argument given is the integer array A.

The second argument given is the integer array B.

**Output Format**

Return the minimum starting gas station's index if you can travel around the circuit once, otherwise return -1.

**For Example**

Input 1:

A = [1, 2]

B = [2, 1]

Output 1:

1

Explanation 1:

If you start from index 0, you can fill in A[0] = 1 amount of gas. Now your tank has 1 unit of gas. But you need B[0] = 2 gas to travel to station 1.

If you start from index 1, you can fill in A[1] = 2 amount of gas. Now your tank has 2 units of gas. You need B[1] = 1 gas to get to station 0. So, you travel to station 0 and still have 1 unit of gas left over. You fill in A[0] = 1 unit of additional gas, making your current gas = 2. It costs you B[0] = 2 to get to station 1, which you do and complete the circuit.



Graph

**Smallest sequence with given Primes**

Given three prime number(p1, p2, p3) and an integer k. Find the first(smallest) k integers which have only p1, p2, p3 or a combination of them as their prime factors.

**Example:**

*Input :*   
Prime numbers : [2,3,5]   
k : 5

If primes are given as p1=2, p2=3 and p3=5 and k is given as 5, then the sequence of first 5 integers will be:

*Output:*   
{2,3,4,5,6}

*Explanation :*   
4 = p1 \* p1 ( 2 \* 2 )  
6 = p1 \* p2 ( 2 \* 3 )

**Note:** The sequence should be sorted in ascending order



**Valid Path**

There is a rectangle with left bottom as (0, 0) and right up as (x, y). There are N circles such that their centers are inside the rectangle.  
Radius of each circle is R. Now we need to find out if it is possible that we can move from (0, 0) to (x, y) without touching any circle.

**Note :** We can move from any cell to any of its 8 adjecent neighbours and we cannot move outside the boundary of the rectangle at any point of time.  
  
  
**Input Format**

1st argument given is an Integer x.

2nd argument given is an Integer y.

3rd argument given is an Integer N, number of circles.

4th argument given is an Integer R, radius of each circle.

5th argument given is an Array A of size N, where A[i] = x cordinate of ith circle

6th argument given is an Array B of size N, where B[i] = y cordinate of ith circle

**Output Format**

Return YES or NO depending on weather it is possible to reach cell (x,y) or not starting from (0,0).

**Constraints**

0 <= x, y, R <= 100

1 <= N <= 1000

Center of each circle would lie within the grid

**For Example**

Input:

x = 2

y = 3

N = 1

R = 1

A = [2]

B = [3]

Output:

NO

Explanation:

There is NO valid path in this case



**Level Order**

Given a binary tree, return the level order traversal of its nodes’ values. (ie, from left to right, level by level).

**Example :**  
Given binary tree

3

/ \

9 20

/ \

15 7

return its level order traversal as:

[

[3],

[9,20],

[15,7]

]

Also think about a version of the question where you are asked to do a level order traversal of the tree when depth of the tree is much greater than number of nodes on a level.



**Smallest Multiple With 0 and 1**

You are given an integer N. You have to find smallest multiple of N which consists of digits 0 and 1 only. Since this multiple could be large, return it in form of a string.

**Note**:

* Returned string should not contain leading zeroes.

For example,

For N = 55, 110 is smallest multiple consisting of digits 0 and 1.

For N = 2, 10 is the answer.



**Black Shapes**

Given **N x M** character matrix **A** of O's and X's, where O = white, X = black.  
Return the number of black shapes. A black shape consists of one or more adjacent X's (diagonals not included)  
**Input Format:**

The First and only argument is a N x M character matrix.

**Output Format:**

Return a single integer denoting number of black shapes.

**Constraints:**

1 <= N,M <= 1000

A[i][j] = 'X' or 'O'

**Example:**

Input 1:

A = [ OOOXOOO

OOXXOXO

OXOOOXO ]

Output 1:

3

Explanation:

3 shapes are :

(i) X

X X

(ii)

X

(iii)

X

X

**Note:** we are looking for connected shapes here.

XXX

XXX

XXX

is just **one** single connected black shape.



**Capture Regions on Board**

Given a 2D board containing 'X' and 'O', capture all regions surrounded by 'X'.

A region is captured by flipping all 'O's into 'X's in that surrounded region.

**Input Format:**

First and only argument is a N x M character matrix A

**Output Format:**

make changes to the the input only as matrix is passed by reference.

**Constraints:**

1 <= N,M <= 1000

**For Example:**

Input 1:

A = [ [X, X, X, X],

[X, O, O, X],

[X, X, O, X],

[X, O, X, X] ]

Output 1:

After running your function, the board should be:

A = [ [X, X, X, X],

[X, X, X, X],

[X, X, X, X],

[X, O, X, X] ]

Explanation:

O in (4,2) is not surrounded by X from below.



**Commutable Islands**

There are **A** islands and there are **M** bridges connecting them. Each bridge has some **cost** attached to it.

We need to find bridges with **minimal cost** such that all islands are connected.

It is guaranteed that input data will contain **at least one** possible scenario in which all islands are connected with each other.

**Input Format:**

The first argument contains an integer, A, representing the number of islands.

The second argument contains an 2-d integer matrix, B, of size M x 3:

=> Island B[i][0] and B[i][1] are connected using a bridge of cost B[i][2].

**Output Format:**

Return an integer representing the minimal cost required.

**Constraints:**

1 <= A, M <= 6e4

1 <= B[i][0], B[i][1] <= A

1 <= B[i][2] <= 1e3

**Examples:**

Input 1:

A = 4

B = [ [1, 2, 1]

[2, 3, 4]

[1, 4, 3]

[4, 3, 2]

[1, 3, 10] ]

Output 1:

6

Explanation 1:

We can choose bridges (1, 2, 1), (1, 4, 3) and (4, 3, 2), where the total cost incurred will be (1 + 3 + 2) = 6.

Input 2:

A = 4

B = [ [1, 2, 1]

[2, 3, 2]

[3, 4, 4]

[1, 4, 3] ]

Output 2:

6

Explanation 2:

We can choose bridges (1, 2, 1), (2, 3, 2) and (1, 4, 3), where the total cost incurred will be (1 + 2 + 3) = 6.



**Possibility of finishing all courses given pre-requisites**

There are a total of **A** courses you have to take, labeled from **1** to **A**.

Some courses may have prerequisites, for example to take course **2** you have to first take course **1**, which is expressed as a pair: **[1,2]**.

Given the total number of courses and a list of prerequisite pairs, is it possible for you to finish all courses?

Return **1** if it is **possible** to finish all the courses, or **0** if it is **not possible** to finish all the courses.

**Input Format:**

The first argument of input contains an integer A, representing the number of courses.

The second argument of input contains an integer array, B.

The third argument of input contains an integer array, C.

**Output Format:**

Return a boolean value:

1 : If it is possible to complete all the courses.

0 : If it is not possible to complete all the courses.

**Constraints:**

1 <= A <= 6e4

1 <= length(B) = length(C) <= 1e5

1 <= B[i], C[i] <= A

**Example:**

Input 1:

A = 3

B = [1, 2]

C = [2, 3]

Output 1:

1

Explanation 1:

It is possible to complete the courses in the following order:

1 -> 2 -> 3

Input 2:

A = 2

B = [1, 2]

C = [2, 1]

Output 2:

0

Explanation 2:

It is not possible to complete all the courses.



**Cycle**

****

****

**Largest Distance between nodes of a Tree**

**Find largest distance  
Given an arbitrary unweighted rooted tree which consists of N (2 <= N <= 40000) nodes. The goal of the problem is to find largest distance between two nodes in a tree. Distance between two nodes is a number of edges on a path between the nodes (there will be a unique path between any pair of nodes since it is a tree). The nodes will be numbered 0 through N - 1.**

**The tree is given as an array P, there is an edge between nodes P[i] and i (0 <= i < N). Exactly one of the i’s will have P[i] equal to -1, it will be root node.**

**Example:  
If given P is [-1, 0, 0, 0, 3], then node 0 is the root and the whole tree looks like this:**

**0**

**/ | \**

**1 2 3**

**\**

**4**

**One of the longest path is 1 -> 0 -> 3 -> 4 and its length is 3, thus the answer is 3. Note that there are other paths with maximal distance.**

****

**Stepping Numbers**

Given N and M find all stepping numbers in range N to M

**The stepping number:**

A number is called as a stepping number if the adjacent digits have a difference of 1.

e.g 123 is stepping number, but 358 is not a stepping number

**Example:**

N = 10, M = 20

all stepping numbers are 10 , 12

Return the numbers in sorted order.



**Word Search Board**

Given a 2D board and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The cell itself does not count as an adjacent cell.   
The same letter cell may be used more than once.

**Example :**

Given board =

[

["ABCE"],

["SFCS"],

["ADEE"]

]

word = "ABCCED", -> returns 1,

word = "SEE", -> returns 1,

word = "ABCB", -> returns 1,

word = "ABFSAB" -> returns 1

word = "ABCD" -> returns 0



**Convert Sorted List to Binary Search Tree**

Given a singly linked list where elements are sorted in ascending order, convert it to a height balanced BST.

**A height balanced BST :** a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

**Example :**

Given A : 1 -> 2 -> 3

A height balanced BST :

2

/ \

1 3



**Sum Of Fibonacci Numbers**

How many minimum numbers from fibonacci series are required such that sum of numbers should be equal to a given Number N?  
**Note : repetition of number is allowed.**

**Example:**

N = 4

Fibonacci numbers : 1 1 2 3 5 .... so on

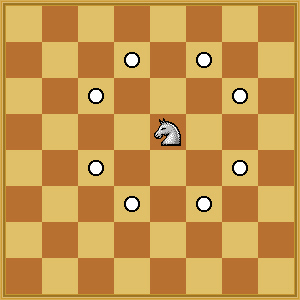
here 2 + 2 = 4

so minimum numbers will be 2



**Knight On Chess Board**

Given any source point, **(C, D)** and destination point, **(E, F)** on a chess board, we need to find whether Knight can move to the destination or not.



The above figure details the movements for a knight ( **8** possibilities ).

If yes, then what would be the **minimum** number of steps for the knight to move to the said point.  
If knight can not move from the source point to the destination point, then return **-1**.

**Note:** A knight cannot go out of the board.

**Input Format:**

The first argument of input contains an integer A.

The second argument of input contains an integer B.

=> The chessboard is of size A x B.

The third argument of input contains an integer C.

The fourth argument of input contains an integer D.

=> The Knight is initially at position (C, D).

The fifth argument of input contains an integer E.

The sixth argument of input contains an integer F.

=> The Knight wants to reach position (E, F).

**Output Format:**

If it is possible to reach the destination point, return the minimum number of moves.

Else return -1.

**Constraints:**

1 <= A, B <= 500

**Example**

Input 1:

A = 8

B = 8

C = 1

D = 1

E = 8

F = 8

Output 1:

6

Explanation 1:

The size of the chessboard is 8x8, the knight is initially at (1, 1) and the knight wants to reach position (8, 8).

The minimum number of moves required for this is 6.



**Word Ladder I**

iven two words **A** and **B**, and a dictionary, **C**, find the length of **shortest** transformation sequence from **A** to **B**, such that:

* You must change exactly **one** character in every transformation.
* Each intermediate word must exist in the dictionary.

**Note:**

1. Return **0** if there is no such transformation sequence.
2. All words have the same length.
3. All words contain only lowercase alphabetic characters.

**Input Format:**

The first argument of input contains a string, A.

The second argument of input contains a string, B.

The third argument of input contains an array of strings, C.

**Output Format:**

Return an integer representing the minimum number of steps required to change string A to string B.

**Constraints:**

1 <= length(A), length(B), length(C[i]) <= 25

1 <= length(C) <= 5e3

**Example :**

Input 1:

A = "hit"

B = "cog"

C = ["hot", "dot", "dog", "lot", "log"]

Output 1:

5

Explanation 1:

"hit" -> "hot" -> "dot" -> "dog" -> "cog"



**Word Ladder II**

Given two words (**start** and **end**), and a dictionary, find the shortest transformation sequence from **start** to **end**, such that:

* Only one letter can be changed at a time
* Each intermediate word must exist in the dictionary

If there are multiple such sequence of shortest length, return all of them. Refer to the example for more details.

**Note:**

* All words have the same length.
* All words contain only lowercase alphabetic characters.

**Input Format**

The first argument is string start.

The second argument is string end.

The third argument is an array of strings dict

**Output Format**

Return all transformation sequences such that first word of each sequence is start and last word is end, all intermediate words belongs to dictionary(dict) and consecutive words had atmost 1 difference.

**Example :**

:

start = "hit"

end = "cog"

dict = ["hot","dot","dog","lot","log"]

Return

[

["hit","hot","dot","dog","cog"],

["hit","hot","lot","log","cog"]

]



**Clone Graph**

Clone an undirected graph. Each node in the graph contains a label and a list of its neighbors.

