Accenture Interview Coding Round Questions

1. Problem Statement: Two Sum

The problem statement is: We are given a list of N elements and a number M. We have to find two elements in the given list whose sum is M.
List = [a1, a2, ... , aN]
So, if the two elements are ai and aj, then:
ai + aj = M

Example:
Given numbers = [2, 7, 11, 15], target = 9,

Because nums[0] + nums[1] = 2 + 7 = 9,

return [0, 1]

2. Problem Statement: Largest Contiguous Sum

Kadane's Algorithm: Given an integer array arr, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum and print the subarray.
Example:

Example 1:

Input: arr = [-2,1,-3,4,-1,2,1,-5,4]

Output: 6

Explanation: [4,-1,2,1] has the largest sum = 6.

Examples 2:

Input: arr = [1]

Output: 1

Explanation: Array has only one element and which is giving positive sum of 1.

3. Problem Statement: Next Greater Permutation

Next Greater Permutation: Given an array Arr[] of integers, rearrange the numbers of the given array into the lexicographically next greater permutation of numbers.
If such an arrangement is not possible, it must rearrange it as the lowest possible order (i.e., sorted in ascending order).
Example:

Array: [1, 2, 3, 4]

Next Greater Permutation: [1, 2, 4, 3]

Next Greater Permutation: [1, 3, 2, 4]

Next Greater Permutation: [1, 3, 4, 2]

Next Greater Permutation: [1, 4, 2, 3]

Next Greater Permutation: [1, 4, 3, 2]

Next Greater Permutation: [2, 1, 3, 4]

Array: [4, 3, 2, 1]

Next Greater Permutation: [1, 2, 3, 4]

4. Problem Statement: Matrix Rotation

Matrix rotation: Given a matrix, rotate the matrix 90 degrees clockwise.

Matrix:
1 2 3
4 5 6
7 8 9
After rotation:
7 4 1
8 5 2
9 6 3

5. Problem Statement: Merge Overlapping Intervals

The problem statement is: Given a list of intervals, merge them to get a list of non-overlapping intervals.

interval = [start, end]

Example:

Intervals: [[1, 2], [2, 3], [1, 4], [5, 6]]

[1, 2] and [2, 3] can be merged to form [1, 3].

Now, [1, 3] and [1, 4] can be merged to form [1, 4].

[1, 4] and [5, 6] have no intersection.

Hence above intervals are merged to form: [[1, 4], [5, 6]]

6. Problem Statement: Max Consecutive Ones

The problem statement is: Given an array A, find the maximum number of consecutive 1s in the array.

Example:
A: [1, 1, 3, 2, 3, 1, 1, 1]

Max consecutive 1s: 3

Approach

1. Initialize 2 variables to 0 (one for counting and other for maximum count)

2. By using a loop (loop through the nums list) check whether the number is 1 if it is 1 then

3. Increment a counter by 1 (note: counter is initially 0)

4. Check if the maximum count is greater than the counter, if yes then assign the counter value to maximum count if it is not 1 then assign 0 to counter

5. After looping through the list return the maximum count

7. Problem Statement: Pascal's Triangle

In this triangle, the value at a position is equal to the sum of values of the 2 elements just above it.

Examples

The 2nd element of 5th row is 1+3 => 4

The 3rd element of 5th row is 3+3 => 6

The 4th element of 5th row is 3+1 => 4

Given a number n, find the first n row of pascal's triangle.

8. Problem Statement: Kth Largest Element

Kth largest element in an array : Given an array and a number k where k is smaller than the size of the array, we need to find the k'th largest element in the given array. It is given that all array elements are distinct.

arr[] = [3,2,1,5,6,4]
k = 2

output: 5

9. Problem Statement: Search Rotated Sorted Array

Search for a given number in a sorted array, with unique elements, that has been rotated by some arbitrary number. Return -1 if the number does not exist.

arr[] = [5, 6, 7, 8, 9, 10, 1, 2, 3]
target = 3

10. Problem Statement: Trapping Rain Water

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.

Example: arr = [0,1,0,2,1,0,1,3,2,1,2,1]

Output: 6

11. Problem Statement: Intersection of Two Linked Lists

Given the heads of two singly linked-lists headA and headB, return the node at which the two lists intersect. If the two linked lists have no intersection at all, return null.
Example
listA = [4,1,8,4,5],

listB = [5,6,1,8,4,5]

Output: Intersected at '8'

12. Problem Statement: Median of Row-wise Sorted Matrix

We are given a row-wise sorted matrix of size r*c, we need to find the median of the matrix given. It is assumed that r*c is always odd.

Example:
1 3 5
2 6 9
3 6 9
Output: Median is 5

13. Problem Statement: Clone List with Random Pointer

A linked list of length n is given such that each node contains an additional random pointer, which could point to any node in the list, or null.

Construct a deep copy of the list. The deep copy should consist of exactly n brand new nodes, where each new node has its value set to the value of its corresponding original node. Both the next and random pointer of the new nodes should point to new nodes in the copied list such that the pointers in the original list and copied list represent the same list state.

Example:

head = [[7,null],[13,0],[11,4],[10,2],[1,0]]

Output: [[7,null],[13,0],[11,4],[10,2],[1,0]]

14. Problem Statement: Merge Two Sorted Linked List

You are given the heads of two sorted linked lists list1 and list2. Merge the two lists in a one sorted list. The list should be made by splicing together the nodes of the first two lists. Return the head of the merged linked list.

Example: list1 = [1,2,4], list2 = [1,3,4]

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

15. Problem Statement: Reversing a Linked List in K-groups

Reversing a Linked List Given a linked list and a positive number k, reverse the nodes in groups of k. All the remaining nodes after multiples of k should be left as it is.

Example:

k: 3

Linked list: 1→2→3→4→5→6→7→8→9

Result: 3→2→1→6→5→4→9→8→7

16. Problem Statement: Non-Repeating Element

Given a sorted list of numbers in which all elements appear twice except one element that appears only once, find the number that appears only once.
Example:

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

Output: 2

17. Problem Statement: Implement Min Stack

Implement a stack that supports the following operations in O(1) time complexity:

push(x) : Push the element x onto the stack.

pop() : Removes the element at the top of the stack.

top() : Get the topmost element of the stack.

getMin() : Get the minimum element in the stack.

Assume that pop, top and getMin will only be called on non-empty stack.

18. Problem Statement: Balanced Parentheses

Balanced parentheses : Given an expression string exp, write a program to examine whether the pairs and the orders of “{“, “}”, “(“, “)”, “[“, “]” are correct in exp.

Example:

Input: exp = "[()]{}{[()()]()}"

Output: Balanced

Input: exp = "[(])"

Output: Not Balanced

19. Problem Statement: Next Greater Element

Given an array, print the Next Greater Element (NGE) for every element. The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

Example:

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

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

20. Problem Statement: Sliding Window Maximum

Sliding Window Maximum: Given an array and an integer K in sliding window maximum problem, find the maximum for each and every contiguous subarray of size k.

Example:

arr[] = [1, 2, 3, 1, 4, 5, 2, 3, 6]

K = 3

Output: 3 3 4 5 5 5 6

Explanation

Maximum of 1, 2, 3 is 3

Maximum of 2, 3, 1 is 3

Maximum of 3, 1, 4 is 4

Maximum of 1, 4, 5 is 5

Maximum of 4, 5, 2 is 5

Maximum of 5, 2, 3 is 5

Maximum of 2, 3, 6 is 6

21. Problem Statement: Longest Substring Without Repeat

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

Example: s: "logicmojo"

Output: 6 i.e (logicm)

22. Problem Statement: Maximum Path Sum of Binary Tree

Given a binary tree, find the maximum path sum. The path may start and end at any node in the tree.

Output: 42
Explanation: The optimal path is 15 -> 20 -> 7 with a path sum of 15 + 20 + 7 = 42.

Approach 1: Brute Force

Traverse left and right subtree of each node and calculate maximum possible sum path. We can update the max path sum passing through each node n in the tree by traversing the n's left subtree and right subtree.

Algorithm:

1. Assume we have nodes numbered 1 to N
2. sum(i) = Maximum sum of a path containing node(i). Clearly the solution of the problem is max(sum(1), sum(2), ..., sum(N))
3. Now, what is the maximum sum of a path containing a particular node(i)?
4. left_result: maximum path sum starting at node(i).left; right_result: maximum path sum starting at node(i).right; sum(i) = max(left_result, 0) + max(right_result, 0) + node(i).val

Time Complexity would be O(n2) and space complexity would be O(n)

Approach 2: Using Recursive Approach

The above approach is also recursive in nature but not the optimal one so to approve the solution we need to think on some important points in above solution, i.e. how can we try to compute the max sum for any node if you have the max sum for their left and right sub-tree?

Each node can be the root of the final maximum path sum. The root here means the topmost node in a path. We calculate the maximum Path Sum rooted at each node and update the max sum during the traversal.

There can only be four different cases when a particular node is involved in the max path.

1. If its the only Node
2. Max path through Left Child + Node
3. Max path through right Child + Node
4. Max path through Left Child + Node + Right Child

23. Problem Statement: Lowest Common Ancestor in Binary Tree

The lowest common ancestor of two nodes p and q is the lowest node in the binary tree that has p and q as its descendants.

Note: A node is also considered a descendant of itself.
Given the reference to the root node and two nodes p and q in a binary tree, return the reference to the lowest common ancestor of p and q.

Note: You can assume that p and q will be present in the tree.

24. Problem Statement: Bottom View of Binary Tree

There are different ways to look at a binary tree. The bottom view of a binary tree contains the set of nodes that will be visible if you look at the binary tree from the bottom.

Note: If there are multiple bottom-most nodes for a horizontal distance from root, use the later one in level-order traversal.

Given the root node of a binary tree, return an array containing the node elements in the bottom view, from left to right.

Approach Recursive
The bottom view of a binary tree refers to the bottommost nodes present at their horizontal distance.
For the nodes of a binary tree, the horizontal distance is defined as follows.
Horizontal distance of the root = 0

Horizontal distance of a left child = horizontal distance of its parent - 1
Horizontal distance of a right child = horizontal distance of its parent + 1

25. Problem Statement: Rat In A Maze

You are given a maze in the form of a matrix of size n * m. Each cell is either clear or blocked denoted by 1 and 0 respectively. A rat sits at the top-left cell and there exists a block of cheese at the bottom-right cell. Both these cells are guaranteed to be clear. You need to find if the rat can get the cheese if it can move only in one of the two directions - down and right. It can't move to blocked cells.

26. Problem Statement: Combination Sum

Given an array of distinct integers candidates and a target integer target, return a list of all unique combinations of candidates where the chosen numbers sum to target. You may return the combinations in any order.
Example:

arr = [2,3,5], target = 8

Output: [[2,2,2,2],[2,3,3],[3,5]]

27. Problem Statement: Rotting Apples

You are given an n * m grid where each position can contain one of the three values:
Every day, all fresh apples which are adjacent to any rotten apple become rotten. Two cells are adjacent if they have a common edge, i.e., each cell can have upto four adjacent cells.
Find the minimum number of days required for all the apples to be rotten. If this is not possible return -1.

Example: matrix = [[2,1,1],[1,1,0],[0,1,1]]

Output: 4

28. Problem Statement: Diameter of Binary Tree

Given a binary tree, return the length of the diameter of the tree. The diameter of a binary tree is the length of the longest path between any two nodes of the tree. The length is the number of edges in the path. The path may or may not include the root node.

29. Problem Statement: Construct Binary Tree from Preorder and Inorder Traversal

Given two integer arrays preorder and inorder where preorder is the preorder traversal of a binary tree and inorder is the inorder traversal of the same tree, construct and return the binary tree.

30. Problem Statement: Longest Increasing Subsequence (LIS)

Longest Increasing Subsequence:
Given an array A, find the length of the longest strictly increasing subsequence (LIS). A subsequence is a sequence that can be derived from an array by deleting some or no elements such that the order of the remaining elements remain the same.

Examples: A: [10, 20, 2, 5, 3, 8, 8, 25, 6]

Result: 4

Explanation: Longest increasing subsequence: [2, 5, 8, 25]

LISArray[0] is unused

Maximum index of LISArray can be at (Size of arr).

31. Problem Statement: Word Search Board

Given an m x n grid of characters board and a string word, return true if word exists in the grid. The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once.
Examples:

board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]],

word = "ABCCED"

Output: True

32. Problem Statement: Number of Islands

You are given a 2-D matrix surface of size n*m. Each cell of the surface is either 1 (land) or 0 (water). Find the number of islands on the surface. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. Assume all four edges of the surface are all surrounded by water.

Example: surface: [[1, 1, 0, 1],[1, 0, 1, 1],[0, 1, 0, 1]]

Output: 3

33. Problem Statement: Knight's Journey on a Chessboard

You have a chessboard of size n*n. A knight sits on the board at a position start(x, y). The knight wants to go to another cell end(x, y). Find the minimum number of moves required to go from the start position to the end position. If this is not possible, return -1.

Example: N = 30, knightpos = [1, 1], targetpos = [30, 30]

Output: 20

34. Problem Statement: Coin Change

Coin Change Problem: You are given coins of different denominations, represented by an array - coins of size n. You are also given a value - target. Return the fewest number of coins that you need to make up that amount. If that amount of

money cannot be made up by any combination of the coins, return -1. You may assume that you have an infinite number of each kind of coin.

Example: coins: [1,2,5], amount: 11

Output: Result: 3

Explanation: The Minimum coins needed are 5 + 5 + 1

35. Problem Statement: LRU Cache

LRU Cache : Implement Least Recently Used (LRU) cache. You need to implement the following for the LRUCache class:

LRUCache(int capacity) initializes the cache to store data of size: capacity.

int get(int key) returns the value of the key if it exists, otherwise returns -1.

void add(int key, int value) updates the value of the key if the key exists.

Otherwise, add the key-value pair to the cache. If the number of keys exceeds the

capacity from this operation, evict the least recently used key.

Note: Try to achieve each operation in O(1) time complexity.

36. Problem Statement: Find xth Node from End of Linked List

Given a linked list, find the xth node from the end.

Example: Linked list: 1→2→3→4; x: 2

Result: 1→2→4

37. Problem Statement: Pythagorean Triplet in an array

Given an array of integers, write a function that returns true if there is a triplet (a, b, c) that satisfies a2 + b2 = c2.

Example: arr = [3, 1, 4, 6, 5]
Output: True, There is a Pythagorean triplet (3, 4, 5).

Approach 1: Brute Force

A simple solution is to run three loops, three loops pick three array elements, and check if the current three elements form a Pythagorean Triplet. Time complexity will be O(n3) i.e. not the optimal one so we need to optimise it.

Efficient: Using Hashing

The problem can also be solved using hashing. We can use a hash map to mark all the values of the given array. Using two loops, we can iterate for all the possible combinations of a and b, and then check if there exists the third value c. If there exists any such value, then there is a Pythagorean triplet.

38. Problem Statement: Check if BST is valid or not

Check BST: Given the root of a binary tree, determine if it is a valid binary search tree (BST).

A valid BST is defined as follows:

The left subtree of a node contains only nodes with keys less than the node's key.
The right subtree of a node contains only nodes with keys greater than the node's key.
Both the left and right subtrees must also be binary search trees

Example: root = [2,1,3]
Output: True

Approach 1: Brute Force

We need to check is the left subtree of a node contains any element greater than the node's value and whether the right subtree of a node contains any element smaller than the node's value.

We shall design two helper functions getMin(root) and getMax(root) which will get the minimum and maximum values of the tree with a given root. Then we will check :
If the node's value is greater than the maximum value in the left subtree or not
The node's value is lesser than the minimum value in the right subtree or not.
Basically, we will check if this expression holds true or not: getMax(root.left) < root.val < getMin(root.right)
Time complexity will be O(n2) i.e. not the optimal one so we need to optimise it.

Approach 2: Using Inorder Traversal
We know that the inorder traversal of a binary search tree gives a sorted order of its elements. We shall use this to our advantage and traverse the tree in inorder fashion and store the elements in an array. We shall then traverse the array in a linear manner and check if it's in sorted order or not.

39. There are n stairs, a person standing at the bottom wants to reach the top. The person can climb either 1 stair or 2 stairs at a time. Count the number of ways, the person can reach the top.

Consider the example shown in the diagram. The value of n is 3. There are 3 ways to reach the top. The diagram is taken from Fibonacci puzzle
Examples:
Input: n = 1
Output: 1

There is only one way to climb 1 stair

Input: n = 2

Output: 2

There are two ways: (1, 1) and (2)

Input: n = 4

Output: 5

(1, 1, 1, 1), (1, 1, 2), (2, 1, 1), (1, 2, 1), (2, 2)

Method 1: The first method uses the technique of recursion to solve this problem.

Approach: We can easily find the recursive nature in the above problem. The person can reach nth stair from either (n-1)th stair or from (n-2)th stair. Hence, for each stair n, we try to find out the number of ways to reach n-1th stair and n-2th stair and add them to give the answer for the nth stair. Therefore the expression for such an approach comes out to be :

ways(n) = ways(n-1) + ways(n-2)

The above expression is actually the expression for Fibonacci, but there is one thing to notice, the value of ways(n) is equal to fibonacci(n+1).

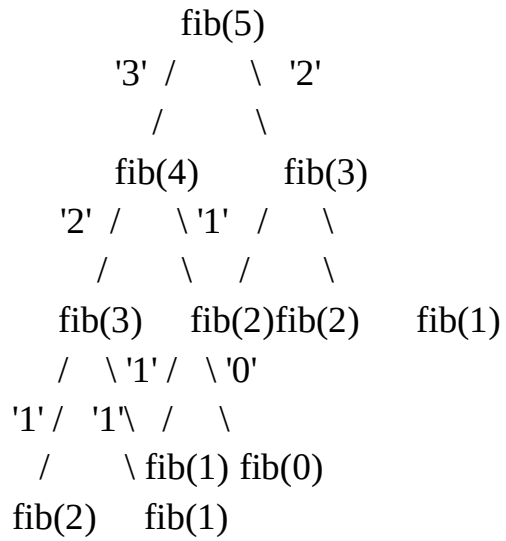
ways(1) = fib(2) = 1

ways(2) = fib(3) = 2

ways(3) = fib(4) = 3

For a better understanding, let's refer to the recursion tree below -:

Input: N = 4



Output:

Number of ways = 5

40. Given an array arr[] of N distinct integers, output the array in such a way that the first element is first maximum and the second element is the first minimum, and so on.

Example 1:
Input: N = 7, arr[] = {7, 1, 2, 3, 4, 5, 6}

Output: 7 1 6 2 5 3 4

Explanation: The first element is first

maximum and second element is first

minimum and so on.

Example 2:

Input: N = 8, arr[] = {1, 6, 9, 4, 3, 7, 8, 2}

Output: 9 1 8 2 7 3 6 4

Your Task:
This is a function problem. You don't need to take any input, as it is already accomplished by the driver code. You just need to complete the function alternateSort() that takes array arr and integer N as parameters and returns the desired array as output.

Expected Time Complexity: O(NlogN).
Expected Auxiliary Space: O(N).

Constraints:

$1 \leq N \leq 106$

41. Given an m x n 2D grid map of '1's which represents land and '0's that represents water, return the number of islands (surrounded by water and formed by connecting adjacent lands in 2 directions - vertically or horizontally). Assume that the boundary cases - which is all four edges of the grid are surrounded by water.
Constraints are:

m == grid.length
n == grid[i].length
1 <= m, n <= 300
grid[i][j] can only be '0' or '1'.

Example:
Input: grid = [
[“1” , “1” , “1” , “0” , “0”],
[“1” , “1” , “0” , “0” , “0”],
[“0” , “0” , “1” , “0” , “1”],
[“0” , “0” , “0” , “1” , “1”]
]

42. Given an array of strings strs, group the anagrams together. You can return the answer in any order.
An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Example 1:
Input: strs = ["eat","tea","tan","ate","nat","bat"]
Output: [["bat"],["nat","tan"],["ate","eat","tea"]]
Example 2:
Input: strs = [""]
Output: [[""]]
Example 3:
Input: strs = ["a"]
Output: [["a"]]

Constraints:
1 <= strs.length <= 104
0 <= strs[i].length <= 100
strs[i] consists of lowercase English letters.

43. Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).

Implement the MyQueue class:
void push(int x) Pushes element x to the back of the queue.
int pop() Removes the element from the front of the queue and returns it.
int peek() Returns the element at the front of the queue.
boolean empty() Returns true if the queue is empty, false otherwise.

Notes:
You must use only standard operations of a stack, which means only push to top, peek/pop from top, size, and is empty operations are valid.
Depending on your language, the stack may not be supported natively. You may simulate a stack using a list or deque (double-ended queue) as long as you use only a stack's standard operations.

Example 1:
Input
["MyQueue", "push", "push", "peek", "pop", "empty"]
[[], [1], [2], [], [], []]
Output
[null, null, null, 1, 1, false]

Explanation
MyQueue myQueue = new MyQueue();
myQueue.push(1); // queue is: [1]
myQueue.push(2); // queue is: [1, 2] (leftmost is front of the queue)
myQueue.peek(); // return 1
myQueue.pop(); // return 1, queue is [2]
myQueue.empty(); // return false

Constraints:
1 <= x <= 9
At most 100 calls will be made to push, pop, peek, and empty.
All the calls to pop and peek are valid.

Follow-up: Can you implement the queue such that each operation is amortized O(1) time complexity? In other words, performing n operations will take overall O(n) time even if one of those operations may take longer.

44. Given a string s, return the longest palindromic substring in s.

Example 1:
Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
Example 2:
Input: s = "cbbd"
Output: "bb"

Constraints:
1 <= s.length <= 1000
s consist of only digits and English letters.

45. Given the roots of two binary trees root and subRoot, return true if there is a subtree of root with the same structure and node values of subRoot and false otherwise.
A subtree of a binary tree tree is a tree that consists of a node in tree and all of this node's descendants. The tree tree could also be considered as a subtree of itself.

Example 1:

Input: root = [3,4,5,1,2], subRoot = [4,1,2]

Output: true

Example 2:

Input: root = [3,4,5,1,2,null,null,null,null,0], subRoot = [4,1,2]

Output: false

Constraints:

The number of nodes in the root tree is in the range [1, 2000].

The number of nodes in the subRoot tree is in the range [1, 1000].

-104 <= root.val <= 104

-104 <= subRoot.val <= 104

46. Given a string s and a dictionary of strings wordDict, return true if s can be segmented into a space-separated sequence of one or more dictionary words.
Note that the same word in the dictionary may be reused multiple times in the segmentation.

Example 1:

Input: s = "leetcode", wordDict = ["leet","code"]
Output: true

Explanation: Return true because "leetcode" can be segmented as "leet code".
Example 2:
Input: s = "applepenapple", wordDict = ["apple","pen"]
Output: true
Explanation: Return true because "applepenapple" can be segmented as "apple pen apple".
Note that you are allowed to reuse a dictionary word.
Example 3:
Input: s = "catsandog", wordDict = ["cats","dog","sand","and","cat"]
Output: false

Constraints:
1 <= s.length <= 300
1 <= wordDict.length <= 1000
1 <= wordDict[i].length <= 20
s and wordDict[i] consist of only lowercase English letters.
All the strings of wordDict are unique.

47. You are given an integer array coins representing coins of different denominations and an integer amount representing a total amount of money.
Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1.

You may assume that you have an infinite number of each kind of coin.

Example 1:
Input: coins = [1,2,5], amount = 11
Output: 3
Explanation: 11 = 5 + 5 + 1
Example 2:
Input: coins = [2], amount = 3
Output: -1
Example 3:
Input: coins = [1], amount = 0
Output: 0

Constraints:
1 <= coins.length <= 12
1 <= coins[i] <= 231 - 1
0 <= amount <= 104

48. Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array nums = [0,1,2,4,5,6,7] might become:
[4,5,6,7,0,1,2] if it was rotated 4 times.
[0,1,2,4,5,6,7] if it was rotated 7 times.
Notice that rotating an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].
Given the sorted rotated array nums of unique elements, return the minimum element of this array.
You must write an algorithm that runs in O(log n) time.

Example 1:
Input: nums = [3,4,5,1,2]
Output: 1
Explanation: The original array was [1,2,3,4,5] rotated 3 times.
Example 2:
Input: nums = [4,5,6,7,0,1,2]
Output: 0
Explanation: The original array was [0,1,2,4,5,6,7] and it was rotated 4 times.
Example 3:
Input: nums = [11,13,15,17]
Output: 11

Explanation: The original array was [11,13,15,17] and it was rotated 4 times.

Constraints:
n == nums.length
1 <= n <= 5000
-5000 <= nums[i] <= 5000
All the integers of nums are unique.
nums is sorted and rotated between 1 and n times.

49. You are given an array of strings products and a string searchWord.
Design a system that suggests at most three product names from products after each character of searchWord is typed. Suggested products should have common prefix with searchWord. If there are more than three products with a common prefix return the three lexicographically minimums products.
Return a list of lists of the suggested products after each character of searchWord is typed.

Example 1:

Input: products = ["mobile","mouse","moneypot","monitor","mousepad"],

searchWord = "mouse"

Output: [

["mobile","moneypot","monitor"],

["mobile","moneypot","monitor"],

["mouse","mousepad"],

["mouse","mousepad"],

["mouse","mousepad"]

]

Explanation: products sorted lexicographically =

["mobile","moneypot","monitor","mouse","mousepad"]

After typing m and mo all products match and we show user

["mobile","moneypot","monitor"]

After typing mou, mous and mouse the system suggests ["mouse","mousepad"]

Example 2:

Input: products = ["havana"], searchWord = "havana"

Output: [["havana"],["havana"],["havana"],["havana"],["havana"],["havana"]]

Example 3:

Input: products = ["bags","baggage","banner","box","cloths"], searchWord = "bags"

Output:

[["baggage","bags","banner"],["baggage","bags","banner"],["baggage","bags"],["bags"]]

Constraints:
1 <= products.length <= 1000
1 <= products[i].length <= 3000
1 <= sum(products[i].length) <= 2 * 104
All the strings of products are unique.
products[i] consists of lowercase English letters.
1 <= searchWord.length <= 1000
searchWord consists of lowercase English letters.

50. Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

Example 1:

Input: n = 3

Output: ["((()))","(()())","(())()","()(())","()()()"]

Example 2:

Input: n = 1

Output: ["()"]

Constraints:
1 <= n <= 8

51. There is a robot on an m x n grid. The robot is initially located at the top-left corner (i.e., grid[0][0]). The robot tries to move to the bottom-right corner (i.e., grid[m - 1][n - 1]). The robot can only move either down or right at any point in time.

Given the two integers m and n, return the number of possible unique paths that the robot can take to reach the bottom-right corner.
The test cases are generated so that the answer will be less than or equal to 2 * 109.

Example 1:

Input: m = 3, n = 7

Output: 28

Example 2:

Input: m = 3, n = 2

Output: 3

Explanation: From the top-left corner, there are a total of 3 ways to reach the bottom-right corner:

1. Right -> Down -> Down

2. Down -> Down -> Right

3. Down -> Right -> Down

Constraints:
1 <= m, n <= 100

52. Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane and an integer k, return the k closest points to the origin (0, 0).

The distance between two points on the X-Y plane is the Euclidean distance (i.e., $\sqrt{}$ (x1 - x2)2 + (y1 - y2)2).
You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is in).

Example 1:

Input: points = [[1,3],[-2,2]], k = 1
Output: [[-2,2]]

Explanation:

The distance between (1, 3) and the origin is sqrt(10).
The distance between (-2, 2) and the origin is sqrt(8).
Since sqrt(8) < sqrt(10), (-2, 2) is closer to the origin.
We only want the closest k = 1 points from the origin, so the answer is just [[-2,2]].
Example 2:

Input: points = [[3,3],[5,-1],[-2,4]], k = 2

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

Explanation: The answer [[-2,4],[3,3]] would also be accepted.

Constraints:
1 <= k <= points.length <= 104
-104 < xi, yi < 104

53. Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.

Example 1:

Input: intervals = [[1,3],[2,6],[8,10],[15,18]]

Output: [[1,6],[8,10],[15,18]]

Explanation: Since intervals [1,3] and [2,6] overlaps, merge them into [1,6].

Example 2:

Input: intervals = [[1,4],[4,5]]

Output: [[1,5]]

Explanation: Intervals [1,4] and [4,5] are considered overlapping.

Constraints:
1 <= intervals.length <= 104
intervals[i].length == 2
0 <= starti <= endi <= 104

54. Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return the number of islands.
An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:

Input: grid = [

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

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

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

["0","0","0","0","0"]

]

Output: 1

Example 2:

Input: grid = [

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

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

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

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

]

Output: 3

Constraints:
m == grid.length
n == grid[i].length
1 <= m, n <= 300
grid[i][j] is '0' or '1'.

55. A linked list of length n is given such that each node contains an additional random pointer, which could point to any node in the list, or null.
Construct a deep copy of the list. The deep copy should consist of exactly n brand new nodes, where each new node has its value set to the value of its corresponding

original node. Both the next and random pointer of the new nodes should point to new nodes in the copied list such that the pointers in the original list and copied list represent the same list state. None of the pointers in the new list should point to nodes in the original list.

For example,
If there are two nodes X and Y in the original list, where X.random --> Y, then for the corresponding two nodes x and y in the copied list, x.random --> y.
Return the head of the copied linked list.
The linked list is represented in the input/output as a list of n nodes. Each node is represented as a pair of [val, random_index] where:
val: an integer representing Node.val
random_index: the index of the node (range from 0 to n-1) that the random pointer points to, or null if it does not point to any node.
Your code will only be given the head of the original linked list.

Example 1:

Input: head = [[7,null],[13,0],[11,4],[10,2],[1,0]]

Output: [[7,null],[13,0],[11,4],[10,2],[1,0]]

Example 2:

Input: head = [[1,1],[2,1]]

Output: [[1,1],[2,1]]

Example 3:

Input: head = [[3,null],[3,0],[3,null]]

Output: [[3,null],[3,0],[3,null]]

Constraints:
0 <= n <= 1000
-104 <= Node.val <= 104
Node.random is null or is pointing to some node in the linked list.

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

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, 2 is written as II in Roman numeral, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

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

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

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

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

Given a roman numeral, convert it to an integer.

Example 1:

Input: s = "III"

Output: 3

Explanation: III = 3.

Example 2:

Input: s = "LVIII"

Output: 58

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

Example 3:

Input: s = "MCMXCIV"

Output: 1994

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

Constraints:
1 <= s.length <= 15
s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').
It is guaranteed that s is a valid roman numeral in the range [1, 3999].

57. Design a class to find the kth largest element in a stream. Note that it is the kth largest element in the sorted order, not the kth distinct element.
Implement KthLargest class:
KthLargest(int k, int[] nums) Initializes the object with the integer k and the stream of integers nums.
int add(int val) Appends the integer val to the stream and returns the element representing the kth largest element in the stream.

Example 1:
Input

["KthLargest", "add", "add", "add", "add", "add"]
[[3, [4, 5, 8, 2]], [3], [5], [10], [9], [4]]
Output
[null, 4, 5, 5, 8, 8]

Explanation

KthLargest kthLargest = new KthLargest(3, [4, 5, 8, 2]);

kthLargest.add(3); // return 4

kthLargest.add(5); // return 5

kthLargest.add(10); // return 5

kthLargest.add(9); // return 8

kthLargest.add(4); // return 8

Constraints:
1 <= k <= 104
0 <= nums.length <= 104
-104 <= nums[i] <= 104
-104 <= val <= 104

At most 104 calls will be made to add.
It is guaranteed that there will be at least k elements in the array when you search for the kth element.

58. You are given an array of logs. Each log is a space-delimited string of words, where the first word is the identifier.
There are two types of logs:
Letter-logs: All words (except the identifier) consist of lowercase English letters.
Digit-logs: All words (except the identifier) consist of digits.
Reorder these logs so that:
The letter-logs come before all digit-logs.
The letter-logs are sorted lexicographically by their contents. If their contents are the same, then sort them lexicographically by their identifiers.
The digit-logs maintain their relative ordering.
Return the final order of the logs.

Example 1:

Input: logs = ["dig1 8 1 5 1","let1 art can","dig2 3 6","let2 own kit dig","let3 art zero"]
Output: ["let1 art can","let3 art zero","let2 own kit dig","dig1 8 1 5 1","dig2 3 6"]

Explanation:
The letter-log contents are all different, so their ordering is "art can", "art zero", "own kit dig".
The digit-logs have a relative order of "dig1 8 1 5 1", "dig2 3 6".

Example 2:

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

Constraints:
1 <= logs.length <= 100
3 <= logs[i].length <= 100
All the tokens of logs[i] are separated by a single space.
logs[i] is guaranteed to have an identifier and at least one word after the identifier.

59. Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.
Implement the MinStack class:
MinStack() initializes the stack object.
void push(int val) pushes the element val onto the stack.
void pop() removes the element on the top of the stack.
int top() gets the top element of the stack.
int getMin() retrieves the minimum element in the stack.

Example 1:
Input
["MinStack","push","push","push","getMin","pop","top","getMin"]
[[],[-2],[0],[-3],[],[],[],[]]

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

Explanation
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();   // return 0
minStack.getMin(); // return -2

Constraints:
-231 <= val <= 231 - 1
Methods pop, top and getMin operations will always be called on non-empty stacks.
At most 3 * 104 calls will be made to push, pop, top, and getMin.

60. You are given the heads of two sorted linked lists list1 and list2.
Merge the two lists in a one sorted list. The list should be made by splicing together the nodes of the first two lists.
Return the head of the merged linked list.

Example 1:

Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]

Example 2:
Input: list1 = [], list2 = []
Output: []

Example 3:
Input: list1 = [], list2 = [0]
Output: [0]

Constraints:
The number of nodes in both lists is in the range [0, 50].
-100 <= Node.val <= 100
Both list1 and list2 are sorted in non-decreasing order.

61. Given a positive integer n, generate an n x n matrix filled with elements from 1 to n2 in spiral order.

Example 1:

Input: n = 3
Output: [[1,2,3],[8,9,4],[7,6,5]]

Example 2:
Input: n = 1
Output: [[1]]

Constraints:
1 <= n <= 20

62. 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.

Example Input
A = [1, 2] B = [2, 1]

Example Output
1

Example Explanation
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.

63. 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.

64. 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]

65. Find the longest increasing subsequence of a given array of integers, A.
In other words, find a subsequence of array in which the subsequence's elements are in strictly increasing order, and in which the subsequence is as long as possible.
This subsequence is not necessarily contiguous, or unique.
In this case, we only care about the length of the longest increasing subsequence.

Input Format:
The first and the only argument is an integer array A.
Output Format:
Return an integer representing the length of the longest increasing subsequence.
Constraints:
1 <= length(A) <= 2000
0 <= A[i] <= 2500

Example :
Input 1:
A = [1, 2, 1, 5]

Output 1:
3

Explanation 1:
The sequence : [1, 2, 5]

Input 2:
A = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]

Output 2:
6

Explanation 2:
The sequence : [0, 2, 6, 9, 13, 15] or [0, 4, 6, 9, 11, 15] or [0, 4, 6, 9, 13, 15]

66. Find the contiguous subarray within an array (containing at least one number) which has the largest product.
Return an integer corresponding to the maximum product possible.
Example :
Input : [2, 3, -2, 4]
Return : 6

Possible with [2, 3]

67. A message containing letters from A-Z is being encoded to numbers using the following mapping:
'A' -> 1
'B' -> 2
...
'Z' -> 26
Given an encoded message A containing digits, determine the total number of ways to decode it modulo 109 + 7.

Problem Constraints
1 <= |A| <= 105

Input Format
The first and the only argument is a string A.

Output Format
Return a single integer denoting the total number of ways to decode it modulo 109 + 7.

Example Input
Input 1:

A = "8"
Input 2:
A = "12"

Example Output
Output 1:
1
Output 2:
2

Example Explanation
Explanation 1:
Given encoded message "8", it could be decoded as only "H" (8).
The number of ways decoding "8" is 1.
Explanation 2:
Given encoded message "12", it could be decoded as "AB" (1, 2) or "L" (12).
The number of ways decoding "12" is 2.

68. Say you have an array, A, for which the ith element is the price of a given stock on day i.
Design an algorithm to find the maximum profit.
You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times).
However, you may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

Input Format: The first and the only argument is an array of integer, A.

Output Format: Return an integer, representing the maximum possible profit.

Constraints: 0 <= len(A) <= 1e5 1 <= A[i] <= 1e7 Example:
Input 1:
A = [1, 2, 3]

Output 1:
2

Explanation 1:
=> Buy a stock on day 0.
=> Sell the stock on day 1. (Profit +1)
=> Buy a stock on day 1.
=> Sell the stock on day 2. (Profit +1)

Overall profit = 2

Input 2:
A = [5, 2, 10]

Output 2:
8

Explanation 2:
=> Buy a stock on day 1.
=> Sell the stock on on day 2. (Profit +8)

Overall profit = 8

69. Say you have an array, A, for which the ith element is the price of a given stock on day i.
Design an algorithm to find the maximum profit. You may complete at most 2 transactions.
Return the maximum possible profit.
Note: You may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

Input Format:
The first and the only argument is an integer array, A.
Output Format:
Return an integer, representing the maximum possible profit.
Constraints:
1 <= length(A) <= 7e5
1 <= A[i] <= 1e7
Examples:
Input 1:
A = [1, 2, 1, 2]

Output 1:
2

Explanation 1:
Day 0 : Buy
Day 1 : Sell
Day 2 : Buy
Day 3 : Sell

Input 2:
A = [7, 2, 4, 8, 7]

Output 2:
6

Explanation 2:
Day 1 : Buy
Day 3 : Sell

70. Say you have an array, A, for which the ith element is the price of a given stock on day i.
If you were only permitted to complete at most one transaction (i.e, buy one and sell one share of the stock), design an algorithm to find the maximum profit.
Return the maximum possible profit.

Problem Constraints
0 <= len(A) <= 7e5

1 <= A[i] <= 1e7

Input Format
The first and the only argument is an array of integers, A.

Output Format
Return an integer, representing the maximum possible profit.

Example Input
Input 1:
A = [1, 2]
Input 2:
A = [1, 4, 5, 2, 4]

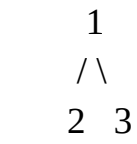
Example Output
Output 1:
1
Output 2:
4

Example Explanation
Explanation 1:
Buy the stock on day 0, and sell it on day 1.
Explanation 2:
Buy the stock on day 0, and sell it on day 2

71. Given a binary tree T, find the maximum path sum.
The path may start and end at any node in the tree.
Input Format:
The first and the only argument contains a pointer to the root of T, A.
Output Format:
Return an integer representing the maximum sum path.
Constraints:
1 <= Number of Nodes <= 7e4

-1000 <= Value of Node in T <= 1000

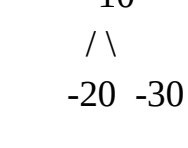
Example :
Input 1:



Output 1:
6

Explanation 1:
The path with maximum sum is: 2 -> 1 -> 3

Input 2:



Output 2:
-10

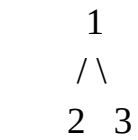
Explanation 2
The path with maximum sum is: -10

72. Given a binary tree T, find the maximum path sum.The path may start and end at any node in the tree.

Input Format:
The first and the only argument contains a pointer to the root of T, A.

Output Format:
Return an integer representing the maximum sum path.
Constraints:
1 <= Number of Nodes <= 7e4
-1000 <= Value of Node in T <= 1000

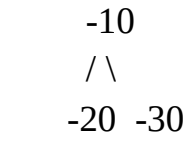
Example:
Input 1:



Output 1:
6

Explanation 1:
The path with maximum sum is: 2 -> 1 -> 3

Input 2:



Output 2:
-10

Explanation 2
The path with maximum sum is: -10

73. Given a string A, partition A such that every substring of the partition is a palindrome.
Return the minimum cuts needed for a palindrome partitioning of A.

Input Format:
The first and the only argument contains the string A.

Output Format:
Return an integer, representing the answer as described in the problem statement.
Constraints:
1 <= length(A) <= 501
Examples:
Input 1:

A = "aba"

Output 1:
0

Explanation 1:
"aba" is already a palindrome, so no cuts are needed.

Input 2:
A = "aab"

Output 2:
1

Explanation 2:
Return 1 since the palindrome partitioning ["aa","b"] could be produced using 1 cut.

74. Given a 2D integer array A of size M x N, you need to find a path from top left to bottom right which minimizes the sum of all numbers along its path.
NOTE: You can only move either down or right at any point in time.

Input Format
First and only argument is a 2D integer array A of size M x N.

Output Format
Return a single integer denoting the minimum sum of a path from cell (1, 1) to cell (M, N).

Example Input
Input 1:
A = [[1, 3, 2]
[4, 3, 1]
[5, 6, 1]
]

Example Output
Output 1:
9

Example Explanation
Explanation 1:
The path is 1 -> 3 -> 2 -> 1 -> 1
So (1 + 3 + 2 + 1 + 1) = 8

75. Given an array of non-negative integers, A, of length N, you are initially positioned at the first index of the array.
Each element in the array represents your maximum jump length at that position.
Return the minimum number of jumps required to reach the last index.
If it is not possible to reach the last index, return -1.
Input Format:
The first and the only argument contains an integer array, A.
Output Format:
Return an integer, representing the answer as described in the problem statement.
Constraints:
1 <= N <= 1e6
0 <= A[i] <= 50000
Examples:
Input 1:
A = [2, 1, 1]

Output 1:
1

Explanation 1:
The shortest way to reach index 2 is
Index 0 -> Index 2
that requires only 1 jump.

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

Output 2:
2

Explanation 2:
The shortest way to reach index 4 is
Index 0 -> Index 1 -> Index 4
that requires 2 jumps.

76. Given two strings A and B, find the minimum number of steps required to convert A to B. (each operation is counted as 1 step.)
You have the following 3 operations permitted on a word:
Insert a character
Delete a character
Replace a character

Input Format:
The first argument of input contains a string, A.
The second argument of input contains a string, B.
Output Format:
Return an integer, representing the minimum number of steps required.
Constraints:
1 <= length(A), length(B) <= 450
Examples:
Input 1:
A = "abad"
B = "abac"

Output 1:
1

Explanation 1:
Operation 1: Replace d with c.

Input 2:
A = "Anshuman"
B = "Antihuman"

Output 2:
2

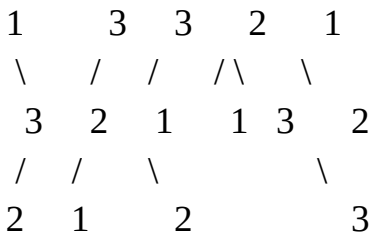
Explanation 2:
=> Operation 1: Replace s with t.
=> Operation 2: Insert i.

77. Given an integer A, how many structurally unique BST's (binary search trees) exist that can store values 1…A?

Input Format:
The first and the only argument of input contains the integer, A.
Output Format:
Return an integer, representing the answer asked in problem statement.
Constraints:
1 <= A <= 18
Example:
Input 1:
A = 3

Output 1:
5

Explanation 1:



78. Given an array of non-negative integers, A, you are initially positioned at the 0th index of the array.
Each element in the array represents your maximum jump length at that position.
Determine if you are able to reach the last index.

Input Format:
The first and the only argument of input will be an integer array A.

Output Format:

Return an integer, representing the answer as described in the problem statement.
=> 0 : If you cannot reach the last index.
=> 1 : If you can reach the last index.

Constraints:
1 <= len(A) <= 106

0 <= A[i] <= 30

Examples:
Input 1:
A = [2,3,1,1,4]

Output 1:
1

Explanation 1:
Index 0 -> Index 2 -> Index 3 -> Index 4

Input 2:
A = [3,2,1,0,4]

Output 2:
0

Explanation 2:
There is no possible path to reach the last index.

79. You are climbing a stair case and it takes A steps to reach to the top.
Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Input Format:
The first and the only argument contains an integer A, the number of steps.
Output Format:
Return an integer, representing the number of ways to reach the top.

Constrains:
1 <= A <= 36

Example:
Input 1:
A = 2 Output 1:

2 Explanation 1:

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

A = 3 Output 2:

3 Explanation 2:

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

80. 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.

81. 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}.