



AlgoTutor

# AlgoTutor's SDE INTERVIEW DSA SHEET



+91-7260058093



info@algotutor.io



www.algotutor.io

# ARRAYS

## Q 1. Binary Search

Given an array of integers `nums` which is sorted in ascending order, and an integer `target`, write a function to search `target` in `nums`. If `target` exists, then return its index. Otherwise, return -1.

You must write an algorithm with  $O(\log n)$  runtime complexity.

**Example 1:**

**Input:** `nums = [-1,0,3,5,9,12]`, **target** = 9

**Output:** 4

[Practice Now](#)

## Q 2. Find the Duplicate Number

Given an array of integers `nums` containing  $n + 1$  integers where each integer is in the range  $[1, n]$  inclusive.

There is only one repeated number in `nums`, return this repeated number.

You must solve the problem without modifying the array `nums` and uses only constant extra space.

**Example 1:**

**Input:** `nums = [1,3,4,2,2]`

**Output:** 2

[Practice Now](#)

## Q 3. Search Insert Position

Given a sorted array of distinct integers 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 must write an algorithm with  $O(\log n)$  runtime complexity.

**Example 1:**

**Input:** `nums = [1,3,4,2,2]`

**Output:** 2

[Practice Now](#)

# ARRAYS

## Q 4. Sort Colors

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

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

You must solve this problem without using the library's sort function.

**Example 1:**

**Input:** `nums = [2,0,2,1,1,0]`

**Output:** `[0,0,1,1,2,2]`

[Practice Now](#)

## Q 5. Find First and Last Position of Element in Sorted Array

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return `[-1, -1]`.

You must write an algorithm with  $O(\log n)$  runtime complexity.

**Example 1:**

**Input:** `nums = [5,7,7,8,8,10]`, **target** = 8

**Output:** `[3,4]`

[Practice Now](#)

## Q 6. Length of Last Word

Given a string `s` consisting of words and spaces, return the length of the last word in the string.

A word is a maximal substring consisting of non-space characters only.

**Example 1:**

**Input:** `s = "Hello World"`

**Output:** 5

[Practice Now](#)

# ARRAYS


## Q 7. Set Matrix Zeroes

Given an  $m \times n$  integer matrix `matrix`, if an element is 0, set its entire row and column to 0's.

You must do it in place.

**Example 1:**

1	1	1
1	0	1
1	1	1



1	0	1
0	0	0
1	0	1

**Input:** `matrix = [[1,1,1],[1,0,1],[1,1,1]]`

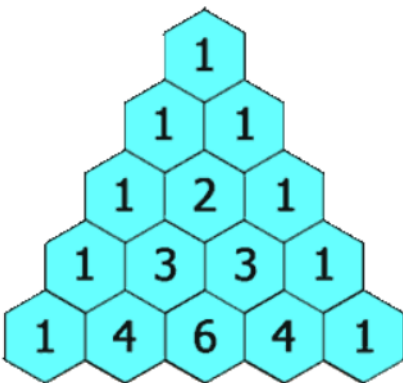
**Output:** `[[1,0,1],[0,0,0],[1,0,1]]`

[Practice Now](#)

## Q 8. Pascal's Triangle

Given an integer `numRows`, return the first `numRows` of Pascal's triangle.

In Pascal's triangle, each number is the sum of the two numbers directly above it as shown:



**Example 1:**

**Input:** `numRows = 5`

**Output:** `[[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]`

[Practice Now](#)

# ARRAYS

## Q 9. Single Element in a Sorted Array

You are given a sorted array consisting of only integers where every element appears exactly twice, except for one element which appears exactly once.

Return the single element that appears only once.

Your solution must run in  $O(\log n)$  time and  $O(1)$  space.

**Example 1:**

**Input:** `nums = [1,1,2,3,3,4,4,8,8]`

**Output:** 2

[Practice Now](#)

## Q 10. Search a 2D Matrix

You are given an  $m \times n$  integer matrix matrix with the following two properties:

- Each row is sorted in non-decreasing order.
- The first integer of each row is greater than the last integer of the previous row.

Given an integer target, return true if target is in matrix or false otherwise.

You must write a solution in  $O(\log(m * n))$  time complexity.

**Example 1:**

1	3	5	7
10	11	16	20
23	30	34	60

**Input:** `matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]]`, `target = 3`

**Output:** true

[Practice Now](#)

# ARRAYS

## Q 11. Pow(x, n)

Implement  $\text{pow}(x, n)$ , which calculates  $x$  raised to the power  $n$  (i.e.,  $x^n$ ).

**Example 1:**

**Input:**  $x = 2.00000$ ,  $n = 10$

**Output:** 1024.00000

[Practice Now](#)

## Q 12. Find a Peak Element II

A peak element in a 2D grid is an element that is strictly greater than all of its adjacent neighbors to the left, right, top, and bottom.

Given a 0-indexed  $m \times n$  matrix  $\text{mat}$  where no two adjacent cells are equal, find any peak element  $\text{mat}[i][j]$  and return the length 2 array  $[i, j]$ .

You may assume that the entire matrix is surrounded by an outer perimeter with the value -1 in each cell.

You must write an algorithm that runs in  $O(m \log(n))$  or  $O(n \log(m))$  time.

**Example 1:**

-1	-1	-1	-1
-1	1	4	-1
-1	3	2	-1
-1	-1	-1	-1

**Input:**  $\text{mat} = [[1,4],[3,2]]$

**Output:** [0,1]

[Practice Now](#)



# ARRAYS

## Q 13. Max Value of Equation

You are given an array points containing the coordinates of points on a 2D plane, sorted by the x-values, where  $\text{points}[i] = [x_i, y_i]$  such that  $x_i < x_j$  for all  $1 \leq i < j \leq \text{points.length}$ . You are also given an integer  $k$ .

Return the maximum value of the equation  $y_i + y_j + |x_i - x_j|$  where  $|x_i - x_j| \leq k$  and  $1 \leq i < j \leq \text{points.length}$ .

It is guaranteed that there exists at least one pair of points that satisfy the constraint  $|x_i - x_j| \leq k$ .

**Example 1:**

**Input:**  $x = [[1,3],[2,0],[5,10],[6,-10]]$ ,  $k = 1$ ,  $n = 1$

**Output:** 4

[Practice Now](#)

## Q 14. Jump Game

You are given an integer array nums. You are initially positioned at the array's first index, and each element in the array represents your maximum jump length at that position.

Return true if you can reach the last index, or false otherwise.

**Example 1:**

-1	-1	-1	-1
-1	1	4	-1
-1	3	2	-1
-1	-1	-1	-1

**Input:**  $\text{nums} = [2,3,1,1,4]$

**Output:** true

[Practice Now](#)

# STRING

## Q 1. Remove Outermost Parentheses

A valid parentheses string is either empty `""`, `"(" + A + ")"`, or `A + B`, where `A` and `B` are valid parentheses strings, and `+` represents string concatenation.

For example, `""`, `"()"`, `"(())"`, and `"(()())"` are all valid parentheses strings.

A valid parentheses string `s` is primitive if it is nonempty, and there does not exist a way to split it into `s = A + B`, with `A` and `B` nonempty valid parentheses strings.

Given a valid parentheses string `s`, consider its primitive decomposition: `s = P1 + P2 + ... + Pk`, where `Pi` are primitive valid parentheses strings.

Return `s` after removing the outermost parentheses of every primitive string in the primitive decomposition of `s`.

**Example 1:**

**Input:** `s = "(()())()"`

**Output:** `"()()"`

[Practice Now](#)

## Q 2. Reverse Words in a String

Given an input string `s`, reverse the order of the words.

A word is defined as a sequence of non-space characters. The words in `s` will be separated by at least one space.

Return a string of the words in reverse order concatenated by a single space.

Note that `s` may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

**Example 1:**

**Input:** `s = "the sky is blue"`

**Output:** `"blue is sky the"`

[Practice Now](#)



# STRING

## Q 3. Integer to Roman

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

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

For example, 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 an integer, convert it to a roman numeral.

**Example 1:**

**Input:** num = 3

**Output:** "|||"

[Practice Now](#)



**AlgoTutor**

# Why Choose **AlgoTutor**?



100% Placement Assistance



200+ Successful Alumni



100% Success Rate



Learn from scratch



1-1 personal mentorship  
from Industry experts



147(Avg.)% Salary Hike





23 LPA (Avg.) CTC



Career Services

Follow Us:    

 +91-7260058093  
 info@algotutor.io  
 www.algotutor.io

# STRING

## Q 4. Simplify Path

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

Given a string path, which is an absolute path (starting with a slash '/') to a file or directory in a Unix-style file system, convert it to the simplified canonical path.

In a Unix-style file system, a period '.' refers to the current directory, a double period '..' refers to the directory up a level, and any multiple consecutive slashes (i.e. '//') are treated as a single slash '/'. For this problem, any other format of periods such as '...' are treated as file/directory names.

- The canonical path should have the following format:
- The path starts with a single slash '/'.
- Any two directories are separated by a single slash '/'.
- The path does not end with a trailing '/'.
- The path only contains the directories on the path from the root directory to the target file or directory (i.e., no period '.' or double period '..')

Return the simplified canonical path.

**Example 1:**

**Input:** path = "/home/"

**Output:** "/home"

[Practice Now](#)

## Q 5. Longest Palindromic Substring

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

**Example 1:**

**Input:** s = "babad"

**Output:** "bab"

[Practice Now](#)

# STRING

## Q 6. Isomorphic Strings

Given two strings *s* and *t*, determine if they are isomorphic.

Two strings *s* and *t* are isomorphic if the characters in *s* can be replaced to get *t*.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

**Example 1:**

**Input:** *s* = "egg", *t* = "add"

**Output:** true

[Practice Now](#)

## Q 7. Longest Common Prefix

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

**Example 1:**

**Input:** *strs* = ["flower","flow","flight"]

**Output:** "fl"

[Practice Now](#)

## Q 8. Valid Palindrome II

Given a string *s*, return true if the *s* can be palindrome after deleting at most one character from it.

**Example 1:**

**Input:** *s* = "aba"

**Output:** true

[Practice Now](#)

# STRING

## Q 9. Find the Index of the First Occurrence in a String

Given two strings needle and haystack, return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

**Example 1:**

**Input:** haystack = "sadbutsad", needle = "sad"

**Output:** 0

[Practice Now](#)

## Q 10. Basic Calculator II

Given a string s which represents an expression, evaluate this expression and return its value.

The integer division should truncate toward zero.

You may assume that the given expression is always valid. All intermediate results will be in the range of  $[-2^{31}, 2^{31} - 1]$ .

Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().

**Example 1:**

**Input:** s = "3+2\*2"

**Output:** 7

[Practice Now](#)

## Explore Our Popular Courses



**Data Structure  
& Algorithms**

**Advance System  
Design  
(LLD + HLD)**



**Advanced Data  
Science &  
Machine Learning**

**MERN Full Stack  
Development**



# STRING

## Q 11. Largest Odd Number in String

You are given a string `num`, representing a large integer. Return the largest-valued odd integer (as a string) that is a non-empty substring of `num`, or an empty string `""` if no odd integer exists.

A substring is a contiguous sequence of characters within a string.

**Example 1:**

**Input:** `num = "52"`

**Output:** `"5"`

[Practice Now](#)

## Q 12. Count and Say

The count-and-say sequence is a sequence of digit strings defined by the recursive formula:

`countAndSay(1) = "1"`

- `countAndSay(n)` is the way you would "say" the digit string from
- `countAndSay(n-1)`, which is then converted into a different digit string.

To determine how you "say" a digit string, split it into the minimal number of substrings such that each substring contains exactly one unique digit. Then for each substring, say the number of digits, then say the digit. Finally, concatenate every said digit.

For example, the saying and conversion for digit string `"3322251"`:

**Example 1:**

`"3322251"`  
two 3's, three 2's, one 5, and one 1  
`2 3 + 3 2 + 1 5 + 1 1`  
`"23321511"`

Given a positive integer `n`, return the `n`th term of the count-and-say sequence.

**Input:** `n = 1`

**Output:** `"1"`

[Practice Now](#)



# STRING

## Q 13. Minimum Window Substring

Given two strings  $s$  and  $t$  of lengths  $m$  and  $n$  respectively, return the minimum window substring of  $s$  such that every character in  $t$  (including duplicates) is included in the window. If there is no such substring, return the empty string `""`.

The testcases will be generated such that the answer is unique.

**Example 1:**

**Input:**  $s = \text{"ADOBECODEBANC"}, t = \text{"ABC"}$

**Output:** `"BANC"`

[Practice Now](#)

## Q 14. Valid Anagram

Given two strings  $s$  and  $t$ , return true if  $t$  is an anagram of  $s$ , and false otherwise.

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:**  $s = \text{"anagram"}, t = \text{"nagaram"}$

**Output:** true

[Practice Now](#)

## Why AlgoTutor?

200+ Student Placed At:



- ✓ More than **136% hike growth** for 5 out of 8 working professionals.
- ✓ Average Package of **CTC 22 Lakh**.
- ✓ Learn from top industry experts.

# LINKED LIST

## Q 1. Delete Node in a Linked List

There is a singly-linked list head and we want to delete a node node in it.

You are given the node to be deleted node. You will not be given access to the first node of head.

All the values of the linked list are unique, and it is guaranteed that the given node node is not the last node in the linked list.

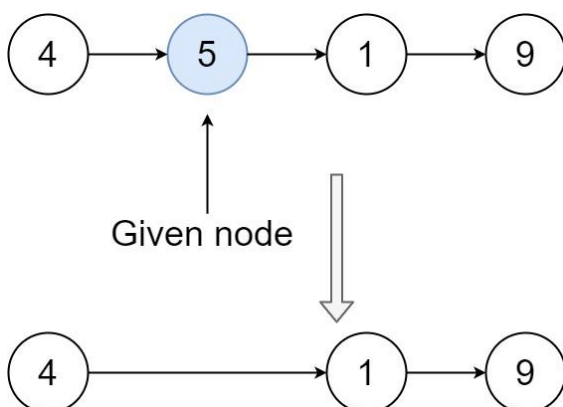
Delete the given node. Note that by deleting the node, we do not mean removing it from memory. We mean:

- The value of the given node should not exist in the linked list.
- The number of nodes in the linked list should decrease by one.
- All the values before node should be in the same order.
- All the values after node should be in the same order.

Custom testing:

- For the input, you should provide the entire linked list head and the node to be given node. node should not be the last node of the list and should be an actual node in the list.
- We will build the linked list and pass the node to your function.
- The output will be the entire list after calling your function.

**Example 1:**



**Input:** head = [4,5,1,9], node = 5

**Output:** [4,1,9]

[Practice Now](#)

# LINKED LIST

## Q 2. Middle of the Linked List

Given the head of a singly linked list, return the middle node of the linked list.

If there are two middle nodes, return the second middle node.

**Example 1:**



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

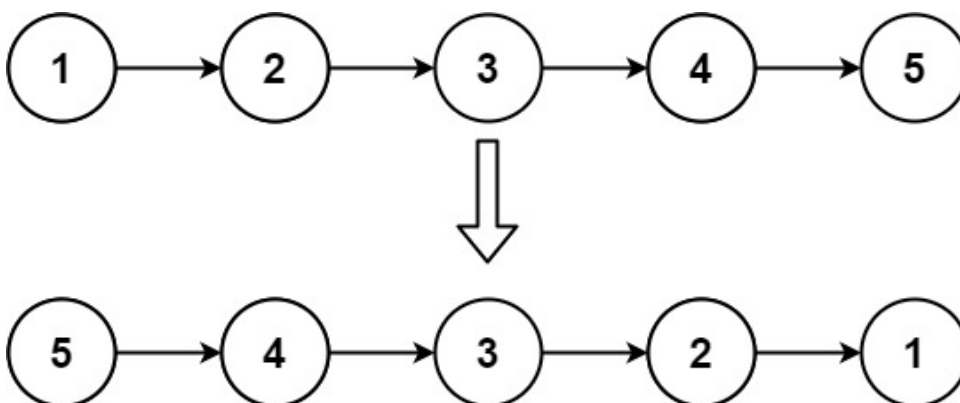
**Output:** [3,4,5]

[Practice Now](#)

## Q 3. Reverse Linked List

Given the head of a singly linked list, reverse the list, and return the reversed list.

**Example 1:**



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

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

[Practice Now](#)

# LINKED LIST

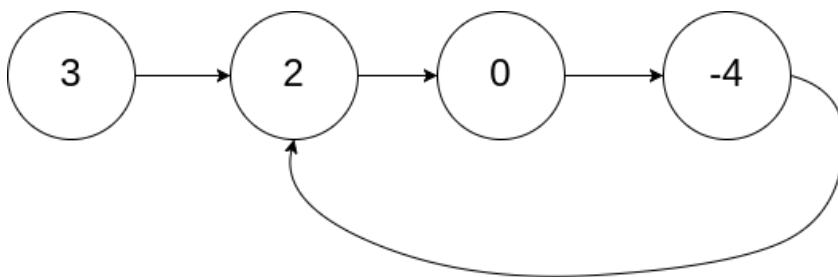
## Q 4. Linked List Cycle

Given head, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

**Example 1:**



**Input:** head = [3,2,0,-4], pos = 1

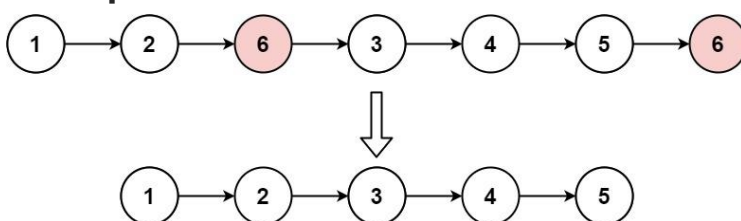
**Output:** true

[Practice Now](#)

## Q 5. Remove Linked List Elements

remove all the nodes of the linked list that has Node.val == val, and return the new head.

**Example 1:**



**Input:** head = [1,2,6,3,4,5,6], val = 6

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

[Practice Now](#)

# LINKED LIST

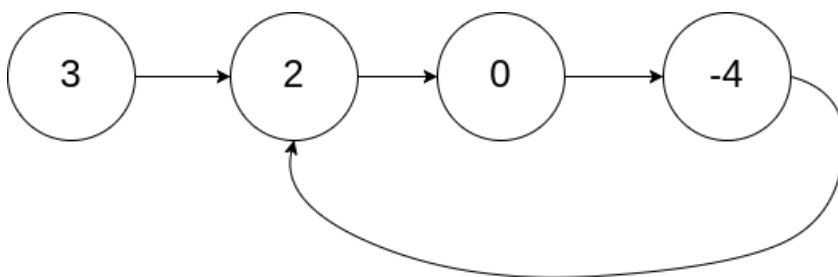
## Q 6. Linked List Cycle II

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

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to (0-indexed). It is -1 if there is no cycle. Note that pos is not passed as a parameter.

Do not modify the linked list.

**Example 1:**



**Input:** head = [3,2,0,-4], pos = 1

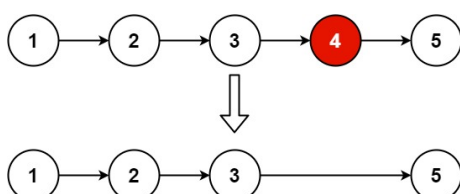
**Output:** tail connects to node index 1

[Practice Now](#)

## Q 7. Remove Nth Node From End of List

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

**Example 1:**



**Input:** head = [1,2,3,4,5], n = 2

**Output:** [1,2,3,5]

[Practice Now](#)

# LINKED LIST

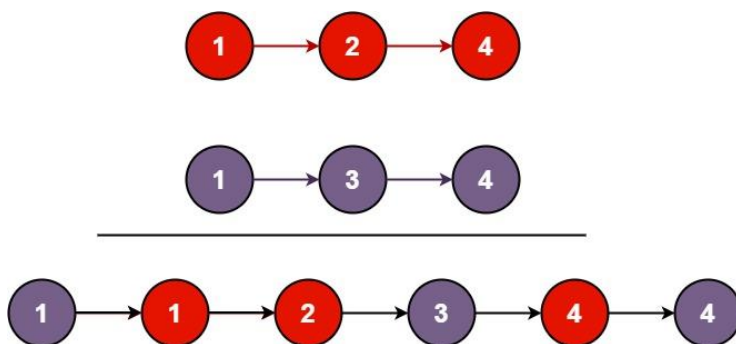
## Q 8. Merge Two Sorted Lists

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]

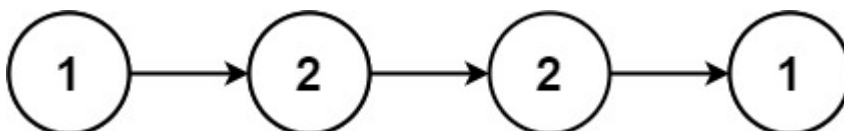
**Output:** [1,1,2,3,4,4]

[Practice Now](#)

## Q 9. Palindrome Linked List

Given the head of a singly linked list, return true if it is a Palindrome or false otherwise.

**Example 1:**



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

**Output:** true

[Practice Now](#)



# LINKED LIST

## Q 10. Odd Even Linked List

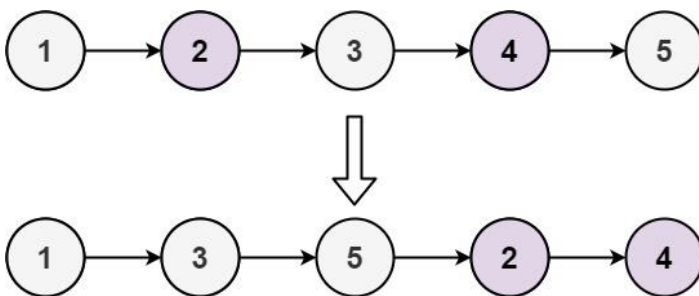
Given the head of a singly linked list, group all the nodes with odd indices together followed by the nodes with even indices, and return the reordered list.

The first node is considered odd, and the second node is even, and so on.

Note that the relative order inside both the even and odd groups should remain as it was in the input.

You must solve the problem in  $O(1)$  extra space complexity and  $O(n)$  time complexity.

**Example 1:**



**Input:** list1 = [1,2,3,4,5]

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

[Practice Now](#)

**Meet Our Alumni**

**Preeti Duhan**

J.P.Morgan

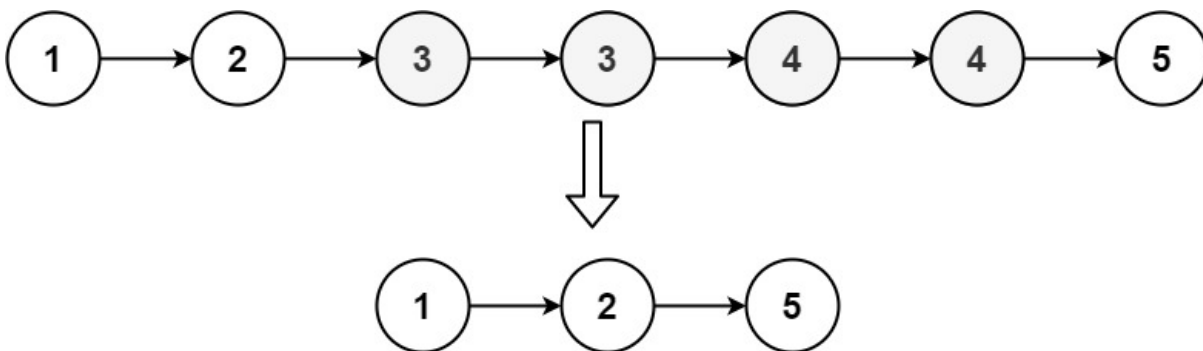


# LINKED LIST

## Q 11. Remove Duplicates from Sorted List II

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

**Example 1:**



**Input:** list1 = [1,2,3,4,5]

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

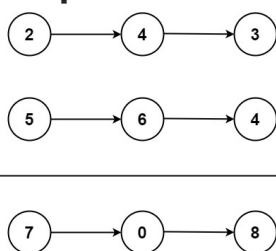
[Practice Now](#)

## Q 12. Add Two Numbers

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example 1:**



**Input:** l1 = [2,4,3], l2 = [5,6,4]

**Output:** [7,0,8]

[Practice Now](#)

# STACKS & QUEUES

## Q 1. Implement Queue using Stacks

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**

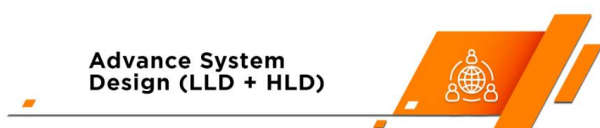
```
["MyQueue", "push", "push", "peek", "pop", "empty"]  
[[], [1], [2], [], [], []]
```

**Output**

```
[null, null, null, 1, 1, false]
```

[Practice Now](#)

## Explore Our Popular Courses



# STACKS & QUEUES

## Q 2. Implement Stack using Queues

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).

Implement the MyStack class:

- void push(int x) Pushes element x to the top of the stack.
- int pop() Removes the element on the top of the stack and returns it.
- int top() Returns the element on the top of the stack.
- boolean empty() Returns true if the stack is empty, false otherwise.

**Example 1:**

**Input**

```
["MyStack", "push", "push", "top", "pop", "empty"]  
[[], [1], [2], [], [], []]
```

**Output**

```
[null, null, null, 2, 2, false]
```

[Practice Now](#)

## Q 3. Backspace String Compare

Given two strings s and t, return true if they are equal when both are typed into empty text editors. '#' means a backspace character.

Note that after backspacing an empty text, the text will continue empty.

**Example 1:**

**Input:** s = "ab#c", t = "ad#c"

**Output:** true

[Practice Now](#)

# STACKS & QUEUES

## Q 4. Valid Parentheses

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

An input string is valid if:

Open brackets must be closed by the same type of brackets.

Open brackets must be closed in the correct order.

Every close bracket has a corresponding open bracket of the same type.

**Example 1:**

**Input:** `s = "()"`

**Output:** `true`

[Practice Now](#)

## Q 5. Min Stack

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.

You must implement a solution with  $O(1)$  time complexity for each function.

**Example 1:**

**Input**

```
["MinStack","push","push","push","getMin","pop","top","getMin"]  
[[],[-2],[0],[-3],[],[],[],[[
```

**Output**

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

[Practice Now](#)

# STACKS & QUEUES

## Q 6. Next Greater Element |

The next greater element of some element  $x$  in an array is the first greater element that is to the right of  $x$  in the same array.

You are given two distinct 0-indexed integer arrays `nums1` and `nums2`, where `nums1` is a subset of `nums2`.

For each  $0 \leq i < \text{nums1.length}$ , find the index  $j$  such that `nums1[i] == nums2[j]` and determine the next greater element of `nums2[j]` in `nums2`. If there is no next greater element, then the answer for this query is `-1`.

Return an array `ans` of length `nums1.length` such that `ans[i]` is the next greater element as described above.

**Example 1:**

**Input:** `nums1 = [4,1,2]`, `nums2 = [1,3,4,2]`

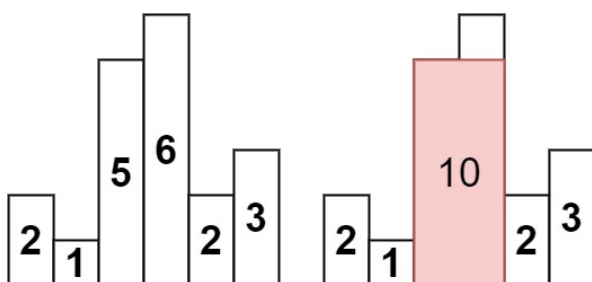
**Output:** `[-1,3,-1]`

[Practice Now](#)

## Q 7. Largest Rectangle in Histogram

Given an array of integers `heights` representing the histogram's bar height where the width of each bar is 1, return the area of the largest rectangle in the histogram.

**Example 1:**



**Input:** `heights = [2,1,5,6,2,3]`

**Output:** 10

[Practice Now](#)



# STACKS & QUEUES

## Q 8. Online Stock Span

Design an algorithm that collects daily price quotes for some stock and returns the span of that stock's price for the current day.

The span of the stock's price in one day is the maximum number of consecutive days (starting from that day and going backward) for which the stock price was less than or equal to the price of that day.

- For example, if the prices of the stock in the last four days is [7,2,1,2] and the price of the stock today is 2, then the span of today is 4 because starting from today, the price of the stock was less than or equal 2 for 4 consecutive days.
- Also, if the prices of the stock in the last four days is [7,34,1,2] and the price of the stock today is 8, then the span of today is 3 because starting from today, the price of the stock was less than or equal 8 for 3 consecutive days.

Implement the StockSpanner class:

- StockSpanner() Initializes the object of the class.
- int next(int price) Returns the span of the stock's price given that today's price is price.

### Example 1:

#### Input

```
["StockSpanner", "next", "next", "next", "next", "next", "next", "next"]
```

```
[[ ], [100], [80], [60], [70], [60], [75], [85]]
```

#### Output

```
[null, 1, 1, 1, 2, 1, 4, 6]
```

[Practice Now](#)

# STACKS & QUEUES

## Q 9. Minimum Insertions to Balance a Parentheses String

Given a parentheses string  $s$  containing only the characters '(' and ')'. A parentheses string is balanced if:

Any left parenthesis '(' must have a corresponding two consecutive right parenthesis '))'.

Left parenthesis '(' must go before the corresponding two consecutive right parenthesis '))'.

In other words, we treat '(' as an opening parenthesis and '))' as a closing parenthesis.

**For example**, "()", "()((()))" and "(()())" are balanced, ")()", "())" and "(()))" are not balanced.

You can insert the characters '(' and ')' at any position of the string to balance it if needed.

Return the minimum number of insertions needed to make  $s$  balanced.

**Example 1:**

**Input  $s$**  = "()())"

**Output** = 1

[Practice Now](#)

## Q 10. 132 Pattern

Given an array of  $n$  integers  $nums$ , a 132 pattern is a subsequence of three integers  $nums[i]$ ,  $nums[j]$  and  $nums[k]$  such that  $i < j < k$  and  $nums[i] < nums[k] < nums[j]$ .

Return true if there is a 132 pattern in  $nums$ , otherwise, return false.

**Example 1:**

**Input:  $nums$**  = [1,2,3,4]

**Output:** false

[Practice Now](#)

# STACKS & QUEUES

## Q 11. 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 can trap after raining.

**Example 1:**



**Input:** height = [0,1,0,2,1,0,1,3,2,1,2,1]

**Output** = 6

[Practice Now](#)

## Q 12. Merge Intervals

Given an array of intervals where  $\text{intervals}[i] = [\text{start}_i, \text{end}_i]$ , 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]]

[Practice Now](#)

# STACKS & QUEUES

## Q 13. Sum of Subarray Minimums

Given an array of integers `arr`, find the sum of  $\min(b)$ , where  $b$  ranges over every (contiguous) subarray of `arr`. Since the answer may be large, return the answer modulo  $10^9 + 7$ .

**Example 1:**

**Input:** `arr = [3,1,2,4]`

**Output** = 17

[Practice Now](#)

## Q 14. LRU Cache

Design a data structure that follows the constraints of a Least Recently Used (LRU) cache.

Implement the `LRUCache` class:

`LRUCache(int capacity)` Initialize the LRU cache with positive size capacity.

`int get(int key)` Return the value of the key if the key exists, otherwise return -1.

`void put(int key, int value)` Update 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.

The functions `get` and `put` must each run in  $O(1)$  average time complexity.

**Example 1:**

**Input**

`["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]`  
`[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]`

**Output**

`[null, null, null, 1, null, -1, null, -1, 3, 4]`

[Practice Now](#)

# STACKS & QUEUES

## Q 15. Remove K Digits

Given string num representing a non-negative integer num, and an integer k, return the smallest possible integer after removing k digits from num.

**Example 1:**

**Input:** num = "1432219", k = 3

**Output:** "1219"

[Practice Now](#)

## Q 16. Rotting Oranges

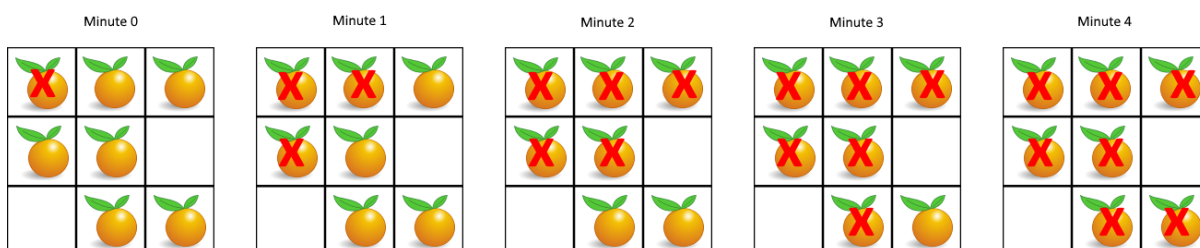
You are given an m x n grid where each cell can have one of three values:

- 0 representing an empty cell,
- 1 representing a fresh orange, or
- 2 representing a rotten orange.

Every minute, any fresh orange that is 4-directionally adjacent to a rotten orange becomes rotten.

Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return -1.

**Example 1:**



**Input:** grid = [[2,1,1],[1,1,0],[0,1,1]]

**Output:** 4

[Practice Now](#)

# STACKS & QUEUES

## Q 17. Maximal Rectangle

Given a rows x cols binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return its area

**Example 1:**

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0

**Input:** matrix =

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

**Output:** 6

[Practice Now](#)

## Q 18. Evaluate Reverse Polish Notation

You are given an array of strings tokens that represents an arithmetic expression in a Reverse Polish Notation.

Evaluate the expression. Return an integer that represents the value of the expression.

Note that:

- The valid operators are '+', '-', '\*', and '/'.
- Each operand may be an integer or another expression.
- The division between two integers always truncates toward zero.
- There will not be any division by zero.
- The input represents a valid arithmetic expression in a reverse polish notation.
- The answer and all the intermediate calculations can be represented in a 32-bit integer.

**Example 1:**

**Input:** tokens = `["2","1","+","3","*"]`

**Output:** 9

[Practice Now](#)



# RECURSION & BACKTRACKING

## Q 1. Subsets II

Given an integer array `nums` that may contain duplicates, return all possible Subsets (the power set).

The solution set must not contain duplicate subsets. Return the solution in any order.

**Example 1:**

**Input:** `nums = [1,2,2]`

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

[Practice Now](#)

## Q 2. Letter Combinations of a Phone Number

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



**Example 1:**

**Input:** `digits = "23"`

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

[Practice Now](#)

# RECURSION & BACKTRACKING

## Q 3. 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.

The same number may be chosen from candidates an unlimited number of times. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to target is less than 150 combinations for the given input.

**Example 1:**

**Input:** candidates = [2,3,6,7], target = 7

**Output:** [[2,2,3],[7]]

[Practice Now](#)

## Q 4. Combination Sum II

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

Each number in candidates may only be used once in the combination.

**Note:** The solution set must not contain duplicate combinations.

**Example 1:**

**Input:** candidates = [10,1,2,7,6,1,5], target = 8

**Output:**

```
[
  [1,1,6],
  [1,2,5],
  [1,7],
  [2,6]
]
```

[Practice Now](#)

# RECURSION & BACKTRACKING

## Q 5. Permutations

Given an array `nums` of distinct integers, return all the possible permutations. You can return the answer in any order.

**Example 1:**

**Input:** `nums = [1,2,3]`

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

[Practice Now](#)

## Q 6. Permutations II

Given a collection of numbers, `nums`, that might contain duplicates, return all possible unique permutations in any order.

**Example 1:**

**Input:** `nums = [1,1,2]`

**Output:**

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

[Practice Now](#)

## Q 7. Palindrome Partitioning

Given a string `s`, partition `s` such that every substring of the partition is a palindrome. Return all possible palindrome partitioning of `s`.

**Example 1:**

**Input:** `s = "aab"`

**Output:** `["a","a","b"],["aa","b"]`

[Practice Now](#)

# RECURSION & BACKTRACKING

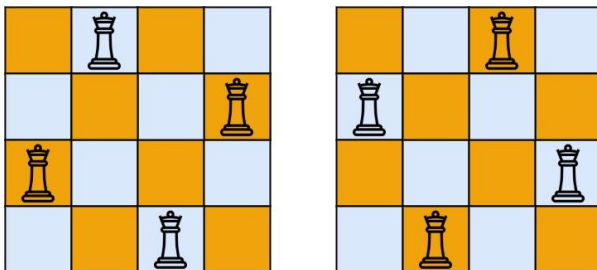
## Q 8. N-Queens

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

Given an integer n, return all distinct solutions to the n-queens puzzle. You may return the answer in any order.

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

**Example 1:**



**Input:** n = 4

**Output:** [".Q...", "...Q", "Q...", ".Q."], [".Q.", "Q...", "...Q", ".Q.."]]

[Practice Now](#)

## Q 9. Combination Sum III

Find all valid combinations of k numbers that sum up to n such that the following conditions are true:

- Only numbers 1 through 9 are used.
- Each number is used at most once.

Return a list of all possible valid combinations. The list must not contain the same combination twice, and the combinations may be returned in any order.

**Example 1:**

**Input:** k = 3, n = 7

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

[Practice Now](#)

# RECURSION & BACKTRACKING

## Q 10. Sudoku Solver

Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy all of the following rules:

Each of the digits 1-9 must occur exactly once in each row.

Each of the digits 1-9 must occur exactly once in each column.

Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.

The '.' character indicates empty cells.

**Example 1:**

**Input:** board =

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

**Output:**

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

[Practice Now](#)

**Want to up your Skill?**  
**Join our Popular courses**

DSA with  
System Design  
(HLD + LLD)

DSA & System  
Design with  
Full Stack

Advanced  
Data Science  
& Machine  
Learning

MERN Full  
Stack Web  
Development

# HEAPS

## Q 1. Kth Largest Element in an Array

Given an integer array `nums` and an integer `k`, return the `k`th largest element in the array.

Note that it is the `k`th largest element in the sorted order, not the `k`th distinct element.

You must solve it in  $O(n)$  time complexity.

**Example 1:**

**Input:** `nums = [3,2,1,5,6,4]`, `k = 2`

**Output:** 5

[Practice Now](#)

## Q 2. Ugly Number II

An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5.

Given an integer `n`, return the `n`th ugly number.

**Example 1:**

**Input:** `n = 10`

**Output:** 12

[Practice Now](#)

## Q 3. Sort Characters By Frequency

Given a string `s`, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string.

Return the sorted string. If there are multiple answers, return any of them.

**Example 1:**

**Input:** `s = "tree"`

**Output:** "eert"

[Practice Now](#)

# HEAPS

## Q 4. Top K Frequent Words

Given an array of strings words and an integer k, return the k most frequent strings.

Return the answer sorted by the frequency from highest to lowest. Sort the words with the same frequency by their lexicographical order.

**Example 1:**

**Input:** words = ["i","love","leetcode","i","love","coding"], k = 2

**Output:** ["i","love"]

[Practice Now](#)

## Q 5. Find Median from Data Stream

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

- For example, for arr = [2,3,4], the median is 3.
- For example, for arr = [2,3], the median is  $(2 + 3) / 2 = 2.5$ .

Implement the MedianFinder class:

- MedianFinder() initializes the MedianFinder object.
- void addNum(int num) adds the integer num from the data stream to the data structure.
- double findMedian() returns the median of all elements so far. Answers within 10<sup>-5</sup> of the actual answer will be accepted.

**Example 1:**

**Input**

["MedianFinder", "addNum", "addNum", "findMedian", "addNum", "findMedian"]  
[[ ], [1], [2], [ ], [3], [ ]]

**Output**

[null, null, null, 1.5, null, 2.0]

[Practice Now](#)



# HEAPS

## Q 6. Find K Pairs with Smallest Sums

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

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

Return the `k` pairs  $(u_1, v_1), (u_2, v_2), \dots, (u_k, v_k)$  with the smallest sums.

**Example 1:**

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

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

[Practice Now](#)

## Q 7. Kth Smallest Element in a Sorted Matrix

Given an  $n \times n$  matrix where each of the rows and columns is sorted in ascending order, return the `k`th smallest element in the matrix.

Note that it is the `k`th smallest element in the sorted order, not the `k`th distinct element.

You must find a solution with a memory complexity better than  $O(n^2)$ .

**Example 1:**

**Input:** `matrix = [[1,5,9],[10,11,13],[12,13,15]]`, `k = 8`

**Output:** 13

[Practice Now](#)

## For Admission Enquiry



**+91-7260058093**



**info@algotutor.io**



# HEAPS

## Q 8. Find K Pairs with Smallest Sums

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

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

Return the `k` pairs  $(u_1, v_1), (u_2, v_2), \dots, (u_k, v_k)$  with the smallest sums.

**Example 1:**

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

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

[Practice Now](#)

## Q 9. Kth Smallest Element in a Sorted Matrix

Given an  $n \times n$  matrix where each of the rows and columns is sorted in ascending order, return the `k`th smallest element in the matrix.

Note that it is the `k`th smallest element in the sorted order, not the `k`th distinct element.

You must find a solution with a memory complexity better than  $O(n^2)$ .

**Example 1:**

**Input:** `matrix = [[1,5,9],[10,11,13],[12,13,15]]`, `k = 8`

**Output:** 13

[Practice Now](#)

## Q 10. Reorganize String

Given a string `s`, rearrange the characters of `s` so that any two adjacent characters are not the same.

Return any possible rearrangement of `s` or return `""` if not possible.

**Example 1:**

**Input:** `s = "aab"`

**Output:** `"aba"`

[Practice Now](#)

# BINARY SEARCH

## Q 1. Median of Two Sorted Arrays

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return the median of the two sorted arrays.

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

**Example 1:**

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

**Output:** 2.00000

[Practice Now](#)

## Q 2. Divide Two Integers

Given two integers `dividend` and `divisor`, divide two integers without using multiplication, division, and mod operator.

The integer division should truncate toward zero, which means losing its fractional part. For example, 8.345 would be truncated to 8, and -2.7335 would be truncated to -2.

Return the quotient after dividing `dividend` by `divisor`.

Note: Assume we are dealing with an environment that could only store integers within the 32-bit signed integer range:  $[-2^{31}, 2^{31} - 1]$ . For this problem, if the quotient is strictly greater than  $2^{31} - 1$ , then return  $2^{31} - 1$ , and if the quotient is strictly less than  $-2^{31}$ , then return  $-2^{31}$ .

**Example 1:**

**Input:** `dividend = 10`, `divisor = 3`

**Output:** 3

[Practice Now](#)

# BINARY SEARCH

## Q 3. Most Beautiful Item for Each Query

You are given a 2D integer array `items` where `items[i] = [pricei, beautyi]` denotes the price and beauty of an item respectively.

You are also given a 0-indexed integer array `queries`. For each `queries[j]`, you want to determine the maximum beauty of an item whose price is less than or equal to `queries[j]`. If no such item exists, then the answer to this query is 0.

Return an array `answer` of the same length as `queries` where `answer[j]` is the answer to the `j`th query.

**Example 1:**

**Input:** `items = [[1,2],[3,2],[2,4],[5,6],[3,5]]`, **queries** = `[1,2,3,4,5,6]`

**Output:** `[2,4,5,5,6,6]`

[Practice Now](#)

## Q 4. Minimum Absolute Sum Difference

You are given two positive integer arrays `nums1` and `nums2`, both of length `n`.

The absolute sum difference of arrays `nums1` and `nums2` is defined as the sum of  $|nums1[i] - nums2[i]|$  for each  $0 \leq i < n$  (0-indexed).

You can replace at most one element of `nums1` with any other element in `nums1` to minimize the absolute sum difference.

Return the minimum absolute sum difference after replacing at most one element in the array `nums1`. Since the answer may be large, return it modulo  $10^9 + 7$ .

$|x|$  is defined as:

- $x$  if  $x \geq 0$ , or
- $-x$  if  $x < 0$ .

**Example 1:**

**Input:** `nums1 = [1,7,5]`, `nums2 = [2,3,5]`

**Output:** 3

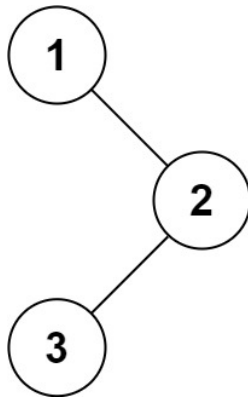
[Practice Now](#)

# BINARY TREE

## Q 1. Binary Tree Inorder Traversal

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

**Example 1:**



**Input:** root = [1,null,2,3]

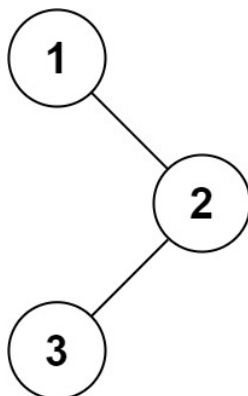
**Output:** [1,3,2]

[Practice Now](#)

## Q 2. Binary Tree Preorder Traversal

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

**Example 1:**



**Input:** root = [1,null,2,3]

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

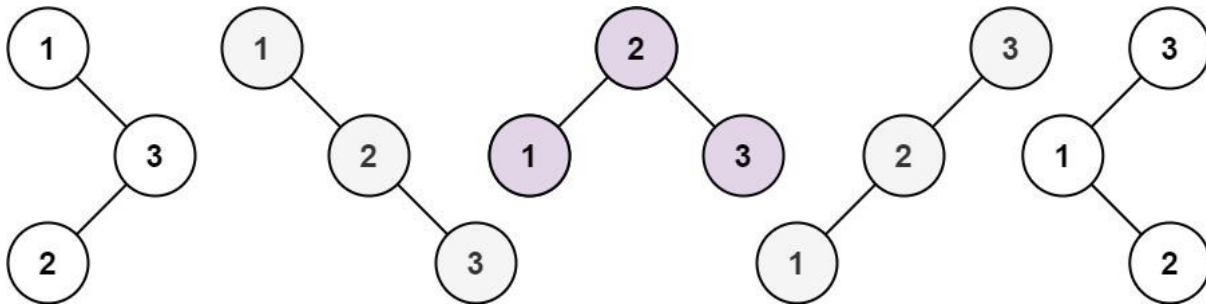
[Practice Now](#)

# BINARY TREE

## Q 3. Unique Binary Search Trees II

Given an integer  $n$ , return the number of structurally unique BST's (binary search trees) which has exactly  $n$  nodes of unique values from 1 to  $n$ .

**Example 1:**



**Input:**  $n = 3$

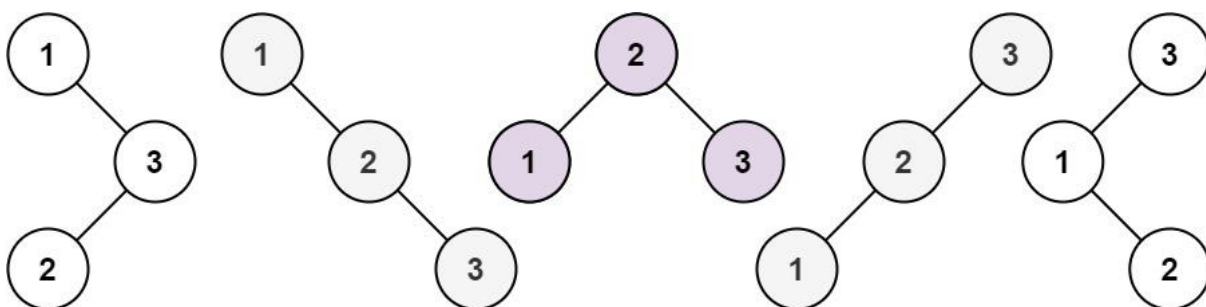
**Output:** 5

[Practice Now](#)

## Q 4. Unique Binary Search Trees II

Given an integer  $n$ , return all the structurally unique BST's (binary search trees), which has exactly  $n$  nodes of unique values from 1 to  $n$ . Return the answer in any order.

**Example 1:**



**Input:**  $n = 3$

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

[Practice Now](#)

# BINARY TREE

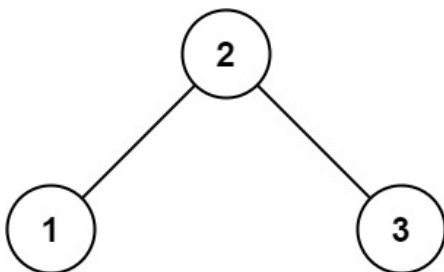
## Q 5. Validate Binary Search Tree

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 1:**



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

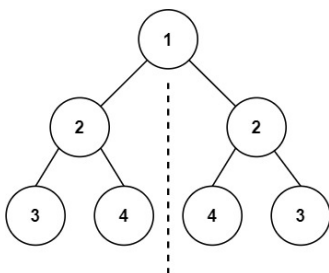
**Output:** true

[Practice Now](#)

## Q 6. Symmetric Tree

Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

**Example 1:**



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

**Output:** true

[Practice Now](#)

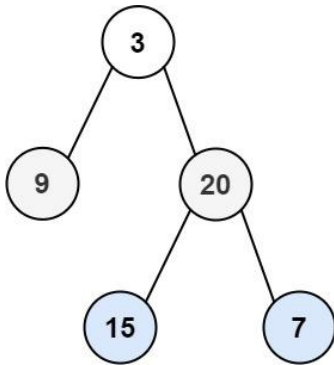


# BINARY TREE

## Q 7. Binary Tree Level Order Traversal

Given the root of a binary tree, return the level order traversal of its nodes' values. (i.e., from left to right, level by level).

**Example 1:**



**Input:** root = [3,9,20,null,null,15,7]

**Output:** [[3],[9,20],[15,7]]

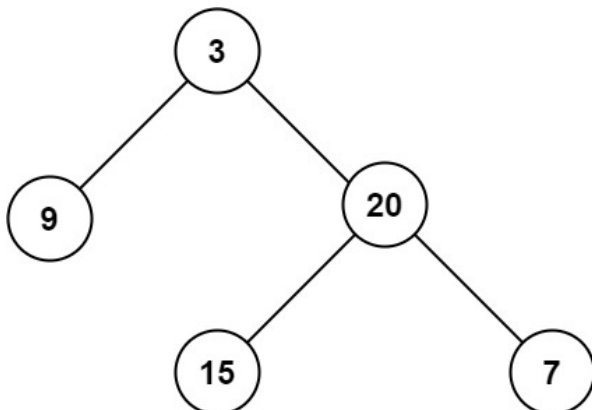
[Practice Now](#)

## Q 8. Maximum Depth of Binary Tree

Given the root of a binary tree, return its maximum depth.

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

**Example 1:**



**Input:** root = [3,9,20,null,null,15,7]

**Output:** 3

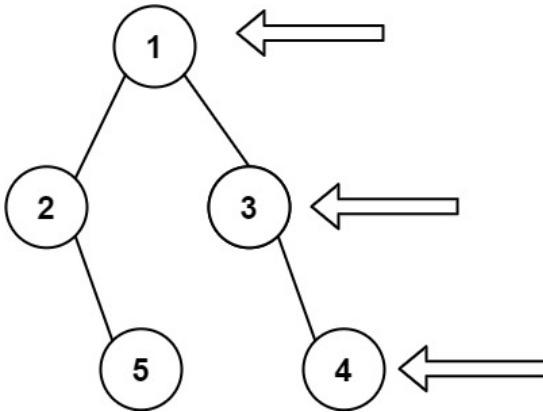
[Practice Now](#)

# BINARY TREE

## Q 9. Binary Tree Right Side View

Given the root of a binary tree, imagine yourself standing on the right side of it, return the values of the nodes you can see ordered from top to bottom.

**Example 1:**



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

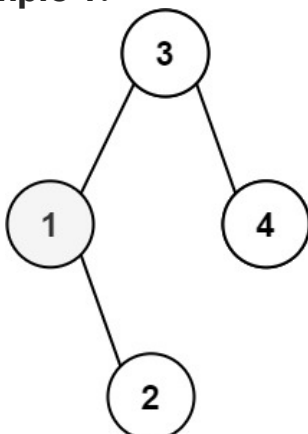
**Output:** [1,3,4]

[Practice Now](#)

## Q 10. Kth Smallest Element in a BST

Given the root of a binary search tree, and an integer k, return the kth smallest value (1-indexed) of all the values of the nodes in the tree.

**Example 1:**



**Input:** root = [3,1,4,null,2], k = 1

**Output:** 1

[Practice Now](#)

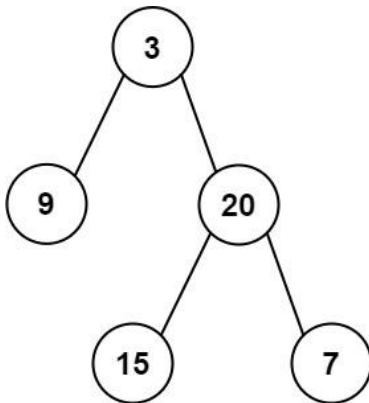
# BINARY TREE

## Q 11. Sum of Left Leaves

Given the root of a binary tree, return the sum of all left leaves.

A leaf is a node with no children. A left leaf is a leaf that is the left child of another node.

**Example 1:**



**Input:** root = [3,9,20,null,null,15,7]

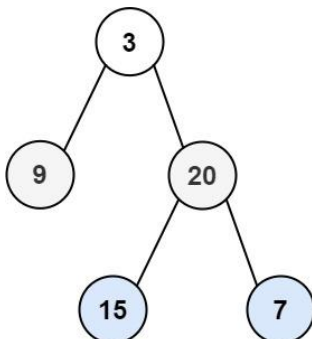
**Output:** 24

[Practice Now](#)

## Q 12. Binary Tree Zigzag Level Order Traversal

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

**Example 1:**



**Input:** root = [3,9,20,null,null,15,7]

**Output:** [[3],[20,9],[15,7]]

[Practice Now](#)

# BINARY TREE

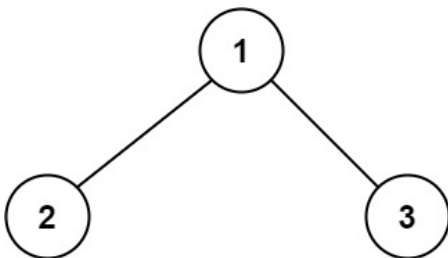
## Q 13. Binary Tree Maximum Path Sum

A path in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence at most once. Note that the path does not need to pass through the root.

The path sum of a path is the sum of the node's values in the path.

Given the root of a binary tree, return the maximum path sum of any non-empty path.

**Example 1:**



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

**Output:** 6

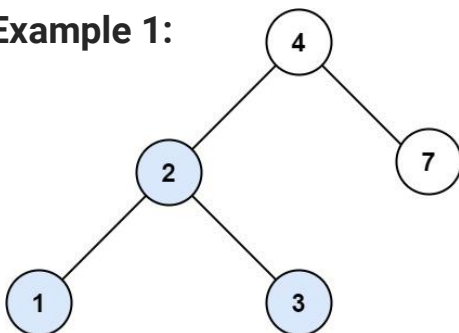
[Practice Now](#)

## Q 14. Search in a Binary Search Tree

You are given the root of a binary search tree (BST) and an integer val.

Find the node in the BST that the node's value equals val and return the subtree rooted with that node. If such a node does not exist, return null.

**Example 1:**



**Input:** root = [4,2,7,1,3], val = 2

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

[Practice Now](#)

# GRAPHS

## Q 1. Number of Islands

Given an  $m \times n$  2D binary 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

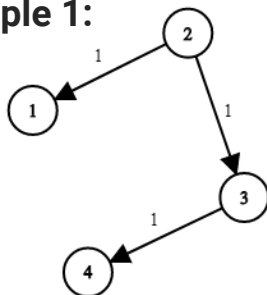
[Practice Now](#)

## Q 2. Network Delay Time

You are given a network of  $n$  nodes, labeled from 1 to  $n$ . You are also given times, a list of travel times as directed edges  $times[i] = (u_i, v_i, w_i)$ , where  $u_i$  is the source node,  $v_i$  is the target node, and  $w_i$  is the time it takes for a signal to travel from source to target.

We will send a signal from a given node  $k$ . Return the minimum time it takes for all the  $n$  nodes to receive the signal. If it is impossible for all the  $n$  nodes to receive the signal, return -1.

**Example 1:**



**Input:** times = [[2,1,1],[2,3,1],[3,4,1]],  $n = 4$ ,  $k = 2$

**Output:** 2

[Practice Now](#)

# GRAPHS

## Q 3. Decode String

Given an encoded string, return its decoded string.

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

You may assume that the input string is always valid; there are no extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers,  $k$ . For example, there will not be input like `3a` or `2[4]`.

The test cases are generated so that the length of the output will never exceed 105.

**Example 1:**

**Input:** `s = "3[a]2[bc]"`

**Output:** `"aaabcbcb"`

[Practice Now](#)

## Q 4. Shortest Bridge

You are given an  $n \times n$  binary matrix grid where 1 represents land and 0 represents water.

An island is a 4-directionally connected group of 1's not connected to any other 1's. There are exactly two islands in grid.

You may change 0's to 1's to connect the two islands to form one island.

Return the smallest number of 0's you must flip to connect the two islands.

**Example 1:**

**Input:** `grid = [[0,1],[1,0]]`

**Output:** 1

[Practice Now](#)

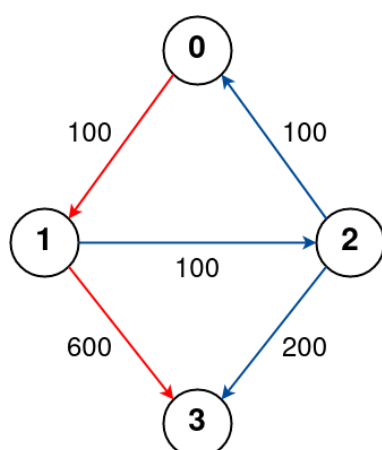
# GRAPHS

## Q 5. Cheapest Flights Within K Stops

There are  $n$  cities connected by some number of flights. You are given an array `flights` where `flights[i] = [fromi, toi, pricei]` indicates that there is a flight from city `fromi` to city `toi` with cost `pricei`.

You are also given three integers `src`, `dst`, and `k`, return the cheapest price from `src` to `dst` with at most `k` stops. If there is no such route, return -1.

**Example 1:**



**Input:** `n = 4`, `flights = [[0,1,100],[1,2,100],[2,0,100],[1,3,600],[2,3,200]]`, `src = 0`, `dst = 3`, `k = 1`

**Output:** 700

[Practice Now](#)

## Q 6. Remove Boxes

You are given several boxes with different colors represented by different positive numbers.

You may experience several rounds to remove boxes until there is no box left. Each time you can choose some continuous boxes with the same color (i.e., composed of  $k$  boxes,  $k \geq 1$ ), remove them and get  $k * k$  points.

Return the maximum points you can get.

**Input:** `boxes = [1,3,2,2,2,3,4,3,1]`

**Output:** 23

[Practice Now](#)



# GRAPHS

## Q 7. Maximum Number of Non-Overlapping Substrings

Given a string  $s$  of lowercase letters, you need to find the maximum number of non-empty substrings of  $s$  that meet the following conditions:

The substrings do not overlap, that is for any two substrings  $s[i..j]$  and  $s[x..y]$ , either  $j < x$  or  $i > y$  is true.

A substring that contains a certain character  $c$  must also contain all occurrences of  $c$ .

Find the maximum number of substrings that meet the above conditions. If there are multiple solutions with the same number of substrings, return the one with minimum total length. It can be shown that there exists a unique solution of minimum total length.

Notice that you can return the substrings in any order.

### Example 1:

**Input:**  $s = \text{"adefaddaccc"}$

**Output:**  $[\text{"e"}, \text{"f"}, \text{"ccc"}]$

Explanation: The following are all the possible substrings that meet the conditions:

```
[  
  "adefaddaccc"  
  "adefadda",  
  "ef",  
  "e",  
  "f",  
  "ccc",  
]
```

[Practice Now](#)

**For Admission Enquiry**



**+91-7260058093**



**info@algotutor.io**



# GRAPHS

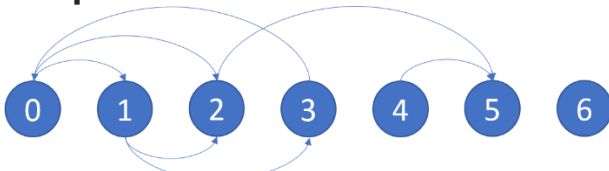
## Q 8. Find Eventual Safe States

There is a directed graph of  $n$  nodes with each node labeled from 0 to  $n - 1$ . The graph is represented by a 0-indexed 2D integer array `graph` where `graph[i]` is an integer array of nodes adjacent to node  $i$ , meaning there is an edge from node  $i$  to each node in `graph[i]`.

A node is a terminal node if there are no outgoing edges. A node is a safe node if every possible path starting from that node leads to a terminal node (or another safe node).

Return an array containing all the safe nodes of the graph. The answer should be sorted in ascending order.

**Example 1:**



**Input:** `graph = [[1,2],[2,3],[5],[0],[5],[],[]]`

**Output:** `[2,4,5,6]`

[Practice Now](#)

## Q 9. Word Ladder

A transformation sequence from word `beginWord` to word `endWord` using a dictionary `wordList` is a sequence of words `beginWord` -> `s1` -> `s2` -> ... -> `sk` such that:

- Every adjacent pair of words differs by a single letter.
- Every `si` for  $1 \leq i \leq k$  is in `wordList`. Note that `beginWord` does not need to be in `wordList`.
- `sk == endWord`

Given two words, `beginWord` and `endWord`, and a dictionary `wordList`, return the number of words in the shortest transformation sequence from `beginWord` to `endWord`, or 0 if no such sequence exists.

**Input:** `beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]`

**Output:** 5

[Practice Now](#)

# GRAPHS

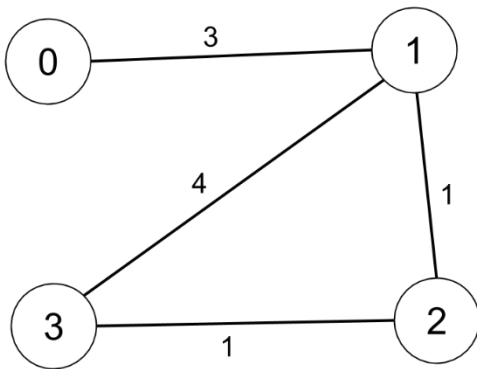
## Q 10. Find the City With the Smallest Number of Neighbors at a Threshold Distance

There are  $n$  cities numbered from 0 to  $n-1$ . Given the array `edges` where `edges[i] = [fromi, toi, weighti]` represents a bidirectional and weighted edge between cities `fromi` and `toi`, and given the integer `distanceThreshold`.

Return the city with the smallest number of cities that are reachable through some path and whose distance is at most `distanceThreshold`. If there are multiple such cities, return the city with the greatest number.

Notice that the distance of a path connecting cities  $i$  and  $j$  is equal to the sum of the edges' weights along that path.

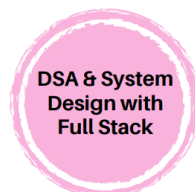
**Example 1:**



**Input:**  $n = 4$ , `edges = [[0,1,3],[1,2,1],[1,3,4],[2,3,1]]`, `distanceThreshold = 4`  
**Output:** 3

**Practice Now**

**Want to up your Skill?**  
Join our Popular courses



## Alumni Testimonial



★★★★★

comviva → J.P.Morgan

UpSkilling at AlgoTutor was really a great experience for me. They have very well structured curriculum and Instructors from Top Product based company.

Saloni Gupta

# GRAPHS

## Q 11. Minimum Cost to Make at Least One Valid Path in a Grid

Given an  $m \times n$  grid. Each cell of the grid has a sign pointing to the next cell you should visit if you are currently in this cell. The sign of  $\text{grid}[i][j]$  can be:

- 1 which means go to the cell to the right. (i.e go from  $\text{grid}[i][j]$  to  $\text{grid}[i][j + 1]$ )
- 2 which means go to the cell to the left. (i.e go from  $\text{grid}[i][j]$  to  $\text{grid}[i][j - 1]$ )
- 3 which means go to the lower cell. (i.e go from  $\text{grid}[i][j]$  to  $\text{grid}[i + 1][j]$ )
- 4 which means go to the upper cell. (i.e go from  $\text{grid}[i][j]$  to  $\text{grid}[i - 1][j]$ )

















Notice that there could be some signs on the cells of the grid that point outside the grid.

You will initially start at the upper left cell  $(0, 0)$ . A valid path in the grid is a path that starts from the upper left cell  $(0, 0)$  and ends at the bottom-right cell  $(m - 1, n - 1)$  following the signs on the grid. The valid path does not have to be the shortest.

You can modify the sign on a cell with cost = 1. You can modify the sign on a cell one time only.

Return the minimum cost to make the grid have at least one valid path.

**Example 1:**

**Input:** `grid = [[1,1,1,1],[2,2,2,2],[1,1,1,1],[2,2,2,2]]`

**Output:** 3

[Practice Now](#)

# DYNAMIC PROGRAMMING

## Q 1. Climbing Stairs

You are climbing a staircase. It takes  $n$  steps to reach the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

**Example 1:**

**Input:**  $n = 2$

**Output:** 2

[Practice Now](#)

## Q 2. Maximum Product Subarray

Given an integer array `nums`, find a subarray that has the largest product, and return the product.

The test cases are generated so that the answer will fit in a 32-bit integer.

**Input:** `nums = [2,3,-2,4]`

**Output:** 6

[Practice Now](#)

## Q 3. Longest Palindromic Substring

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

**Input:** `s = "babad"`

**Output:** "bab"

[Practice Now](#)

## Q 4. Maximum Subarray

Given an integer array `nums`, find the subarray with the largest sum, and return its sum.

**Input:** `nums = [-2,1,-3,4,-1,2,1,-5,4]`

**Output:** 6

[Practice Now](#)

# DYNAMIC PROGRAMMING

## Q 5. Longest Increasing Subsequence

Given an integer array `nums`, return the length of the longest strictly increasing subsequence.

**Example 1:**

**Input:** `nums = [10,9,2,5,3,7,101,18]`

**Output:** 4

[Practice Now](#)

## Q 6. Ones and Zeroes

You are given an array of binary strings `strs` and two integers `m` and `n`.

Return the size of the largest subset of `strs` such that there are at most `m` 0's and `n` 1's in the subset.

A set `x` is a subset of a set `y` if all elements of `x` are also elements of `y`.

**Input:** `strs = ["10","0001","111001","1","0"]`, `m = 5`, `n = 3`

**Output:** 4

[Practice Now](#)

## Q 7. Counting Bits

Given an integer `n`, return an array `ans` of length `n + 1` such that for each `i` ( $0 \leq i \leq n$ ), `ans[i]` is the number of 1's in the binary representation of `i`.

**Input:** `n = 2`

**Output:** `[0,1,1]`

**Explanation:**

0 --> 0

1 --> 1

2 --> 10

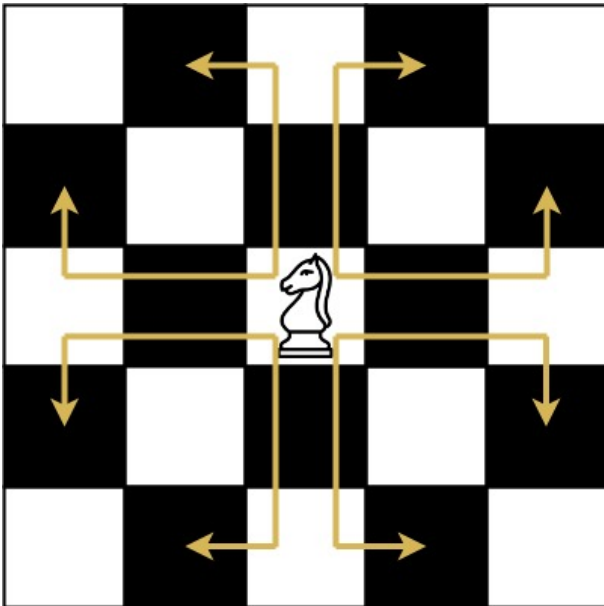
[Practice Now](#)

# DYNAMIC PROGRAMMING

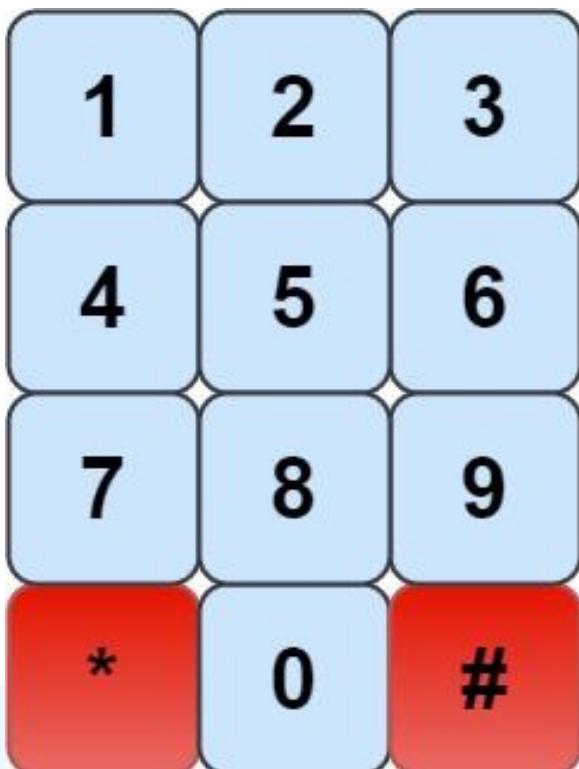
## Q 8. Knight Dialer

The chess knight has a unique movement, it may move two squares vertically and one square horizontally, or two squares horizontally and one square vertically (with both forming the shape of an L). The possible movements of chess knight are shown in this diagram:

A chess knight can move as indicated in the chess diagram below:



We have a chess knight and a phone pad as shown below, the knight can only stand on a numeric cell (i.e. blue cell).





Given an integer  $n$ , return how many distinct phone numbers of length  $n$  we can dial.

You are allowed to place the knight on any numeric cell initially and then you should perform  $n - 1$  jumps to dial a number of length  $n$ . All jumps should be valid knight jumps.

As the answer may be very large, return the answer modulo  $10^9 + 7$ .

**Example 1:**

**Input:**  $n = 1$

**Output:** 10

[Practice Now](#)

### Q 9. Coin Change

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.

**Example1:**

**Input:** `coins = [1,2,5]`, **amount** = 11

**Output:** 3

[Practice Now](#)

### Q 10. Partition Equal Subset Sum

Given an integer array `nums`, return true if you can partition the array into two subsets such that the sum of the elements in both subsets is equal or false otherwise.

**Example1:**

**Input:** `nums = [1,5,11,5]`

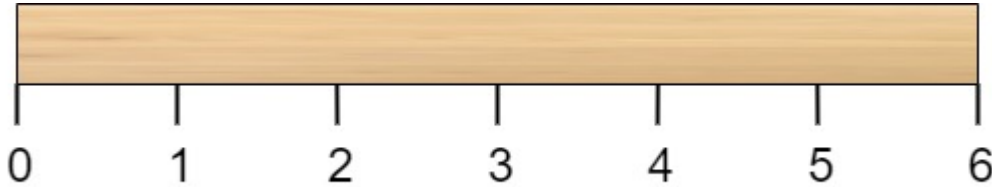
**Output:** true

[Practice Now](#)

# DYNAMIC PROGRAMMING

## Q 11. Minimum Cost to Cut a Stick

Given a wooden stick of length  $n$  units. The stick is labelled from 0 to  $n$ . For example, a stick of length 6 is labelled as follows:



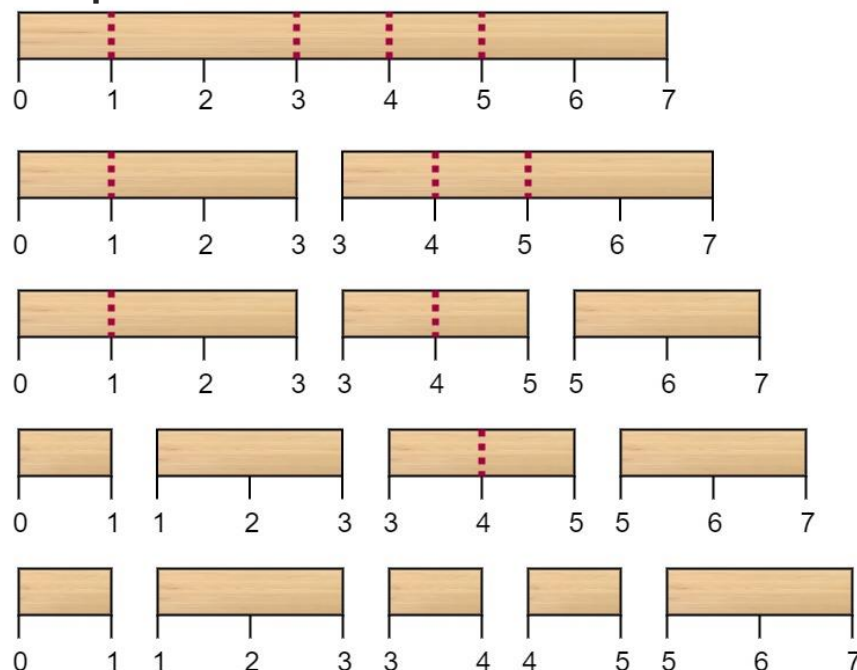
Given an integer array `cuts` where `cuts[i]` denotes a position you should perform a cut at.

You should perform the cuts in order, you can change the order of the cuts as you wish.

The cost of one cut is the length of the stick to be cut, the total cost is the sum of costs of all cuts. When you cut a stick, it will be split into two smaller sticks (i.e. the sum of their lengths is the length of the stick before the cut). Please refer to the first example for a better explanation.

Return the minimum total cost of the cuts.

**Example 1:** `cuts = [3, 5, 1, 4]` (Optimal Ordering)



**Input:**  $n = 7$ , `cuts = [1,3,4,5]`

**Output:** 16

[Practice Now](#)

# DYNAMIC PROGRAMMING

## Q 12. Best Time to Buy and Sell Stock with Cooldown

You are given an array prices where prices[i] is the price of a given stock on the ith day.

Find the maximum profit you can achieve. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times) with the following restrictions:

After you sell your stock, you cannot buy stock on the next day (i.e., cooldown one day).

**Note:** You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

**Example 1:**

**Input:** prices = [1,2,3,0,2]

**Output:** 3

[Practice Now](#)

## Q 13. Best Time to Buy and Sell Stock with Transaction Fee

You are given an array prices where prices[i] is the price of a given stock on the ith day, and an integer fee representing a transaction fee.

Find the maximum profit you can achieve. You may complete as many transactions as you like, but you need to pay the transaction fee for each transaction.

**Note:** You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

**Example 1:**

**Input:** prices = [1,3,2,8,4,9], fee = 2

**Output:** 8

[Practice Now](#)



# AlgoTutor

## Why Choose AlgoTutor?



**100 % PLACEMENT  
ASSISTANCE**



**1:1 PERSONAL  
MENTORSHIP FROM  
INDUSTRY EXPERTS**



**100 % SUCCESS  
RATE**



**23 LPA(AVG.) CTC**



**200+ SUCCESSFUL  
ALUMNI**



**147%(AVG.)  
SALARY HIKE**



**LEARN FROM  
SCRATCH**



**CAREER SERVICES**

**For More Information Contact Us:**

**+91-7260058093 || [info@algotutor.io](mailto:info@algotutor.io) || [www.algotutor.io](http://www.algotutor.io)**

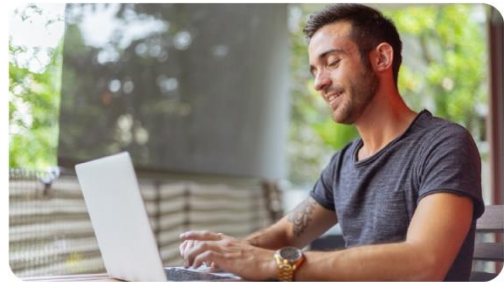


# AlgoTutor

## Want to up your Skill? Join our Popular courses



**DSA WITH SYSTEM  
DESIGN (HLD + LLD)**



**DSA & SYSTEM DESIGN  
WITH FULL STACK**



**ADVANCED DATA  
SCIENCE & MACHINE  
LEARNING**



**MERN FULL STACK  
WEB DEVELOPMENT**

**For More Information Contact Us:**

**+91-7260058093 || [info@algotutor.io](mailto:info@algotutor.io) || [www.algotutor.io](http://www.algotutor.io)**