DSA Interview Questions on Array

1. **Check if Pair with the Given Sum Exists in Array**
    - **Problem**: Given an array of integers and a target sum, find if there is a pair of elements that add up to the target sum.
    - **Example**:
        - Input: `array = [1, 4, 5, 6, 8], target = 9`
        - Output: `True`
        - Explanation: The pair (4, 5) adds up to the target sum of 9.
2. **Best Time to Buy and Sell Stock**
    - **Problem**: Given an array of prices where each element represents the stock price on a given day, find the maximum profit you can make by buying and selling the stock only once.
    - **Example**:
        - Input: `prices = [7, 1, 5, 3, 6, 4]`
        - Output: `5`
        - Explanation: Buy on day 2 (price 1) and sell on day 5 (price 6), profit = 6 - 1 = 5.
3. **Find Duplicates**
    - **Problem**: Given an array of integers, find all the elements that appear more than once.
    - **Example**:
        - Input: `array = [4, 3, 2, 7, 8, 2, 3, 1]`
        - Output: `[2, 3]`
        - Explanation: 2 and 3 are repeated.
4. **Product of Array Except Self**

- **Problem**: Given an array, return an array where each element is the product of all elements in the original array except the one at that index.
- **Example**:
    - Input: `array = [1, 2, 3, 4]`
    - Output: `[24, 12, 8, 6]`
    - Explanation: The product of all elements except the one at index `i`.

5. **Maximum Subarray**

- **Problem**: Find the contiguous subarray with the largest sum within a given array.
- **Example**:
    - Input: `array = [-2, 1, -3, 4, -1, 2, 1, -5, 4]`
    - Output: `6`
    - Explanation: The subarray `[4, -1, 2, 1]` has the largest sum of 6.

6. **Maximum Product Subarray**

- **Problem**: Find the contiguous subarray with the largest product within a given array.
- **Example**:
    - Input: `array = [2, 3, -2, 4]`
    - Output: `6`
    - Explanation: The subarray `[2, 3]` has the largest product of 6.

7. **Find Minimum in Rotated Sorted Array**

- **Problem**: Given a sorted, rotated array of unique integers, find the minimum element.
- **Example**:

- Input: `array = [3, 4, 5, 1, 2]`
- Output: `1`
- Explanation: The sorted array was rotated, and `1` is the smallest element.

8. **Search in Rotated Sorted Array**

- **Problem**: Given a sorted, rotated array of unique integers, find the index of a target value. If not present, return -1.
- **Example**:
  - Input: `array = [4, 5, 6, 7, 0, 1, 2], target = 0`
  - Output: `4`
  - Explanation: `0` is at index 4 in the rotated sorted array.

9. **3 Sum**

- **Problem**: Given an array of integers, find all unique triplets that sum up to zero.
- **Example**:
  - Input: `array = [-1, 0, 1, 2, -1, -4]`
  - Output: `[[-1, 0, 1], [-1, -1, 2]]`
  - Explanation: Unique triplets that sum up to zero.

10. **Container with Most Water**

- **Problem**: Given an array where each element represents the height of a line, find two lines that can form a container that holds the most water.
- **Example**:
  - Input: `height = [1, 8, 6, 2, 5, 4, 8, 3, 7]`
  - Output: `49`
  - Explanation: Lines at index 1 and index 8 form the container with the largest capacity.

11. **Find the Factorial of a Large Number**

- **Problem**: Given a large number `n`, calculate its factorial using an efficient method.
- **Example**:
    - Input: `n = 20`
    - Output: `2432902008176640000`
    - Explanation: Factorial of 20 is 2432902008176640000.

12. **Trapping Rain Water**
- **Problem**: Given an array where each element represents the height of a bar, calculate how much water can be trapped between the bars.
- **Example**:
    - Input: `height = [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]`
    - Output: `6`
    - Explanation: The water trapped is 6 units.

13. **Chocolate Distribution Problem**
- **Problem**: Given an array of packets where each packet contains some chocolates, distribute the packets among students such that the difference between the maximum and minimum chocolates received is minimized.
- **Example**:
    - Input: `array = [12, 4, 7, 9, 2, 23, 25, 41, 30, 40], students = 4`
    - Output: `7`
    - Explanation: Distributing packets `[7, 9, 12, 23]` gives the minimum difference of 7.

14. **Insert Interval**
- **Problem**: Given a set of non-overlapping intervals and a new interval, insert the new interval into the set and merge overlapping intervals if needed.
- **Example**:
    - Input: `intervals = [[1, 3], [6, 9]], newInterval = [2, 5]`
    - Output: `[[1, 5], [6, 9]]`
    - Explanation: The new interval is merged with `[1, 3]`.

15. **Merge Intervals**

- **Problem**: Given a set of overlapping intervals, merge them into a set of disjoint intervals.
- **Example**:
    - Input: `intervals = [[1, 3], [2, 6], [8, 10], [15, 18]]`
    - Output: `[[1, 6], [8, 10], [15, 18]]`
    - Explanation: `[1, 3]` and `[2, 6]` are merged into `[1, 6]`.

16. **Non-overlapping Intervals**

- **Problem**: Given a set of intervals, find the minimum number of intervals to remove so that the remaining intervals are non-overlapping.
- **Example**:
    - Input: `intervals = [[1, 2], [2, 3], [3, 4], [1, 3]]`
    - Output: `1`
    - Explanation: Removing `[1, 3]` makes the rest non-overlapping.

DSA Interview Questions on Matrix

1. **Set Matrix Zeroes**

   - **Problem**: Given a matrix, if an element is zero, set its entire row and column to zero.
   - **Example**:
     - Input:

       ```lua
       matrix = [[1, 1, 1], [1, 0, 1], [1, 1, 1]]
       ```

     - Output:

       ```lua
       [[1, 0, 1], [0, 0, 0], [1, 0, 1]]
       ```

     - Explanation: The second row and column are set to zero because of the zero at `[1, 1]`.

2. **Spiral Matrix**

   - **Problem**: Given a matrix, return all elements in a spiral order starting from the top-left corner.
   - **Example**:
     - Input:

       ```lua
       matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
       ```

     - Output: `[1, 2, 3, 6, 9, 8, 7, 4, 5]`
     - Explanation: The matrix is traversed in spiral order starting from the top-left.

3. **Program to Find the Transpose of a Matrix**

   - **Problem**: Given a matrix, find the transpose, which flips the matrix over its diagonal.
   - **Example**:

- Input:

```lua
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

- Output:

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

- Explanation: The matrix is flipped over its diagonal.

4. **Word Search**

- **Problem**: Given a grid of letters and a word, determine if the word exists in the grid. The word can be constructed by sequentially adjacent letters (horizontally or vertically) without revisiting any cell.
- **Example**:
  - Input:

```css
board = [['A', 'B', 'C', 'E'], ['S', 'F', 'C', 'S'], ['A', 'D', 'E', 'E']], word = "ABCCED"
```

  - Output: `True`
  - Explanation: The word "ABCCED" is found along the given path.

DSA Interview Questions on String

1. **Longest Substring Without Repeating Characters**
    - **Problem**: Given a string, find the length of the longest substring without repeating characters.
    - **Example**:
        - Input: `string = "abcabcbb"`
        - Output: `3`
        - Explanation: The longest substring without repeating characters is "abc," which has a length of 3.
2. **Longest Repeating Character Replacement**
    - **Problem**: Given a string and an integer `k`, find the length of the longest substring containing the same character, allowing at most `k` replacements.
    - **Example**:
        - Input: `string = "AABABBA", k = 1`
        - Output: `4`
        - Explanation: Replace one 'A' to get the longest repeating substring "BBBB."
3. **Smallest Window in a String Containing All Characters of Another String**
    - **Problem**: Given two strings `s` and `t`, find the smallest substring of `s` that contains all characters of `t`.
    - **Example**:
        - Input: `s = "ADOBECODEBANC", t = "ABC"`
        - Output: `"BANC"`
        - Explanation: The smallest window in "ADOBECODEBANC" that contains "A," "B," and "C" is "BANC."
4. **Check Whether Two Strings Are Anagram of Each Other**

- **Problem**: Given two strings, determine if one is an anagram of the other. An anagram is formed by rearranging the characters of a string.
- **Example**:
    - Input: `s1 = "listen", s2 = "silent"`
    - Output: `True`
    - Explanation: Both strings contain the same characters, rearranged.

5. **Print All Anagrams Together**

- **Problem**: Given an array of strings, group anagrams together.
- **Example**:
    - Input: `array = ["eat", "tea", "tan", "ate", "nat", "bat"]`
    - Output: `[["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]`
    - Explanation: Strings are grouped based on their anagram similarity.

6. **Check if Given Parentheses Expression is Balanced or Not**

- **Problem**: Given a string containing only parentheses characters `(`, `)`, `{`, `}`, `[`, and `]`, check if the expression is balanced.
- **Example**:
    - Input: `expression = "{[()()]}"`
    - Output: `True`
    - Explanation: The expression is balanced.

7. **Sentence Palindrome**

- **Problem**: Given a sentence, check if it is a palindrome, ignoring spaces, punctuation, and case.
- **Example**:
    - Input: `sentence = "A man, a plan, a canal, Panama!"`
    - Output: `True`

- Explanation: The sentence reads the same backward and forward when ignoring punctuation and case.

8. **Longest Palindromic Substring**

- **Problem**: Given a string, find the longest substring that is a palindrome.
- **Example**:
  - Input: `string = "babad"`
  - Output: `"bab"` or `"aba"`
  - Explanation: Both "bab" and "aba" are the longest palindromic substrings.

9. **Palindromic Substrings**

- **Problem**: Given a string, count how many substrings are palindromic.
- **Example**:
  - Input: `string = "aaa"`
  - Output: `6`
  - Explanation: All substrings are palindromic.

10. **Longest Common Prefix**

- **Problem**: Given an array of strings, find the longest common prefix among all strings.
- **Example**:
  - Input: `array = ["flower", "flow", "flight"]`
  - Output: `"fl"`
  - Explanation: "fl" is the longest common prefix.

DSA Interview Questions on Linked List

1. **Reverse a Linked List**
    - **Problem**: Given a singly linked list, reverse it.
    - **Example**:
        - Input: `list = 1 -> 2 -> 3 -> 4 -> 5`
        - Output: `5 -> 4 -> 3 -> 2 -> 1`
        - Explanation: The linked list is reversed.
2. **Detect Cycle in a Linked List**
    - **Problem**: Given a singly linked list, determine if a cycle is present.
    - **Example**:
        - Input: `list = 1 -> 2 -> 3 -> 4 -> 2` (cycle back to the second node)
        - Output: `True`
        - Explanation: A cycle is present starting from node 2.
3. **Merge Two Sorted Lists**
    - **Problem**: Given two sorted linked lists, merge them into a single sorted list.
    - **Example**:
        - Input:

          rust

          list1 = 1 -> 3 -> 5 list2 = 2 -> 4 -> 6
        - Output: `1 -> 2 -> 3 -> 4 -> 5 -> 6`
        - Explanation: Both sorted lists are merged.
4. **Merge K Sorted Lists**

- **Problem**: Given `k` sorted linked lists, merge them into a single sorted list.
- **Example**:
  - Input:

    ```
    rust

    list1 = 1 -> 4 -> 5 list2 = 1 -> 3 -> 4 list3 = 2 -> 6
    ```

  - Output: `1 -> 1 -> 2 -> 3 -> 4 -> 4 -> 5 -> 6`
  - Explanation: All `k` lists are merged.

5. **Remove Nth Node from End of List**

- **Problem**: Given a singly linked list and an integer `n`, remove the nth node from the end of the list.
- **Example**:
  - Input: `list = 1 -> 2 -> 3 -> 4 -> 5, n = 2`
  - Output: `1 -> 2 -> 3 -> 5`
  - Explanation: The 4th node from the end is removed.

6. **Reorder List**

- **Problem**: Given a singly linked list, reorder it as follows: first node, last node, second node, second last node, etc.
- **Example**:
  - Input: `list = 1 -> 2 -> 3 -> 4 -> 5`
  - Output: `1 -> 5 -> 2 -> 4 -> 3`
  - Explanation: The list is reordered.

7. **Add 1 to a Number Represented as Linked List**

- **Problem**: Given a linked list where each node contains a digit of a number, add 1 to the number.
- **Example**:
  - Input: `list = 1 -> 9 -> 9`

- Output: `2 -> 0 -> 0`
- Explanation: The number 199 becomes 200.

8. **Find the Middle of a Given Linked List**

- **Problem**: Given a singly linked list, find the middle node. If the list has an even number of nodes, return the second middle node.
- **Example**:
  - Input: `list = 1 -> 2 -> 3 -> 4 -> 5`
  - Output: `3`
  - Explanation: Node 3 is the middle node.

9. **Delete Last Occurrence of an Item from Linked List**

- **Problem**: Given a singly linked list and an item `x`, remove the last occurrence of `x` from the list.
- **Example**:
  - Input: `list = 1 -> 2 -> 3 -> 4 -> 2 -> 5, x = 2`
  - Output: `1 -> 2 -> 3 -> 4 -> 5`
  - Explanation: The last occurrence of 2 is removed.

DSA Interview Questions on Stack & Queue

1. **Convert Infix Expression to Postfix Expression**

    - **Problem**: Given a mathematical expression in infix notation, convert it to postfix notation. The infix notation uses operators placed between operands, while the postfix notation places operators after the operands.
    - **Example**:
        - Input: `"A + B * C"`
        - Output: `"A B C * +"`
        - Explanation: The expression is converted to postfix notation.

2. **Next Greater Element**

    - **Problem**: Given an array, find the next greater element for each element in the array. The next greater element of an element `x` is the first greater element to the right of `x`. If no such element exists, output `-1` for that position.
    - **Example**:
        - Input: `[4, 5, 2, 10, 8]`
        - Output: `[5, 10, 10, -1, -1]`
        - Explanation: For 4, the next greater element is 5; for 5, it's 10; for 2, it's 10; 10 has no greater element, so it's -1; and 8 also has no greater element, so it's -1.

3. **Delete Middle Element of a Stack**

    - **Problem**: Given a stack of integers, delete the middle element without using any additional data structure. If the stack has an even number of elements, remove the element closer to the top of the stack.
    - **Example**:
        - Input: `stack = [1, 2, 3, 4, 5]`

- Output: `[1, 2, 4, 5]`
- Explanation: The middle element 3 is removed.

4. **Check Mirror in n-ary Tree**

- **Problem**: Given two n-ary trees, determine if one is the mirror image of the other.
- **Example**:
  - Input:

    yaml

    Tree 1: Tree 2: 1 1 / | \ / | \ 2 3 4 4 3 2
  - Output: `True`
  - Explanation: The two trees are mirrors of each other.

5. **The Celebrity Problem**

- **Problem**: Given `n` people at a party, find the celebrity. The celebrity is someone who knows no one but is known by everyone. The information about who knows whom is provided through a matrix.
- **Example**:
  - Input:

    lua

    matrix = [[0, 1, 0], [0, 0, 0], [0, 1, 0]]
  - Output: `1`
  - Explanation: Person 1 is the celebrity since they don't know anyone but are known by everyone.

6. **Length of the Longest Valid Substring**

- **Problem**: Given a string containing only parentheses characters `(` and `)`, find the length of the longest valid (well-formed) substring.
- **Example**:

- Input: `"(()))())("`
- Output: `4`
- Explanation: The longest valid substring is `"(())"` with a length of 4.

7. **Print Right View of a Binary Tree**

- **Problem**: Given a binary tree, print its right view. The right view is defined as the nodes visible when the tree is viewed from the right side.
- **Example**:
  - Input:

    makefile

    ```
    tree = 1 / \ 2 3 / \ \ 4 5 6
    ```

  - Output: `[1, 3, 6]`
  - Explanation: The right view shows nodes 1, 3, and 6.

8. **Find the First Circular Tour That Visits All Petrol Pumps**

- **Problem**: Given an array of petrol pumps, each containing fuel capacity and distance to the next petrol pump, find the first index from which a truck can complete a circular tour and visit all petrol pumps. The truck has unlimited fuel capacity but starts with an empty tank.
- **Example**:
  - Input:

    css

    ```
    pumps = [(6, 4), (3, 6), (7, 3)]
    ```

  - Output: `2`
  - Explanation: Starting at index 2, the truck can complete the tour because it gathers enough fuel at each pump.

DSA Interview Questions on Tree

1. **Maximum Depth of Binary Tree**

   - **Problem**: Given a binary tree, find its maximum depth. The depth is the number of nodes along the longest path from the root to a leaf node.
   - **Example**:
     - Input:

       ```mathematica
       tree = [3, 9, 20, None, None, 15, 7]
       ```

     - Output: `3`
     - Explanation: The maximum depth is 3, with the longest path from root 3 to leaf 15 or 7.

2. **Check if Two Trees Have the Same Structure**

   - **Problem**: Given two binary trees, determine if they have the same structure. Two trees are considered to have the same structure if they are structurally identical and their nodes have the same values.
   - **Example**:
     - Input:

       ```css
       tree1 = [1, 2, 3] tree2 = [1, 2, 3]
       ```

     - Output: `True`
     - Explanation: Both trees have identical structures and values.

3. **Invert/Flip Binary Tree**

   - **Problem**: Given a binary tree, invert it (mirror the tree across its root).
   - **Example**:

- Input:

```
css

tree = [4, 2, 7, 1, 3, 6, 9]
```

- Output:

```
csharp

[4, 7, 2, 9, 6, 3, 1]
```

- Explanation: The original tree is mirrored.

4. **Binary Tree Maximum Path Sum**

- **Problem**: Given a binary tree, find the maximum path sum. The path can start and end at any node and includes the sum of node values along the path.
- **Example**:
  - Input:

```
css

tree = [1, 2, 3]
```

  - Output: **6**
  - Explanation: The maximum path sum is 2 -> 1 -> 3.

5. **Binary Tree Level Order Traversal**

- **Problem**: Given a binary tree, return its level order traversal (from left to right, level by level).
- **Example**:
  - Input:

```
mathematica

tree = [3, 9, 20, None, None, 15, 7]
```

  - Output:

```
lua

[[3], [9, 20], [15, 7]]
```

- Explanation: The tree is traversed level by level.

6. **Serialize and Deserialize Binary Tree**

- **Problem**: Design an algorithm to serialize a binary tree to a string and deserialize it back to a binary tree.
- **Example**:
  - Input:

    ```
    mathematica

    tree = [1, 2, 3, None, None, 4, 5]
    ```

  - Output:

    ```
    arduino

    "1 2 # # 3 4 5"
    ```

  - Explanation: The serialized format can be used to reconstruct the binary tree.

7. **Subtree of Another Tree**

- **Problem**: Given two binary trees `root` and `subRoot`, determine if `subRoot` is a subtree of `root`.
- **Example**:
  - Input:

    ```
    css

    root = [3, 4, 5, 1, 2] subRoot = [4, 1, 2]
    ```

  - Output: `True`
  - Explanation: The subtree [4, 1, 2] is present within [3, 4, 5, 1, 2].

8. **Construct Binary Tree from Preorder and Inorder Traversal**

- **Problem**: Given the preorder and inorder traversal sequences of a binary tree, construct the binary tree.
- **Example**:

- Input:

```
css

preorder = [3, 9, 20, 15, 7] inorder = [9, 3, 15, 20, 7]
```

- Output:

```
mathematica

[3, 9, 20, None, None, 15, 7]
```

- Explanation: The binary tree is constructed based on the traversal sequences.

9. **Validate Binary Search Tree**

- **Problem**: Given a binary tree, determine if it is a binary search tree (BST). In a BST, for every node, the left subtree contains only nodes with values less than the node, and the right subtree contains only nodes with values greater than the node.
- **Example**:
  - Input:

```
css

tree = [2, 1, 3]
```

  - Output: `True`
  - Explanation: The tree satisfies the BST properties.

10. **Kth Smallest Element in a BST**

- **Problem**: Given a binary search tree, find the kth smallest element.
- **Example**:
  - Input:

```
css

tree = [3, 1, 4, None, 2], k = 1
```

  - Output: `1`
  - Explanation: The first smallest element is 1.

11. **Lowest Common Ancestor of BST**
- **Problem**: Given a binary search tree and two nodes `p` and `q`, find their lowest common ancestor.
- **Example**:
    - Input:

```
css

tree = [6, 2, 8, 0, 4, 7, 9, None, None, 3, 5], p = 2, q = 8
```

    - Output: `6`
    - Explanation: The lowest common ancestor of nodes 2 and 8 is 6.

12. **Implement Trie (Prefix Tree)**
- **Problem**: Design a trie data structure that supports the following operations: insert a word, search for a word, and check if any word starts with a given prefix.
- **Example**:
    - Input:

```
graphql

trie.insert("apple") trie.search("apple") -> True trie.search("app") -> False trie.startsWith("app") -> True
```

    - Output: `[True, False, True]`
    - Explanation: The trie correctly supports the search and prefix operations.

13. **Add and Search Word**
- **Problem**: Design a data structure that supports adding a word and searching for a word, including regular expression searches with `.` representing any letter.
- **Example**:
    - Input:

```
graphql

wordDict.addWord("bad") wordDict.addWord("dad") wordDict.search("b..") -> True
```

    - Output: `True`

- Explanation: The search correctly finds words that match the pattern "b..".

DSA Interview Questions on Heap

1. **Top K Frequent Elements**

   - **Problem**: Given a non-empty array of integers and an integer `k`, find the `k` most frequent elements.
   - **Example**:
     - Input: `nums = [1, 1, 1, 2, 2, 3], k = 2`
     - Output: `[1, 2]`
     - Explanation: The two most frequent elements are 1 (appearing 3 times) and 2 (appearing 2 times).

2. **Find Median from Data Stream**

   - **Problem**: Design a data structure that supports the insertion of integers and calculates the median of all elements in the data stream efficiently.
   - **Example**:
     - Input:

       scss

       ```
       MedianFinder.insert(1) MedianFinder.insert(2) MedianFinder.findMedian() -> 1.5 MedianFinder.insert(3) MedianFinder.findMedian() -> 2
       ```

     - Output: `[1.5, 2]`
     - Explanation: The median changes as new elements are added.

3. **Largest Triplet Product in a Stream**

   - **Problem**: Given a stream of integers, output the largest product of any three distinct integers encountered so far. If fewer than three integers are available, return -1.
   - **Example**:
     - Input:

       css

`stream = [2, 1, 3, 4, 6]`

- Output: **`[ -1, -1, 6, 24, 72 ]`**
- Explanation: The products are calculated as the stream progresses, starting from 6 (2*1*3) to 72 (4*6*3).

4. **Connect n Ropes with Minimum Cost**

- **Problem**: Given an array of **`n`** ropes, where each rope has a specific length, connect all the ropes into a single rope with minimum cost. The cost to connect two ropes is the sum of their lengths.
- **Example**:
  - Input: **`ropes = [4, 3, 2, 6]`**
  - Output: **`29`**
  - Explanation: First connect 2 and 3 (cost 5), then connect the result with 4 (cost 9), then connect the result with 6 (cost 14), resulting in a total cost of 29.

DSA Interview Questions on Graph

1. **Clone Graph**
    - **Problem**: Given a graph represented by a node, create a deep copy of the graph. Each node contains a unique value and a list of its neighbors.
    - **Example**:
        - Input:

```yaml
node = { value: 1, neighbors: [{ value: 2 }, { value: 3 }] }
```

        - Output:

```yaml
clonedNode = { value: 1, neighbors: [{ value: 2 }, { value: 3 }] }
```

        - Explanation: The graph is deeply copied with the same structure and values.
2. **Course Schedule**
    - **Problem**: There are `n` courses to take, labeled from `0` to `n-1`. Each course may have prerequisites that must be completed before taking the course. Given a list of prerequisite pairs, determine if it is possible to complete all courses.
    - **Example**:
        - Input:

```lua
numCourses = 2, prerequisites = [[1, 0]]
```

        - Output: `True`
        - Explanation: It's possible to complete course 1 after course 0.
3. **Pacific Atlantic Water Flow**

- **Problem**: Given an `m x n` matrix representing heights of land, find all points where water can flow to both the Pacific and Atlantic oceans. Water flows downhill and can flow from one cell to its neighboring cells if the neighboring cell's height is less than or equal to the current cell's height.
- **Example**:
  - Input:

    ```
    css

    heights = [[1, 2, 2, 3, 5], [3, 2, 3, 4, 4], [2, 4, 5, 3, 1], [6, 7, 1, 4, 5], [5, 1, 1, 2, 4]]
    ```
  - Output: **`[[0, 4], [1, 3], [1, 4], [2, 2], [3, 0], [3, 1], [4, 0]]`**
  - Explanation: These points can flow to both the Pacific and Atlantic oceans.

4. **Number of Islands**

- **Problem**: Given a 2D grid where '1' represents land and '0' represents water, count the number of islands. An island is formed by connecting adjacent lands horizontally or vertically.
- **Example**:
  - Input:

    ```
    css

    grid = [[1, 1, 0, 0, 0], [1, 1, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 1, 1]]
    ```
  - Output: **`3`**
  - Explanation: There are three islands in the grid.

5. **Longest Consecutive Sequence**

- **Problem**: Given an unsorted array of integers, find the length of the longest consecutive sequence of numbers.
- **Example**:
  - Input: **`[100, 4, 200, 1, 3, 2]`**
  - Output: **`4`**

- Explanation: The longest consecutive sequence is `[1, 2, 3, 4]`.

6. **Snake and Ladder Problem**

- **Problem**: Given a board configuration for a snake and ladder game, where snakes move you down and ladders move you up, determine the minimum number of dice throws required to reach the final square.
- **Example**:
  - Input:

    ```yaml
    snakes_ladders = {3: 22, 5: 8, 11: 26, 20: 29, 27: 1, 21: 9, 17: 4, 19: 7}
    ```

  - Output: `3`
  - Explanation: The minimum throws required to reach the final square are 3.

7. **Detect Cycle in a Directed Graph**

- **Problem**: Given a directed graph, determine if it contains a cycle. The graph is represented using adjacency lists.
- **Example**:
  - Input:

    ```lua
    numVertices = 4 edges = [[0, 1], [1, 2], [2, 0], [3, 3]]
    ```

  - Output: `True`
  - Explanation: The graph contains cycles [0 -> 1 -> 2 -> 0] and [3 -> 3].

8. **Bridges in a Graph**

- **Problem**: In a connected undirected graph, find all bridges. A bridge (cut-edge) is an edge that, when removed, disconnects the graph.
- **Example**:
  - Input:

    ```
    numVertices = 5 edges = [[0, 1], [0, 2], [1, 2], [1, 3], [3, 4]]
    ```

  - Output: `[[1, 3], [3, 4]]`
  - Explanation: Removing these edges disconnects the graph.

9. **Check Whether a Given Graph is Bipartite or Not**

- **Problem**: Given an undirected graph, determine if it is bipartite. A bipartite graph can be colored using two colors such that no two adjacent nodes have the same color.
- **Example**:
  - Input:

    ```
    numVertices = 4 edges = [[0, 1], [1, 2], [2, 3], [3, 0]]
    ```

  - Output: `False`
  - Explanation: The graph cannot be colored using two colors due to the presence of a cycle with an odd number of vertices.

10. **Find Size of the Largest Region in Boolean Matrix**

- **Problem**: Given a binary matrix where '1' represents land and '0' represents water, find the size of the largest region of connected lands.
- **Example**:
  - Input:

    ```
    ```

```
grid = [[0, 0, 1, 1, 0], [1, 1, 0, 1, 0], [0, 1, 1, 1, 1], [0, 0, 0, 1, 0]]
```

- Output: `8`
- Explanation: The largest region of connected lands is of size 8.

11. **Flood Fill Algorithm**

- **Problem**: Implement a flood fill algorithm. Given a 2D grid, a starting cell `(sr, sc)`, and a new color, replace the starting cell's connected region with the new color.
- **Example**:
  - Input:

```lua
grid = [[1, 1, 1], [1, 1, 0], [1, 0, 1]] sr = 1, sc = 1, newColor = 2
```

  - Output:

```lua
[[2, 2, 2], [2, 2, 0], [2, 0, 1]]
```

  - Explanation: The region starting from cell (1, 1) is filled with color 2.

12. **Strongly Connected Components**

- **Problem**: In a directed graph, identify all strongly connected components (SCCs). An SCC is a subgraph where every node can reach every other node.
- **Example**:
  - Input:

```css
numVertices = 5 edges = [[0, 1], [1, 2], [2, 0], [1, 3], [3, 4]]
```

  - Output: `[[0, 1, 2], [3], [4]]`
  - Explanation: The graph has three strongly connected components.

13. **Topological Sorting**

- **Problem**: Given a directed graph, perform a topological sort on the graph. In topological sorting, each node is ordered such that for every directed edge `u -> v`, `u` comes before `v` in the ordering.

- **Example**:
  - Input:

    ```
    numVertices = 6 edges = [[5, 2], [5, 0], [4, 0], [4, 1], [2, 3], [3, 1]]
    ```

  - Output: **`[5, 4, 2, 3, 1, 0]`**
  - Explanation: The topological ordering is one possible valid ordering.

DSA Interview Questions on Dynamic Programming

1. **Count Ways to Reach the n'th Stair**

   - **Problem**: Given `n` stairs, each step can be climbed one, two, or three stairs at a time. Calculate the number of ways to reach the top (nth) stair.
   - **Example**:
     - Input: `n = 4`
     - Output: `7`
     - Explanation: The possible ways are `[1, 1, 1, 1]`, `[1, 1, 2]`, `[1, 2, 1]`, `[2, 1, 1]`, `[2, 2]`, `[1, 3]`, and `[3, 1]`.

2. **Coin Change**

   - **Problem**: Given a list of coin denominations and an integer amount, find the minimum number of coins required to make up the given amount. If it's not possible, return -1.
   - **Example**:
     - Input:

       ```
       css

       coins = [1, 2, 5], amount = 11
       ```

     - Output: `3`
     - Explanation: 11 can be made up of three coins: `[5, 5, 1]`.

3. **0/1 Knapsack Problem**

   - **Problem**: Given weights and values of `n` items, and a knapsack with a maximum weight capacity, determine the maximum value that can be carried. Each item can only be selected once.
   - **Example**:
     - Input:

       ```
       css

       weights = [1, 3, 4, 5], values = [1, 4, 5, 7], maxWeight = 7
       ```

- Output: `9`
- Explanation: The optimal selection is `[3, 4]`, with values summing up to 9.

4. **Longest Increasing Subsequence**

- **Problem**: Given an array of integers, find the length of the longest increasing subsequence.
- **Example**:
  - Input: `nums = [10, 9, 2, 5, 3, 7, 101, 18]`
  - Output: `4`
  - Explanation: The longest increasing subsequence is `[2, 3, 7, 101]`.

5. **Longest Common Subsequence**

- **Problem**: Given two strings, find the length of the longest subsequence that is common to both strings.
- **Example**:
  - Input: `text1 = "abcde", text2 = "ace"`
  - Output: `3`
  - Explanation: The longest common subsequence is `"ace"`.

6. **Word Break Problem**

- **Problem**: Given a string and a dictionary of words, determine if the string can be segmented into one or more dictionary words.
- **Example**:
  - Input:

    makefile

    s = "leetcode", wordDict = ["leet", "code"]
  - Output: `True`

- Explanation: The string can be segmented into "leet" and "code".

7. **Dice Throw**

- **Problem**: Given `n` dice, each with `m` faces, and a target sum `x`, find the number of ways to get the sum `x`.
- **Example**:
  - Input:

    ```
    n = 2, m = 6, x = 7
    ```

  - Output: `6`
  - Explanation: There are six ways to roll two six-faced dice to reach a sum of 7.

8. **Egg Dropping Puzzle**

- **Problem**: Given `k` eggs and a building with `n` floors, determine the minimum number of attempts required to find the highest floor from which an egg can be dropped without breaking.
- **Example**:
  - Input: `k = 2, n = 10`
  - Output: `4`
  - Explanation: The minimum attempts required are 4.

9. **Matrix Chain Multiplication**

- **Problem**: Given a sequence of matrices, find the most efficient way to multiply them to minimize the number of multiplications required.
- **Example**:
  - Input:

    ```
    matrices = [10, 20, 30, 40, 30]
    ```

- Output: `26000`
- Explanation: The minimum cost to multiply the sequence is 26000.

10. **Combination Sum**

- **Problem**: Given an array of integers and a target sum, find all unique combinations of numbers that add up to the target. The same number can be chosen multiple times.
- **Example**:
  - Input:

    ```
    candidates = [2, 3, 6, 7], target = 7
    ```

  - Output: `[[2, 2, 3], [7]]`
  - Explanation: These combinations sum up to the target.

11. **Subset Sum Problem**

- **Problem**: Given a set of positive integers and a target sum, determine if a subset with the given sum exists.
- **Example**:
  - Input:

    ```
    nums = [3, 34, 4, 12, 5, 2], sum = 9
    ```

  - Output: `True`
  - Explanation: The subset `[4, 5]` sums up to 9.

12. **Find Maximum Possible Stolen Value from Houses**

- **Problem**: In a street of houses, no two adjacent houses can be robbed on the same night. Find the maximum amount that can be stolen.
- **Example**:

- Input: `houses = [6, 7, 1, 30, 8, 2, 4]`
- Output: `41`
- Explanation: The maximum value can be obtained by robbing houses `[7, 30, 4]`.

13. **Count Possible Decodings of a Given Digit Sequence**

- **Problem**: Given a digit string, count the number of possible decodings according to the rules where 'A' corresponds to 1, 'B' corresponds to 2, ..., 'Z' corresponds to 26.
- **Example**:
  - Input: `digits = "226"`
  - Output: `3`
  - Explanation: The possible decodings are `"BZ"`, `"VF"`, and `"BBF"`.

14. **Unique Paths in a Grid with Obstacles**

- **Problem**: Given a 2D grid where some cells have obstacles, calculate the number of unique paths from the top-left corner to the bottom-right corner.
- **Example**:
  - Input:

    lua

    grid = [[0, 0, 0], [0, 1, 0], [0, 0, 0]]
  - Output: `2`
  - Explanation: The two paths to reach the destination are around the obstacle.

15. **Jump Game**

- **Problem**: Given an array of non-negative integers, where each integer represents the maximum number of steps one can jump forward, determine if you can reach the last index.

- **Example**:
  - Input: `nums = [2, 3, 1, 1, 4]`
  - Output: `True`
  - Explanation: It's possible to reach the last index by jumping through the sequence `[2 -> 3 -> 4]`.

16. **Cutting a Rod**

- **Problem**: Given a rod of length `n` and prices of rods of varying lengths, determine the maximum revenue obtainable by cutting and selling the rod in pieces.
- **Example**:
  - Input:

    ```
    css

    prices = [1, 5, 8, 9, 10, 17, 17, 20], n = 8
    ```

  - Output: `22`
  - Explanation: The best way to cut and sell the rod yields a revenue of 22.

17. **Maximum Product Cutting**

- **Problem**: Given a positive integer `n`, break the integer into at least two parts such that the product of the parts is maximized. Return the maximum product.
- **Example**:
  - Input: `n = 10`
  - Output: `36`
  - Explanation: The best product is obtained by breaking the integer into `[3, 3, 4]`.

18. **Count Number of Ways to Cover a Distance**

- **Problem**: Given a distance `d`, count the number of ways to cover the distance using steps of 1, 2, or 3 units.

- **Example**:
    - Input: `d = 4`
    - Output: `7`
    - Explanation: The ways to cover the distance are `[1, 1, 1, 1]`, `[1, 1, 2]`, `[1, 2, 1]`, `[2, 1, 1]`, `[2, 2]`, `[1, 3]`, and `[3, 1]`.