Accolite based Coding Questions

2026 batch By – Mr. Durgesh

StudyHub

Q1. Longest Subarray with Equal 0s and 1s

Statement:
Given a binary array, find the length of the longest subarray with equal number of 0s and 1s.

Input:

```
N
array elements
```

Output:

```
Length of longest subarray
```

Example:
Input

```
6
0 1 0 0 1 1
```

Output

```
6
```

Explanation:
Whole array has 3 zeros and 3 ones → longest length = 6.

Q2. Minimum Window Substring

Statement:
Given strings `S` and `T`, find the minimum window in `S` which contains all characters of `T`.

Input:

```
S
T
```

Output:

```
Minimum window substring
```

Example:
Input

```
ADOBECODEBANC
ABC
```

Output

```
BANC
```

Explanation:
"BANC" is the smallest substring containing A, B, and C.

Q3. Trapping Rain Water

Statement:
Given heights of bars, calculate total water trapped after raining.

Input:

```
N
height array
```

Output:

```
Total trapped water
```

Example:
Input

```
6
3 0 0 2 0 4
```

Output

```
10
```

Explanation:
Water is trapped between bars using left-max and right-max logic.

Q4. Median of Two Sorted Arrays

Statement:
Find median of two sorted arrays without merging them.

Input:

```
Array1
Array2
```

Output:

```
Median value
```

Example:
Input

```
1 3
2
```

Output

```
2
```

Explanation:
Median of merged array [1,2,3] = 2.

Q5. Longest Palindromic Substring

Statement:
Find the longest palindromic substring in a given string.

Input:

```
String
```

Output:

```
Longest palindrome
```

Example:
Input

```
babad
```

Output

```
bab
```

Explanation:
"bab" and "aba" both valid, return any one.

Q6. Kth Largest Element in Stream

Statement:
Given a stream of integers, return kth largest element after each insertion.

Input:

```
k
stream elements
```

Output:

```
kth largest after each insertion
```

Example:
Input

```
3
4 5 8 2
```

Output

```
-1 -1 4 4
```

Explanation:
Less than k elements → output -1.

Q7. Detect Cycle in Directed Graph

Statement:
Check if a directed graph contains a cycle.

Input:

```
V E
Edges
```

Output:

```
YES / NO
```

Example:
Input

```
3 3
0 1
1 2
2 0
```

Output

```
YES
```

Explanation:
Cycle exists: 0 → 1 → 2 → 0.

Q8. Maximum Product Subarray

Statement:
Find the contiguous subarray with the maximum product.

Input:

```
N
array
```

Output:

```
Maximum product
```

Example:
Input

```
4
2 3 -2 4
```

Output

```
6
```

Explanation:
Subarray [2,3] gives maximum product.

Q9. Word Break Problem

Statement:
Check if string can be segmented into dictionary words.

Input:

```
String
Dictionary words
```

Output:

```
YES / NO
```

Example:
Input

```
leetcode
leet code
```

Output

```
YES
```

Explanation:
"leet" + "code" forms the string.

Q10. Merge Intervals

Statement:
Merge all overlapping intervals.

Input:

```
Intervals
```

Output:

```
Merged intervals
```

Example:
Input

```
[1,3] [2,6] [8,10]
```

Output

```
[1,6] [8,10]
```

Explanation:
[1,3] and [2,6] overlap → merge.

Q11. Sliding Window Maximum

Statement:
Find maximum of every subarray of size K.

Input:

```
N K
array
```

Output:

```
Max values
```

Example:
Input

```
8 3
1 3 -1 -3 5 3 6 7
```

Output

```
3 3 5 5 6 7
```

Explanation:
Use deque for O(n) solution.

Q12. Rotten Oranges

Statement:
Find minimum time to rot all oranges.

Input:

```
Grid
```

Output:

```
Time or -1
```

Example:
Output

```
4
```

Explanation:
Use BFS level-wise traversal.

Q13. Largest Rectangle in Histogram

Statement:
Find largest rectangular area in histogram.

Input:

```
N
heights
```

Output:

```
Max area
```

Explanation:
Use stack to track previous smaller elements.

Q14. Minimum Platforms Required

Statement:
Find minimum number of platforms required for trains.

Input:

```
Arrival times
Departure times
```

Output:

```
Minimum platforms
```

Explanation:
Sort arrivals and departures separately.

Q15. Alien Dictionary

Statement:
Find order of characters in alien language.

Input:

```
Dictionary words
```

Output:

`Character order`

Explanation:
Topological sorting on graph.

21. Maximum Subarray Sum with One Deletion

- **Problem:** Given an array of integers, find the maximum sum of a subarray where you can optionally delete at most one element.
- **Input:** `arr = [1, -2, 0, 3]`
- **Output:** `4`
- **Explanation:** Maximum sum is from subarray `[1, 0, 3]` by deleting `-2`.

22. Longest Increasing Path in Matrix

- **Problem:** Given an `m x n` matrix, return the length of the longest increasing path. You can move in 4 directions.
- **Input:** `matrix = [[9,9,4],[6,6,8],[2,1,1]]`
- **Output:** `4`
- **Explanation:** Longest path is `[1,2,6,9]`. Use DFS + memoization.

23. Word Ladder II

- **Problem:** Find all shortest transformation sequences from `beginWord` to `endWord` given a dictionary.
- **Input:** `begin="hit", end="cog", dict=["hot","dot","dog","lot","log","cog"]`

- **Output:** `[["hit","hot","dot","dog","cog"], ["hit","hot","lot","log","cog"]]`
- **Explanation:** BFS to find shortest paths; backtracking to generate sequences.

24. Trapping Rain Water II

- **Problem:** Given a 2D height map, calculate total trapped water after raining.
- **Input:** `heightMap = [[1,4,3],[3,2,5]]`
- **Output:** `2`
- **Explanation:** Use priority queue (min-heap) to track boundary heights and trap water.

25. Palindrome Pairs

- **Problem:** Given a list of unique words, find all pairs `(i,j)` such that concatenation forms a palindrome.
- **Input:** `words = ["abcd","dcba","lls","s","sssll"]`
- **Output:** `[[0,1],[1,0],[3,2],[2,4]]`
- **Explanation:** Use Trie or hashmap to check prefixes/suffixes efficiently.

26. Edit Distance

- **Problem:** Find the minimum number of operations (insert, delete, replace) to convert string `word1` to `word2`.
- **Input:** `word1="horse", word2="ros"`
- **Output:** `3`
- **Explanation:** Dynamic Programming using 2D DP table.

27. Count of Smaller Numbers After Self

- **Problem:** Given an array, return count of smaller numbers to the right for each element.
- **Input:** `nums = [5,2,6,1]`
- **Output:** `[2,1,1,0]`
- **Explanation:** Use Binary Indexed Tree or modified Merge Sort.

28. Largest Rectangle in Histogram

- **Problem:** Given heights of histogram bars, find largest rectangle area.
- **Input:** `heights = [2,1,5,6,2,3]`
- **Output:** `10`
- **Explanation:** Use stack to maintain increasing bars for O(n) solution.

29. Sliding Window Maximum

- **Problem:** Given an array and window size `k`, find max of each sliding window.
- **Input:** `nums = [1,3,-1,-3,5,3,6,7], k=3`
- **Output:** `[3,3,5,5,6,7]`
- **Explanation:** Use deque to maintain candidates for window maximum.

30. Serialize and Deserialize Binary Tree

- **Problem:** Implement functions to serialize a binary tree to string and deserialize back.
- **Input:** `root = [1,2,3,null,null,4,5]`
- **Output:** `Serialized: "1,2,null,null,3,4,null,null,5,null,null"`
- **Explanation:** Use preorder traversal + marker for null nodes.

31. Shortest Subarray with Sum at Least K

- **Problem:** Find the length of shortest contiguous subarray with sum ≥ K.
- **Input:** `arr = [1,2,3,4,5], K=11`
- **Output:** `3`
- **Explanation:** Use prefix sum + deque to optimize search.

32. Russian Doll Envelopes

- **Problem:** Given envelopes `[w,h]`, find max number of envelopes you can nest.
- **Input:** `envelopes = [[5,4],[6,4],[6,7],[2,3]]`
- **Output:** `3`
- **Explanation:** Sort by width, then find LIS on heights.

33. Max Points on a Line

- **Problem:** Given points in 2D plane, find max points that lie on same line.
- **Input:** `points = [[1,1],[2,2],[3,3]]`
- **Output:** `3`
- **Explanation:** Use slope hashmap; handle duplicate points carefully.

34. Word Search II

- **Problem:** Find all words from a given list in a board.
- **Input:** `board = [["o","a","a","n"],["e","t","a","e"],["i","h","k","r"],["i","f","l","v"]], words = ["oath","pea","eat","rain"]`
- **Output:** `["eat","oath"]`

- **Explanation:** Use Trie + DFS from each cell for optimization.

35. Count of Range Sum

- **Problem:** Count the number of range sums within `[lower, upper]` in array.
- **Input:** `nums = [-2,5,-1], lower=-2, upper=2`
- **Output:** `3`
- **Explanation:** Use prefix sums + modified merge sort.

36. Palindrome Partitioning II

- **Problem:** Minimum cuts needed to partition string such that every substring is palindrome.
- **Input:** `s = "aab"`
- **Output:** `1`
- **Explanation:** DP solution: check palindrome substrings + min cuts.

37. Minimum Window Substring

- **Problem:** Given strings `s` and `t`, find min window in `s` containing all chars of `t`.
- **Input:** `s="ADOBECODEBANC", t="ABC"`
- **Output:** `"BANC"`
- **Explanation:** Sliding window + hashmap to track counts.

38. Maximum Sum Rectangle in 2D Matrix

- **Problem:** Find max sum rectangle in 2D array.
- **Input:** `matrix=[[1,2,-1,-4,-20],[…]]`
- **Output:** …

- **Explanation:** Reduce 2D to 1D array + Kadane's algorithm.

39. Alien Dictionary

- **Problem:** Given sorted dictionary of alien language, find order of characters.
- **Input:** `words=["wrt","wrf","er","ett","rftt"]`
- **Output:** `"wertf"`
- **Explanation:** Build graph + topological sort.

40. Burst Balloons

- **Problem:** Given array of balloons with points, maximize coins by bursting in optimal order.
- **Input:** `nums=[3,1,5,8]`
- **Output:** `167`
- **Explanation:** Use DP with boundaries to solve overlapping subproblems.

41. Maximum Product Subarray

- **Problem:** Find the contiguous subarray within an array which has the largest product.
- **Input:** `nums = [2,3,-2,4]`
- **Output:** `6`

- **Explanation:** Subarray `[2,3]` gives maximum product. Track max and min products at each step due to negative numbers.

42. Sliding Window Median

- **Problem:** Find median of each sliding window of size `k` in an array.
- **Input:** `nums = [1,3,-1,-3,5,3,6,7], k=3`
- **Output:** `[1,-1,-1,3,5,6]`
- **Explanation:** Use two heaps (max-heap and min-heap) to maintain lower and upper halves of window.

43. Minimum Path Sum

- **Problem:** Given a `m x n` grid filled with non-negative numbers, find a path from top-left to bottom-right which minimizes sum of all numbers along the path.
- **Input:** `grid = [[1,3,1],[1,5,1],[4,2,1]]`
- **Output:** `7`
- **Explanation:** Use DP: `dp[i][j] = grid[i][j] + min(dp[i-1][j], dp[i][j-1])`.

44. Decode Ways

- **Problem:** A message containing digits can be decoded as letters (A=1,…Z=26). Count possible decodings.
- **Input:** `s = "226"`
- **Output:** `3`
- **Explanation:** `"226"` → `"BZ"`, `"VF"`, `"BBF"`. Use DP to store ways up to each position.

45. Regular Expression Matching

- **Problem:** Implement regex matching with `'.'` and `'*'`.

- **Input:** `s="aab", p="c*a*b"`
- **Output:** `True`
- **Explanation:** DP table to track matches between substrings.

46. Longest Consecutive Sequence

- **Problem:** Find length of longest consecutive elements sequence in an unsorted array.
- **Input:** `nums = [100,4,200,1,3,2]`
- **Output:** `4`
- **Explanation:** Sequence `[1,2,3,4]`. Use HashSet for O(n) solution.

47. Count of Distinct Islands

- **Problem:** Count distinct islands in a 2D grid. Two islands are distinct if their shapes differ.
- **Input:** `grid = [[1,1,0,0],[1,0,0,0],[0,0,1,1]]`
- **Output:** `2`
- **Explanation:** DFS to record shape of each island; use set to count distinct shapes.

48. Maximum XOR of Two Numbers in an Array

- **Problem:** Find maximum XOR of two numbers in an array.
- **Input:** `nums = [3,10,5,25,2,8]`
- **Output:** `28`
- **Explanation:** Use Trie of binary representations to compute max XOR efficiently.

49. Wildcard Matching

- **Problem:** Implement `?` and `*` wildcard matching for strings.
- **Input:** `s="aa", p="*"`
- **Output:** `True`
- **Explanation:** Use DP to handle matching scenarios.

50. Reconstruct Itinerary

- **Problem:** Given tickets `[from, to]`, reconstruct itinerary in lexical order starting from `"JFK"`.
- **Input:** `[["MUC","LHR"],["JFK","MUC"],["SFO","SJC"],["LHR","SFO"]]`
- **Output:** `["JFK","MUC","LHR","SFO","SJC"]`
- **Explanation:** Hierholzer's algorithm for Eulerian path.

51. Minimum Window Subsequence

- **Problem:** Find shortest substring of `S` containing all characters of `T` in order.
- **Input:** `S="abcdebdde", T="bde"`
- **Output:** `"bcde"`
- **Explanation:** Use two pointers + DP to track subsequences.

52. Largest Divisible Subset

- **Problem:** Find largest subset of integers where every pair `(Si, Sj)` satisfies `Si % Sj == 0` or `Sj % Si == 0`.
- **Input:** `nums = [1,2,3]`
- **Output:** `[1,2]` or `[1,3]`
- **Explanation:** Sort + DP to build divisible chains.

53. Maximum Gap

- **Problem:** Find maximum difference between successive elements in sorted form, in linear time.
- **Input:** `nums = [3,6,9,1]`
- **Output:** `3`
- **Explanation:** Use bucket sort / pigeonhole principle for O(n) solution.

54. Kth Largest Element in Array

- **Problem:** Find kth largest element in an unsorted array.
- **Input:** `nums = [3,2,1,5,6,4], k=2`
- **Output:** `5`
- **Explanation:** Use Quickselect or min-heap.

55. Palindromic Substrings

- **Problem:** Count all palindromic substrings in a string.
- **Input:** `s = "aaa"`
- **Output:** `6`
- **Explanation:** Substrings: `"a","a","a","aa","aa","aaa"`. Use DP or expand-around-center.

56. Cheapest Flights Within K Stops

- **Problem:** Find cheapest flight from `src` to `dst` with ≤ K stops.
- **Input:** `n=3, flights=[[0,1,100],[1,2,100],[0,2,500]], src=0, dst=2, K=1`
- **Output:** `200`
- **Explanation:** Use BFS/DP to track min cost for each stop.

57. Sliding Window Maximum Sum of K Elements

- **Problem:** Find maximum sum of each subarray of size `k`.
- **Input:** `nums=[1,2,3,4,5], k=3`
- **Output:** `[6,9,12]`
- **Explanation:** Use prefix sums or sliding window sum.

58. Remove Invalid Parentheses

- **Problem:** Remove minimum parentheses to make string valid and return all possible results.
- **Input:** `s = "()())()"`
- **Output:** `["()()()","(())()"]`
- **Explanation:** BFS to generate all valid strings level by level.

59. Super Egg Drop

- **Problem:** Given `K` eggs and `N` floors, find minimum attempts to determine critical floor.
- **Input:** `K=2, N=6`
- **Output:** `3`
- **Explanation:** DP with states `[eggs][floors]` using optimal drop formula.

60. Word Break II

- **Problem:** Given a string and dictionary, return all possible sentences formed by splitting string into dictionary words.
- **Input:** `s="catsanddog", dict=["cat","cats","and","sand","dog"]`
- **Output:** `["cats and dog","cat sand dog"]`
- **Explanation:** DFS + memoization to build all combinations.

61. Sliding Window Maximum Product

- **Problem:** Find the maximum product of elements in each sliding window of size `k`.
- **Input:** `nums = [2,3,-2,4], k=2`
- **Output:** `[6,-6,-8]`
- **Explanation:** Track max and min products in each window to handle negatives.

62. Number of Islands II

- **Problem:** Given a grid and a sequence of land additions, count number of islands after each addition.
- **Input:** `m=3, n=3, positions=[[0,0],[0,1],[1,2]]`
- **Output:** `[1,1,2]`
- **Explanation:** Use Union-Find to efficiently merge adjacent lands.

63. K Closest Points to Origin

- **Problem:** Find `k` points closest to origin in 2D plane.
- **Input:** `points=[[1,3],[-2,2]], K=1`
- **Output:** `[[-2,2]]`
- **Explanation:** Use heap or quickselect based on Euclidean distance.

64. Shortest Palindrome

- **Problem:** Add characters in front of string to make it palindrome with minimal additions.
- **Input:** `s = "aacecaaa"`
- **Output:** `"aaacecaaa"`
- **Explanation:** Use KMP preprocessing on reversed string to find overlap.

65. Trapping Rain Water

- **Problem:** Calculate how much water can be trapped after raining given bar heights.
- **Input:** `height=[0,1,0,2,1,0,1,3,2,1,2,1]`
- **Output:** `6`
- **Explanation:** Use two pointers to track left and right max boundaries.

66. Longest Repeating Character Replacement

- **Problem:** Replace at most `k` characters to get longest repeating substring.
- **Input:** `s="AABABBA", k=1`
- **Output:** `4`
- **Explanation:** Use sliding window and frequency count.

67. Word Ladder I

- **Problem:** Find length of shortest transformation sequence from `beginWord` to `endWord`.
- **Input:** `begin="hit", end="cog", dict=["hot","dot","dog","lot","log","cog"]`
- **Output:** `5`
- **Explanation:** BFS to find minimum steps.

68. Minimum Path Sum in Triangle

- **Problem:** Given triangle array, find minimum path sum from top to bottom.
- **Input:** `triangle=[[2],[3,4],[6,5,7],[4,1,8,3]]`
- **Output:** `11`
- **Explanation:** DP from bottom to top.

69. Burst Balloons II

- **Problem:** Max coins obtained by bursting balloons in optimal order (variation of previous problem).
- **Input:** `nums=[3,1,5,8]`
- **Output:** `167`
- **Explanation:** DP with boundary handling.

70. Maximum Sum of 3 Non-Overlapping Subarrays

- **Problem:** Find 3 non-overlapping subarrays of length `k` with max total sum.
- **Input:** `nums=[1,2,1,2,6,7,5,1], k=2`
- **Output:** `[0,3,5]`
- **Explanation:** Prefix sums + dynamic programming.

71. Find Median from Data Stream

- **Problem:** Continuously find median from data stream.
- **Input:** `addNum(1), addNum(2), findMedian(), addNum(3), findMedian()`
- **Output:** `1.5, 2`
- **Explanation:** Use two heaps (max-heap & min-heap).

72. Decode String

- **Problem:** Decode strings like `"3[a2[c]]"` → `"accaccacc"`.

- **Input:** `s="3[a2[c]]"`
- **Output:** `"accaccacc"`
- **Explanation:** Use stack to handle nested patterns.

73. Super Ugly Number

- **Problem:** Find nth super ugly number given prime factors.
- **Input:** `n=12, primes=[2,7,13,19]`
- **Output:** `32`
- **Explanation:** Use DP with multiple pointers.

74. Count of Range Sum II

- **Problem:** Count subarrays with sum in `[lower, upper]`.
- **Input:** `nums=[-2,5,-1], lower=-2, upper=2`
- **Output:** `3`
- **Explanation:** Prefix sums + merge sort optimization.

75. Longest Valid Parentheses

- **Problem:** Find length of longest valid parentheses substring.
- **Input:** `s="(()"`
- **Output:** `2`
- **Explanation:** Use stack or DP to track valid sequences.

76. Trapping Rain Water III (2D)

- **Problem:** Given 2D heights map, find trapped water after raining.
- **Input:** `heightMap=[[1,4,3],[3,2,5]]`
- **Output:** `2`
- **Explanation:** Min-heap boundary BFS.

77. Sliding Window Maximum Average

- **Problem:** Find max average of subarray of length `k`.
- **Input:** `nums=[1,12,-5,-6,50,3], k=4`
- **Output:** `12.75`
- **Explanation:** Sliding window sum.

78. Shortest Distance from All Buildings

- **Problem:** Find shortest distance from empty land to all buildings in grid.
- **Input:** `grid=[[1,0,2,0,1],[0,0,0,0,0],[0,0,1,0,0]]`
- **Output:** `7`
- **Explanation:** BFS from each building; accumulate distances.

79. Find K Pairs with Smallest Sums

- **Problem:** Return k pairs with smallest sums from two sorted arrays.
- **Input:** `nums1=[1,7,11], nums2=[2,4,6], k=3`
- **Output:** `[[1,2],[1,4],[1,6]]`
- **Explanation:** Use min-heap to maintain next smallest pair.

80. Range Module

- **Problem:** Implement range module to add/remove/query intervals.
- **Input:** `addRange(10,20), removeRange(14,16), queryRange(10,14)`
- **Output:** `True`

- **Explanation:** Use segment tree or balanced BST to manage ranges.

81. Count of Distinct Subsequences

- **Problem:** Count distinct subsequences of string `S`.
- **Input:** `S="rabbbit"`
- **Output:** `3`
- **Explanation:** Use DP to count subsequences forming target string.

82. Longest Increasing Subsequence II

- **Problem:** Find length of LIS in array of integers.
- **Input:** `nums=[10,9,2,5,3,7,101,18]`
- **Output:** `4`
- **Explanation:** Use DP or binary search approach for O(n log n).

83. Remove K Digits

- **Problem:** Remove `k` digits from number to make it smallest.
- **Input:** `num="1432219", k=3`
- **Output:** `"1219"`
- **Explanation:** Use stack to remove digits greedily.

84. Best Time to Buy and Sell Stock III

- **Problem:** Max profit with at most two transactions.
- **Input:** `prices=[3,3,5,0,0,3,1,4]`
- **Output:** `6`
- **Explanation:** DP with 2 transaction states.

85. Minimum Cost to Merge Stones

- **Problem:** Merge stones into one pile with minimum cost.
- **Input:** `stones=[3,2,4,1], K=2`
- **Output:** `20`
- **Explanation:** DP with intervals.

86. Maximum Sum of Rectangle No Larger Than K

- **Problem:** Find max sum rectangle in 2D matrix ≤ k.
- **Input:** `matrix=[[1,0,1],[0,-2,3]], k=2`
- **Output:** `2`
- **Explanation:** Reduce 2D problem to 1D and use TreeSet for prefix sums.

87. Sliding Window Median II

- **Problem:** Find median after adding each element from stream.
- **Input:** `nums=[1,4,2,3]`
- **Output:** `[1,2.5,2,2.5]`
- **Explanation:** Maintain two heaps for lower and upper halves.

88. Number of Ways to Paint N × 3 Grid

- **Problem:** Count ways to paint grid with no adjacent same colors.
- **Input:** `n=1`
- **Output:** `12`

- **Explanation:** Use DP with color pattern states.

89. Shortest Bridge

- **Problem:** Flip 0s to connect two islands with minimum flips.
- **Input:** `grid=[[0,1],[1,0]]`
- **Output:** `1`
- **Explanation:** BFS from one island to reach the other.

90. Minimum Number of Refueling Stops

- **Problem:** Minimum stops to reach target with fuel stations.
- **Input:** `target=100, startFuel=10, stations=[[10,60],[20,30],[30,30],[60,40]]`
- **Output:** `2`
- **Explanation:** Max-heap to refuel at optimal stations.

91. Number of Valid Words for Each Puzzle

- **Problem:** Count valid words in a list that match each puzzle.
- **Input:** `words=["apple","pleas","please"], puzzles=["aelwxyz","aelpxyz"]`
- **Output:** `[0,1]`
- **Explanation:** Bitmask to efficiently check word containment.

92. Maximum Length of Repeated Subarray

- **Problem:** Find length of longest repeated subarray between two arrays.
- **Input:** `A=[1,2,3,2,1], B=[3,2,1,4,7]`

- **Output:** `3`
- **Explanation:** Use DP or rolling hash.

93. Shortest Superstring

- **Problem:** Find shortest string containing all given strings as substring.
- **Input:** `words=["alex","loves","leetcode"]`
- **Output:** `"alexlovesleetcode"`
- **Explanation:** DP with bitmask to track used words.

94. Maximum Length of Pair Chain

- **Problem:** Find longest chain of pairs `[a,b]` where `b<c` for next pair `[c,d]`.
- **Input:** `pairs=[[1,2],[2,3],[3,4]]`
- **Output:** `2`
- **Explanation:** Sort by second element and apply greedy selection.

95. Count of Range Sum III

- **Problem:** Count subarrays sum within `[lower, upper]`.
- **Input:** `nums=[-2,5,-1], lower=-2, upper=2`
- **Output:** `3`
- **Explanation:** Prefix sums + merge sort optimization.

96. Number of Ways to Split Array

- **Problem:** Count ways to split array into 3 contiguous subarrays with equal sum.
- **Input:** `nums=[0,2,1,0]`
- **Output:** `0`
- **Explanation:** Prefix sum to identify splits.

97. Palindrome Partitioning III

- **Problem:** Split string into `k` substrings with minimum changes to make all palindromes.
- **Input:** `s="abc", k=2`
- **Output:** `1`
- **Explanation:** DP on substring + partition.

98. Max Sum of Rectangle with No Larger Than K II

- **Problem:** 2D matrix max sum ≤ k.
- **Input:** `matrix=[[2,2,-1]], k=3`
- **Output:** `3`
- **Explanation:** Use 1D reduction + BST prefix sum.

99. Minimum Cost to Cut Stick

- **Problem:** Given stick length and cut positions, find min cost to cut stick.
- **Input:** `length=7, cuts=[1,3,4,5]`
- **Output:** `16`
- **Explanation:** DP with interval splitting.

100. Cheapest Path in Weighted Grid

- **Problem:** Find min cost path from top-left to bottom-right with weighted grid.
- **Input:** `grid=[[1,3,1],[1,5,1],[4,2,1]]`
- **Output:** `7`
- **Explanation:** DP or Dijkstra to find min path sum.