Wipro Technical / Computer Fundamentals Real Company Questions By – Mr. Durgesh StudyHub

1. Reverse an array without using extra space.

Input: [1,2,3,4,5]
Output: [5,4,3,2,1]
Concept: Two-pointer swap in-place.

---

2. Find the largest subarray with sum 0.

Input: [1,2,-3,3,1]
Output: [1,2,-3] or [-3,3]
Concept: Use prefix sum + hashmap.

---

3. Count all substrings of a binary string that start and end with 1.

Input: "10101"
Output: 4
Concept: Count number of 1s = n → result = n*(n-1)/2

---

4. Rotate array by k steps to the right.

Input: [1,2,3,4,5], k=2
Output: [4,5,1,2,3]
Concept: Reverse whole array, reverse first k, reverse remaining.

---

5. Detect a loop in a linked list.

Concept: Use **Floyd's cycle detection (slow + fast pointers)**

---

6. Merge two sorted linked lists.

Concept: Maintain dummy head, merge nodes by comparing values.

---

7. Find middle element of linked list in one pass.

Concept: Use **slow and fast pointer**

---

8. Level order traversal of binary tree.

Concept: Use **queue**.

---

9. Check if a binary tree is height-balanced.

Concept: Recursively check left/right height difference ≤1

---

10. Find shortest path in unweighted graph.

Concept: Use **BFS** starting from source node

---

11. Fibonacci using DP.

Input: n = 6
Output: 0,1,1,2,3,5
Concept: Bottom-up tabulation or memoization

---

12. Minimum path sum in a 2D grid.

Input: grid = [[1,3,1],[1,5,1],[4,2,1]]
Output: 7 (1→3→1→1→1)
Concept: DP: dp[i][j]=grid[i][j]+min(dp[i-1][j], dp[i][j-1])

---

13. Subset sum problem: Check if a sum exists.

Input: arr=[3,34,4,12,5,2], sum=9
Output: True
Concept: DP: dp[i][j]=dp[i-1][j] || dp[i-1][j-arr[i]]

---

14. Implement "Two Sum" problem.

Input: nums=[2,7,11,15], target=9
Output: [0,1]
Concept: Use HashMap to store seen values

---

15. Reverse words in a string.

Input: "Hello World"
Output: "World Hello"
Concept: Split string, reverse array, join

---

16. Count distinct elements in an array.

Input: [1,2,2,3,4,1]
Output: 4
Concept: Use **set**

17. Stack using Queues.

Concept: Push costly or pop costly approach

18. Implement LRU cache.

Concept: Use **HashMap** + **Doubly Linked List**

19. Max product of two numbers in array.

Input: [3,5,1,7,9]
Output: 63 (9*7)
Concept: Find largest and second largest element

20. Remove duplicates from a sorted linked list.

Concept: Traverse and remove consecutive duplicates

21. Check if a string is palindrome.

Input: "racecar"
Output: True
Concept: Two-pointer check from both ends

22. Find all prime numbers less than n.

Input: n=10
Output: [2,3,5,7]
Concept: Sieve of Eratosthenes

23. Find duplicate elements in an array.

Input: [1,2,3,2,4,3]
Output: [2,3]
Concept: Use HashMap or count array

24. Longest increasing subsequence.

Input: [10,9,2,5,3,7,101,18]
Output: 4 ([2,3,7,101])
Concept: DP: dp[i]=max(dp[j])+1 if arr[i]>arr[j]

25. Move all zeros to the end.

Input: [0,1,0,3,12]
Output: [1,3,12,0,0]
Concept: Two-pointer swap or overwrite

26. Find the first missing positive integer in an unsorted array.

Input: [3,4,-1,1]
Output: 2
Concept: Place each number at its index (index = number-1) and find the first mismatch.

27. Median of two sorted arrays of same size.

Input: arr1=[1,12,15,26,38], arr2=[2,13,17,30,45]
Output: 16
Concept: Merge method or binary search on smaller array.

28. Find maximum subarray sum (Kadane's Algorithm).

Input: [-2,1,-3,4,-1,2,1,-5,4]
Output: 6 ([4,-1,2,1])
Concept: Maintain max_ending_here and max_so_far

---

29. Serialize and deserialize a binary tree.

Concept: Use preorder traversal + NULL markers
Note: Wipro sometimes asks implementation logic in interviews.

---

30. Count number of ways to reach nth stair (1 or 2 steps).

Input: n=4
Output: 5
Concept: DP / Fibonacci sequence

---

31. Find length of longest substring without repeating characters.

Input: "abcabcbb"
Output: 3 ("abc")
Concept: Sliding window + HashSet

---

32. Minimum number of coins to make a sum.

Input: coins=[1,2,5], amount=11
Output: 3 (5+5+1)
Concept: DP: dp[i]=min(dp[i], dp[i-coin]+1)

---

33. Maximum sum rectangle in a 2D matrix.

Input: [[1,2,-1,-4,-20],[-8,-3,4,2,1],[3,8,10,1,3],[-4,-1,1,7,-6]]
Output: 29
Concept: Kadane's Algorithm on 2D array

34. Trapping Rain Water problem.

Input: [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Concept: Precompute left_max and right_max arrays

35. Check if a graph is bipartite.

Concept: Use BFS coloring or DFS coloring

36. Implement a Priority Queue from scratch.

Concept: Use **Heap (MinHeap/MaxHeap)**

37. Rotate a matrix 90° clockwise in-place.

Input: [[1,2,3],[4,5,6],[7,8,9]]
Output: [[7,4,1],[8,5,2],[9,6,3]]
Concept: Transpose + reverse rows

38. Word Break Problem.

Input: s="leetcode", dict=["leet","code"]
Output: True
Concept: DP: dp[i]=True if s[0..i] can be segmented

39. Longest common subsequence of two strings.

Input: "AGGTAB", "GXTXAYB"
Output: 4 ("GTAB")
Concept: DP: dp[i][j] = dp[i-1][j-1]+1 if chars match else max(dp[i-1][j], dp[i][j-1])

40. Count ways to reach a target score using given moves.

Input: moves=[3,5,10], target=20
Output: Number of combinations
Concept: DP: coin change style

41. Find maximum sum path in two arrays (intersection allowed).

Concept: Use two pointers + sum at intersections

42. Serialize a graph.

Concept: Adjacency list to string or JSON format

43. Implement Trie for a dictionary.

Concept: Insert/search words efficiently

44. Design a HashMap from scratch.

Concept: Array of linked lists + hash function

45. Median of a data stream.

Concept: Use **two heaps** (maxHeap for lower half, minHeap for upper half)

46. Sliding Window Maximum.

Input: nums=[1,3,-1,-3,5,3,6,7], k=3
Output: [3,3,5,5,6,7]
Concept: Deque to store indices of maximums

---

47. Count inversions in an array.

Input: [2,4,1,3,5]
Output: 3
Concept: Modified merge sort

---

48. Check if a number is power of 2.

Input: 16
Output: True
Concept: n>0 and (n & (n-1))==0

---

49. Find the celebrity in a party (knows/no knows).

Concept: Matrix logic or two-pointer elimination

---

50. Minimum number of platforms required for trains.

Input: arrival=[9:00,9:40], departure=[9:10,12:00]
Output: 1
Concept: Sort arrival & departure, use two pointers

51. Implement Min Stack (support push, pop, top, getMin in O(1)).

Concept: Maintain another stack to track minimum values.

52. Find next greater element for each element in an array.

Input: [4,5,2,25]
Output: [5,25,25,-1]
Concept: Use stack to keep track of next greater element.

53. Find all permutations of a string.

Input: "ABC"
Output: ["ABC","ACB","BAC","BCA","CAB","CBA"]
Concept: Backtracking recursion.

54. Implement Kth largest element in an array.

Input: [3,2,1,5,6,4], k=2
Output: 5
Concept: Use MinHeap of size k or QuickSelect.

55. Detect cycle in a directed graph.

Concept: Use DFS + recursion stack.

56. Find all pairs with given sum in an array.

Input: arr=[1,4,2,3,5], sum=5
Output: [(1,4),(2,3)]
Concept: HashMap to track complements.

57. Implement Merge Sort and explain time complexity.

Concept: Divide & conquer, O(n log n) time, O(n) space.

58. Minimum swaps to sort an array.

Input: [4,3,2,1]
Output: 2
Concept: Cycle detection in permutation array.

59. Maximum length chain of pairs.

Input: [(5,24),(15,25),(27,40),(50,60)]
Output: 3
Concept: Sort by second element → DP/Greedy.

60. Longest Palindromic Substring.

Input: "babad"
Output: "bab" or "aba"
Concept: Expand around center or DP.

61. Word Ladder problem.

Input: begin="hit", end="cog", dict=["hot","dot","dog","lot","log"]
Output: 5 ("hit"→"hot"→"dot"→"dog"→"cog")
Concept: BFS shortest path.

62. Find number of islands in a 2D grid.

Input: [[1,1,0,0],[1,1,0,0],[0,0,1,0],[0,0,0,1]]
Output: 3
Concept: DFS/BFS to mark visited land.

63. Implement Queue using two stacks.

Concept: Push costly / Pop costly approach.

64. Largest rectangle in histogram.

Input: [2,1,5,6,2,3]
Output: 10
Concept: Use stack to track bars and compute areas.

65. Sliding Window Median.

Input: nums=[1,3,-1,-3,5,3,6,7], k=3
Output: [1,-1,-1,3,5,6]
Concept: Two heaps to maintain lower and upper halves.

66. Implement AVL tree insertion.

Concept: Maintain balance factor −1,0,1; perform rotations.

67. Maximum sum increasing subsequence.

Input: [1,101,2,3,100,4,5]
Output: 106 ([1,2,3,100])
Concept: DP: dp[i] = max(dp[j]+arr[i]) if arr[i]>arr[j]

68. Count number of BSTs with n nodes.

Input: n=3
Output: 5
Concept: Catalan number: Cn = (2n)!/(n!(n+1)!)

69. Implement Graph DFS & BFS traversal.

Concept: Use recursion/stack for DFS, queue for BFS.

70. Minimum cost to reach last cell in matrix.

Input: matrix=[[1,2,3],[4,8,2],[1,5,3]]
Output: 8 (1→2→2→3)
Concept: DP: dp[i][j]=matrix[i][j]+min(dp[i-1][j], dp[i][j-1])

71. Job Scheduling problem (maximize profit).

Input: jobs=[{id:1,deadline:2,profit:100},{id:2,deadline:1,profit:19},...]
Concept: Sort by profit descending, assign latest free slot.

72. Count total ways to decode a numeric string (like "226").

Concept: DP: dp[i] = dp[i-1] + dp[i-2] if valid two-digit number

73. Serialize and deserialize a DAG.

Concept: Use adjacency list representation with JSON or string format.

74. Maximum sum path in triangle.

Input: triangle=[[2],[3,4],[6,5,7],[4,1,8,3]]
Output: 2+4+7+8=21
Concept: Bottom-up DP.

75. Implement Topological Sort.

Concept: Use DFS post-order or Kahn's algorithm with queue.

76. Find k smallest elements in an array.

Input: [7,10,4,3,20,15], k=3
Output: [3,4,7]
Concept: Use MinHeap of size n or MaxHeap of size k for efficiency.

77. Median of a sliding window.

Input: nums=[1,3,-1,-3,5,3,6,7], k=3
Output: [1,-1,-1,3,5,6]
Concept: Two heaps to maintain lower and upper halves.

78. Kth smallest element in a BST.

Concept: Inorder traversal of BST → sorted order → pick kth element.

79. Implement Dijkstra's shortest path algorithm.

Concept: Use MinHeap (priority queue) and adjacency list/matrix.

80. Implement Bellman-Ford algorithm.

Concept: Handles negative weights, relax edges repeatedly.

81. Detect cycle in undirected graph using Union-Find.

Concept: Use parent array and union by rank.

82. Check if a string matches regex pattern (simplified).

Input: s="aab", pattern="c*a*b"
Output: True
Concept: DP-based regex matching.

---

83. Find maximum product subarray.

Input: [2,3,-2,4]
Output: 6
Concept: Maintain maxProd and minProd at each step.

---

84. Implement Trie with prefix search.

Concept: Each node stores children + end-of-word flag.

---

85. Find minimum window substring containing all characters of another string.

Input: s="ADOBECODEBANC", t="ABC"
Output: "BANC"
Concept: Sliding window + character count hashmap.

---

86. Implement Union-Find / Disjoint Set Union with path compression.

Concept: Efficient for connectivity problems, O(α(n)) amortized.

---

87. Count number of unique BSTs with n nodes.

Input: n=3
Output: 5
Concept: Catalan numbers.

88. Minimum steps to convert string A to B (Edit Distance).

Input: "horse", "ros"
Output: 3
Concept: DP: insertion, deletion, replacement cost.

89. Word Ladder II — all shortest transformation sequences.

Input: begin="hit", end="cog", dict=["hot","dot","dog","lot","log"]
Concept: BFS + backtracking.

90. Maximum rectangle of 1s in a binary matrix.

Input: [[0,1,1,0],[1,1,1,1],[1,1,1,0]]
Output: 6
Concept: Apply Largest Rectangle in Histogram row by row.

91. Implement KMP string matching algorithm.

Concept: Precompute LPS array, then pattern matching in O(n) time.

92. Find shortest superstring from given strings.

Concept: DP + bitmask or greedy approximation.

93. Find maximum XOR of two numbers in array.

Input: [3,10,5,25,2,8]
Output: 28
Concept: Use Trie of binary representations.

94. Longest Palindromic Subsequence.

Input: "bbbab"
Output: 4 ("bbbb")
Concept: DP: dp[i][j]=dp[i+1][j-1]+2 if ends match else max(dp[i+1][j], dp[i][j-1])

---

95. Implement LFU Cache.

Concept: HashMap + Doubly Linked List + frequency count.

---

96. Minimum cost to connect ropes.

Input: [4,3,2,6]
Output: 29
Concept: MinHeap: always combine two smallest ropes.

---

97. Find maximum sum of non-adjacent elements.

Input: [3,2,5,10,7]
Output: 15 (3+5+7)
Concept: DP: incl/excl pattern.

---

98. Implement Floyd-Warshall algorithm for all-pairs shortest path.

Concept: DP over adjacency matrix.

---

99. Serialize and deserialize a N-ary tree.

Concept: Use preorder + children count or JSON style representation.

---

100. Minimum cost path with obstacles in a grid.

Input: -1 indicates obstacle
Concept: DP with obstacle check: dp[i][j]=min(dp[i-1][j], dp[i][j-1])+grid[i][j]

101. Implement Rotten Oranges problem (minimum time to rot all oranges).

Input: 2D grid of 0(empty)/1(fresh)/2(rotten)
Output: Minimum time to rot all oranges or -1 if impossible
Concept: BFS from rotten oranges

---

102. Find maximum path sum in a binary tree (any node to any node).

Concept: Recursive DFS, maintain max sum at each node

---

103. Design Autocomplete System.

Concept: Trie + priority queue to suggest top-k completions

---

104. Implement LRU Cache with expiration time.

Concept: HashMap + Doubly Linked List + timestamp

---

105. Number of ways to paint a fence with k colors and n posts, no more than 2 adjacent same.

Concept: DP: same/ diff pattern

---

106. Count all palindromic substrings in a string.

Input: "aaa"
Output: 6
Concept: Expand around center

---

107. Find all unique triplets that sum to zero (3Sum).

Input: [-1,0,1,2,-1,-4]
Output: [[-1,-1,2],[-1,0,1]]
Concept: Sort + two pointers

---

108. Maximum sum submatrix of size k x k.

Concept: Prefix sum for $O(n^2)$ solution

---

109. Minimum insertions to make string palindrome.

Concept: n − length of longest palindromic subsequence

---

110. Shortest path in weighted grid with obstacles.

Concept: Dijkstra or BFS with priority queue

---

111. Implement Flood Fill algorithm.

Concept: DFS/BFS for changing connected region color

---

112. Count number of distinct subsequences of string s that equals t.

Concept: DP: dp[i][j] = dp[i-1][j-1]+dp[i-1][j]

---

113. Find maximum area of island in binary matrix.

Concept: DFS/BFS to calculate connected 1s

---

114. Implement Segment Tree for range sum query.

Concept: Build tree, query in O(log n), update in O(log n)

---

115. Implement Fenwick Tree (Binary Indexed Tree).

Concept: For prefix sums and updates in O(log n)

---

116. Count all paths from top-left to bottom-right in grid (with obstacles).

Concept: DP: dp[i][j] = dp[i-1][j]+dp[i][j-1]

---

117. Find largest rectangle containing only 1s in a binary matrix.

Concept: Treat each row as histogram, apply largest rectangle algorithm

118. Implement Trie with delete operation.

Concept: Recursive delete and prune empty nodes

119. Serialize and deserialize binary search tree efficiently.

Concept: Preorder traversal, exploit BST property

120. Minimum jumps to reach end of array.

Input: [2,3,1,1,4]
Output: 2 (2→3→end)
Concept: Greedy / DP

121. Find duplicate number in array 1…n (n+1 elements).

Input: [1,3,4,2,2]
Output: 2
Concept: Floyd’s cycle detection in array

122. Find subarray with given sum (positive numbers).

Input: [1,2,3,7,5], sum=12
Output: [2,3,7]
Concept: Sliding window

123. Minimum cost to reach last cell in matrix with diagonal moves allowed.

Concept: DP with min(dp[i-1][j], dp[i][j-1], dp[i-1][j-1])

124. Maximum sum of rectangle in matrix using Kadane's Algorithm.

Concept: Fix left & right columns, apply 1D max subarray on rows

---

125. Implement Word Search II (find all words from dictionary in board).

Concept: Trie + DFS

---

126. Maximum sum circular subarray.

Input: [5,-3,5]
Output: 10
Concept: Max(standard Kadane, total sum−min subarray sum)

---

127. Find minimum cost path in weighted DAG.

Concept: Topological sort + relax edges

---

128. Longest consecutive sequence in array.

Input: [100,4,200,1,3,2]
Output: 4 ([1,2,3,4])
Concept: HashSet O(n) solution

---

129. Count number of subarrays with sum divisible by k.

Concept: Prefix sum mod k + hashmap

---

130. Implement Max Stack (support push, pop, top, getMax in O(1)).

Concept: Stack + auxiliary stack for max

---

131. Find minimum window substring containing all distinct characters.

Concept: Sliding window + character frequency map

---

132. Find k closest elements to x in sorted array.

Concept: Binary search + two pointers

---

133. Find median of two sorted arrays (different sizes).

Concept: Binary search on smaller array, O(log(min(n,m)))

---

134. Maximum sum bitonic subsequence.

Concept: DP: LIS from left, LDS from right

---

135. Count distinct islands in a 2D grid.

Concept: DFS + normalize island shape

---

136. Minimum insertions/deletions to convert string A to B.

Concept: Edit distance, DP

137. Find minimum operations to make all array elements equal (allowed +1 or -1).

Concept: Median minimizes sum of absolute differences

138. Maximum sum path from leaf to leaf in binary tree.

Concept: Recursive DFS, maintain max sum globally

139. Implement Trie autocomplete with frequency (top-k suggestions).

Concept: Trie + priority queue

140. Count number of islands in 3D grid.

Concept: DFS/BFS in 3D space

141. Maximum product of increasing subsequence.

Concept: DP: track max product ending at i

142. Find shortest path in weighted graph with exactly k edges.

Concept: DP on vertices + edges

143. Implement Kahn's algorithm for topological sorting.

Concept: Queue + indegree array

---

144. Maximum sum of non-overlapping subarrays of size k.

Concept: Sliding window + prefix sum

---

145. Maximum profit buy/sell stock at most twice.

Concept: DP: track local & global max

---

146. Count number of binary strings without consecutive 1s of length n.

Concept: DP: f(n)=f(n-1)+f(n-2)

---

147. Find maximum sum rectangle in 2D array containing at least one positive number.

Concept: Kadane's 2D + check for all negative case

---

148. Longest substring with at most k distinct characters.

Concept: Sliding window + hashmap

---

149. Minimum number of refueling stops to reach destination.

Concept: Greedy + max heap for fuel stations

150. Implement Suffix Trie / Suffix Tree for string pattern matching.

Concept: Each path represents a suffix; efficient for substring queries

151. Implement Minimum Window Subsequence.

Input: s="abcdebdde", t="bde"
Output: "bcde"
Concept: Two pointers + DP

152. Count number of paths from source to destination in DAG.

Concept: DP on DAG using topological order

153. Find maximum sum path in a matrix from top-left to bottom-right.

Concept: DP: dp[i][j] = max(dp[i-1][j], dp[i][j-1])+matrix[i][j]

154. Maximum sum submatrix no larger than k.

Concept: Kadane + prefix sum + BST for ≤k

---

155. Implement Median Finder for data stream.

Concept: Two heaps (maxHeap for lower half, minHeap for upper half)

---

156. Maximum number of points on a line.

Concept: Use hashmap to track slopes

---

157. Find minimum number of swaps required to sort array.

Concept: Count cycles in permutation

---

158. Maximum sum of non-overlapping intervals.

Concept: Sort intervals by end → DP/Greedy

---

159. Find longest substring with at most k replacements to make all chars same.

Concept: Sliding window + hashmap

---

160. Minimum number of steps to make array non-decreasing.

Concept: Greedy / DP

---

161. Longest mountain in array.

Input: [2,1,4,7,3,2,5]
Output: 5 ([1,4,7,3,2])
Concept: Two-pass DP: left[i]=length of increasing, right[i]=length of decreasing

162. Maximum profit from stock transactions with cooldown.

Concept: DP: track buy/sell/cooldown state

163. Minimum cost to paint houses with k colors (no two adjacent same).

Concept: DP: dp[i][color]=min(dp[i-1][other colors]+cost[i][color])

164. Find length of longest subarray with at most k distinct integers.

Concept: Sliding window + hashmap

165. Maximum sum path in binary tree with alternating even/odd nodes.

Concept: Recursive DFS with parity check

166. Find shortest bridge to connect two islands in binary matrix.

Concept: BFS + DFS to mark one island, then expand

167. Implement Min Stack supporting push, pop, getMin, getSecondMin in O(1).

Concept: Stack + auxiliary stack for min and second min

168. Maximum number of envelopes Russian doll problem.

Concept: Sort width ascending, height descending → LIS on height

169. Maximum sum path in N-ary tree from leaf to leaf.

Concept: DFS recursion, maintain max globally

170. Find kth largest sum of contiguous subarray.

Concept: MinHeap of size k while traversing all subarrays

171. Minimum steps to convert one word to another (Word Ladder II).

Concept: BFS + backtracking for all shortest sequences

172. Maximum sum of subarray after at most one reversal.

Concept: Kadane + prefix/suffix sum

173. Maximum number of points you can earn in matrix picking non-adjacent rows.

Concept: DP: track max for previous row options

174. Find longest palindromic substring with at most k modifications.

Concept: Expand around center + track allowed changes

175. Implement Trie with wildcard search.

Concept: DFS on children nodes for '.' wildcard

176. Maximum profit from job scheduling with deadlines.

Concept: Sort by profit descending → assign latest available slot

177. Count number of paths with given sum in binary tree.

Concept: Prefix sum hashmap during DFS

178. Find minimum operations to make array elements equal with increment/decrement by 1/2.

Concept: Greedy / median-based approach

179. Maximum sum of k non-overlapping subarrays of size m.

Concept: Sliding window + DP

180. Maximum XOR path in a tree.

Concept: DFS + Trie of XOR prefixes

181. Count number of ways to tile a 3xn board with 2x1 dominos.

Concept: DP with states representing current row configuration

182. Maximum sum of weighted path in DAG.

Concept: Topological sort + relax edges

183. Find maximum sum rectangle in 2D array with at least one positive number.

Concept: Kadane 2D + check for all negative

184. Implement Suffix Array construction and substring search.

Concept: Sort all suffixes or use efficient algorithms O(n log n)

185. Maximum profit from stock with transaction fee.

Concept: DP: track cash & hold states

186. Count distinct subsequences modulo large prime.

Concept: DP: track count of each character

187. Maximum sum circular subarray with at most one deletion.

Concept: Kadane + prefix/suffix sum

---

188. Implement Interval Tree for overlapping interval queries.

Concept: BST storing intervals with max endpoint

---

189. Minimum number of swaps to make binary string alternate.

Concept: Count misplaced 0s and 1s, check parity

---

190. Maximum sum subsequence with no two elements adjacent in array.

Concept: DP: incl/excl pattern

---

191. Maximum number of envelopes nested (Russian doll) with duplicate widths.

Concept: Sort width ascending, height descending → LIS

---

192. Implement LFU cache with O(1) operations.

Concept: HashMap + frequency list

---

193. Find number of subarrays with sum equal to k.

Concept: Prefix sum + hashmap

194. Count number of distinct palindromic substrings.

Concept: DP or Palindromic Tree (Eertree)

195. Maximum sum of subsequence with alternating sign.

Concept: DP: max positive / negative sum ending at i

196. Minimum cost to reach end in a weighted grid with diagonal moves.

Concept: DP: min(dp[i-1][j], dp[i][j-1], dp[i-1][j-1])

197. Find longest arithmetic subsequence.

Concept: DP with hashmap storing difference → $O(n^2)$

198. Maximum sum submatrix with size constraint (at most k x k).

Concept: Prefix sum + iterate all possible windows

199. Count number of subarrays where product is less than k.

Concept: Sliding window, maintain product

200. Implement Suffix Automaton for fast substring queries.

Concept: Build state machine representing all substrings, O(n)