Top 100 Array Questions for FAANG Interviews

EASY LEVEL (1-30)

Basic Operations & Traversal

1. Find the largest element in an array
2. Find the smallest element in an array
3. Find the second largest element in an array
4. Calculate sum of all elements in an array
5. Find the average of array elements
6. Count number of even and odd elements
7. Reverse an array in-place
8. Check if array is sorted in ascending order
9. Find missing number in array of 1 to n
10. Remove duplicates from sorted array

Searching & Finding

11. Linear search in an array
12. Binary search in sorted array
13. Find first and last occurrence of element
14. Find element that appears once (others appear twice)
15. Find intersection of two arrays
16. Find union of two sorted arrays
17. Check if array is a subset of another array
18. Find common elements in three sorted arrays
19. Search in rotated sorted array
20. Find peak element in array

Basic Two Pointers

21. Two sum problem (find pair with given sum)
22. Three sum problem (find triplet with sum zero)
23. Remove element from array
24. Move zeros to end
25. Merge two sorted arrays

26. Check if array can be sorted by reversing subarray
27. Sort array of 0s, 1s, and 2s (Dutch flag)
28. Partition array around pivot
29. Find pair with given difference
30. Container with most water

MEDIUM LEVEL (31-70)

Sliding Window & Subarray

31. Maximum sum subarray (Kadane's algorithm)
32. Maximum sum of k consecutive elements
33. Longest subarray with sum k
34. Minimum window substring
35. Longest substring without repeating characters
36. Maximum product subarray
37. Subarray with given sum
38. Count subarrays with sum k
39. Longest increasing subarray
40. Maximum average subarray of length k

Advanced Two Pointers & Sorting

41. Four sum problem
42. 3Sum closest to target
43. Sort colors (0s, 1s, 2s)
44. Next permutation
45. Minimum number of swaps to sort array
46. Kth largest element in array
47. Top k frequent elements
48. Meeting rooms problem
49. Merge intervals
50. Insert interval

Matrix & 2D Arrays

51. Rotate matrix 90 degrees clockwise
52. Spiral matrix traversal

53. Search in 2D matrix
54. Set matrix zeros
55. Transpose of matrix
56. Find element in row-wise and column-wise sorted matrix
57. Print matrix diagonally
58. Maximum sum rectangle in matrix
59. Count negative numbers in sorted matrix
60. Minimum path sum in matrix

Dynamic Programming on Arrays

61. Longest increasing subsequence
62. Maximum sum increasing subsequence
63. Coin change problem
64. House robber problem
65. Jump game (can reach end)
66. Minimum jumps to reach end
67. Best time to buy and sell stock
68. Best time to buy and sell stock II
69. Palindromic substrings
70. Longest palindromic substring

HARD LEVEL (71-100)

Advanced Algorithms

71. Maximum rectangle in binary matrix
72. Largest rectangle in histogram
73. Trapping rainwater
74. Sliding window maximum
75. Median of two sorted arrays
76. Merge k sorted arrays
77. Find duplicate number (Floyd's cycle detection)
78. First missing positive integer
79. Maximum gap between elements after sorting
80. Count inversions in array

Complex Pattern Problems

81. Russian doll envelopes (2D LIS)
82. Best time to buy and sell stock III (2 transactions)
83. Best time to buy and sell stock IV (k transactions)
84. Maximum profit with cooldown
85. Minimum number of platforms required
86. Job scheduling with profit
87. Activity selection problem
88. Fractional knapsack
89. 0/1 Knapsack problem
90. Unbounded knapsack

Advanced Data Structure Integration

91. LRU Cache implementation
92. Design hit counter
93. Design moving average from data stream
94. Find median from data stream
95. Kth largest element in stream
96. Design stack using arrays
97. Design queue using arrays
98. Design circular queue
99. Design deque using arrays
100. Implement min stack

Key Concepts Covered

Time Complexities to Master:

- O(1) - Direct access, hash operations
- O(log n) - Binary search, heap operations
- O(n) - Linear traversal, single pass algorithms
- O(n log n) - Efficient sorting, divide & conquer
- $O(n^2)$ - Nested loops, brute force approaches
- $O(2^n)$ - Exponential (recursive solutions)

Space Complexities:

- O(1) - In-place algorithms
- O(n) - Additional arrays, recursion stack
- O(k) - Limited extra space

Essential Techniques:

1. **Two Pointers** - Left/right, fast/slow pointers
2. **Sliding Window** - Fixed/variable size windows
3. **Binary Search** - Search space reduction
4. **Sort & Search** - Preprocessing for efficiency
5. **Hash Maps** - O(1) lookups and frequency counting
6. **Prefix Sums** - Range sum queries
7. **Dynamic Programming** - Optimal substructure
8. **Greedy Algorithms** - Local optimal choices
9. **Divide & Conquer** - Problem decomposition
10. **Stack/Queue** - LIFO/FIFO operations

Problem Patterns:

- **Frequency Counting** - Hash maps for occurrences
- **Range Queries** - Prefix sums, segment trees
- **Subsequence Problems** - DP, greedy approaches
- **Interval Problems** - Sorting, merging
- **Matrix Problems** - DFS/BFS, DP
- **Stock Problems** - State machines, DP
- **Game Theory** - Minimax, optimal strategy

Interview Tips:

1. Always clarify constraints and edge cases
2. Start with brute force, then optimize
3. Consider multiple approaches before coding
4. Test with examples including edge cases
5. Analyze time and space complexity
6. Practice coding without IDE assistance
7. Explain your thought process clearly
8. Handle integer overflow scenarios

9. Consider negative numbers and zeros
10. Think about very large input sizes

Company-Specific Focus:

- **Google:** Algorithmic thinking, clean code
- **Facebook/Meta:** System design integration
- **Amazon:** Customer obsession in problem solving
- **Apple:** Attention to detail, edge cases
- **Netflix:** Scalability considerations
- **Microsoft:** Code quality and optimization

This comprehensive list covers all major array operations and algorithmic patterns essential for cracking FAANG interviews. Practice these systematically, focusing on understanding patterns rather than memorizing solutions.