Top 100 String Questions for FAANG Interviews

EASY LEVEL (1-30)

Basic String Operations

1. Find length of string without using built-in function
2. Reverse a string in-place
3. Check if two strings are equal
4. Convert string to uppercase/lowercase
5. Count vowels and consonants in string
6. Remove spaces from string
7. Check if character is vowel or consonant
8. Find ASCII value of characters
9. Count frequency of each character
10. Remove duplicates from string

String Comparison & Validation

11. Check if string is palindrome
12. Check if string contains only digits
13. Check if string contains only alphabets
14. Validate if string is alphanumeric
15. Compare two strings lexicographically
16. Check if string is empty or null
17. Find first non-repeating character
18. Check if string has all unique characters
19. Verify balanced parentheses
20. Check if string is a valid number

Basic Pattern Matching

21. Find substring in string (naive approach)
22. Count occurrences of substring
23. Find all occurrences of pattern
24. Check if string starts with given prefix
25. Check if string ends with given suffix

26. Replace all occurrences of character
27. Remove given character from string
28. Find longest common prefix in array of strings
29. Check if strings are anagrams
30. Group anagrams together

MEDIUM LEVEL (31-70)

Advanced String Manipulation

31. Reverse words in a string
32. Remove extra spaces between words
33. Implement string compression (aabcc → a2b1c2)
34. Expand compressed string
35. Convert roman numeral to integer
36. Convert integer to roman numeral
37. Add two numbers represented as strings
38. Multiply two numbers represented as strings
39. Implement atoi (string to integer)
40. ZigZag string conversion

Pattern Matching & Searching

41. Implement strstr() function
42. KMP (Knuth-Morris-Pratt) pattern matching
43. Rabin-Karp string matching algorithm
44. Find all anagrams in string
45. Minimum window substring containing all characters
46. Longest substring without repeating characters
47. Longest repeating character replacement
48. String matching with wildcards (* and ?)
49. Regular expression matching
50. Implement wildcard pattern matching

Palindromes & Subsequences

51. Longest palindromic substring
52. Longest palindromic subsequence

53. Minimum insertions to make string palindrome
54. Check if string can be rearranged to palindrome
55. Shortest palindrome by adding characters
56. Count palindromic substrings
57. Longest common subsequence
58. Longest common substring
59. Edit distance between two strings
60. Minimum steps to make strings equal

String Transformation

61. One edit distance between strings
62. Transform string A to B with minimum operations
63. Interleaving strings problem
64. Word break problem
65. Word break II (return all sentences)
66. Decode ways (A=1, B=2, ..., Z=26)
67. Letter combinations of phone number
68. Generate parentheses combinations
69. Restore IP addresses
70. Valid palindrome II (remove at most one character)

HARD LEVEL (71-100)

Advanced Algorithms

71. Suffix array construction
72. Longest repeated substring
73. Find all distinct palindromic sub-strings
74. Minimum number of palindromic partitions
75. Count distinct subsequences
76. Distinct subsequences between two strings
77. Scrambled string problem
78. Word ladder transformation
79. Word ladder II (find all shortest paths)
80. Alien dictionary (topological sort)

Complex Pattern Problems

81. Find duplicate file in system
82. Text justification
83. Basic calculator (evaluate string expression)
84. Basic calculator II (with +, -, *, /)
85. Basic calculator III (with parentheses)
86. Parse lisp expression
87. Ternary expression parser
88. Number of atoms in molecule
89. Decode string (k[encoded_string])
90. Encode and decode strings

String Data Structure Design

91. Design search autocomplete system
92. Design add and search words data structure
93. Implement trie (prefix tree)
94. Word search in 2D board
95. Word search II (multiple words)
96. Design in-memory file system
97. Design log storage system
98. Design compressed string iterator
99. Design phone directory
100. Stream of characters and queries

Key Concepts Covered

Time Complexities to Master:

- O(1) - Direct character access, hash operations
- O(n) - Single pass string traversal
- O(n log n) - Sorting-based approaches
- $O(n^2)$ - Nested loops, DP solutions
- O(n×m) - Two string comparisons
- $O(2^n)$ - Exponential (backtracking solutions)

Space Complexities:

- O(1) - In-place string modifications
- O(n) - Additional string/array storage
- O(k) - Limited vocabulary/character set

Essential String Algorithms:

1. Pattern Matching

- **Naive Algorithm** - O(n×m) brute force
- **KMP Algorithm** - O(n+m) with preprocessing
- **Rabin-Karp** - O(n+m) average, rolling hash
- **Boyer-Moore** - Skip characters efficiently
- **Z-Algorithm** - Linear time pattern matching

2. String Hashing

- **Rolling Hash** - Sliding window hashing
- **Polynomial Hash** - Multiple hash functions
- **Double Hashing** - Collision avoidance

3. Trie (Prefix Tree)

- **Standard Trie** - Word storage and retrieval
- **Compressed Trie** - Space optimization
- **Suffix Trie** - All suffix storage

4. Suffix Structures

- **Suffix Array** - Sorted suffix positions
- **LCP Array** - Longest common prefixes
- **Suffix Tree** - Compressed suffix trie

Problem Patterns:

1. Two Pointers Technique

- Palindrome checking
- String reversal
- Remove duplicates
- Merge operations

2. Sliding Window

- Substring problems
- Character frequency
- Minimum/maximum window
- Fixed/variable window size

3. Dynamic Programming

- Edit distance
- Palindrome problems
- Subsequence matching
- String transformation

4. Backtracking

- Generate combinations
- Word break variations
- Pattern matching with wildcards
- Constraint satisfaction

5. Hash Map/Set

- Character frequency counting
- Anagram detection
- Duplicate finding
- Fast lookups

6. Stack-Based Solutions

- Parentheses validation
- Expression evaluation
- String decoding
- Nested structure parsing

String Manipulation Techniques:

1. In-Place Operations

- Character swapping
- String reversal
- Space removal
- Case conversion

2. String Building

- StringBuilder/StringBuffer usage
- Efficient concatenation
- Memory optimization
- Buffer management

3. Character Set Handling

- ASCII vs Unicode
- Case sensitivity
- Special characters
- Locale considerations

Advanced Concepts:

1. String Compression

- Run-length encoding
- Huffman coding concepts
- Dictionary-based compression
- Pattern-based compression

2. Text Processing

- Tokenization
- Parsing techniques
- Regular expressions
- Natural language processing basics

3. Internationalization

- Unicode handling
- Multi-byte characters
- Collation and sorting
- Normalization

Interview Tips:

1. Problem Analysis

- Identify string length constraints

- Consider character set (ASCII/Unicode)
- Check for case sensitivity
- Handle null/empty strings

2. Optimization Strategies

- Choose appropriate data structures
- Consider preprocessing benefits
- Analyze space-time trade-offs
- Use built-in functions wisely

3. Edge Cases

- Empty strings
- Single character strings
- Very long strings
- Special characters
- Unicode considerations

4. Common Pitfalls

- Off-by-one errors in indexing
- String immutability in some languages
- Memory management
- Integer overflow in hashing

Company-Specific Patterns:

Google

- Complex algorithmic problems
- System design with strings
- Search and indexing problems
- Text processing at scale

Facebook/Meta

- Social media text processing
- Real-time string operations
- Content moderation algorithms
- Internationalization challenges

Amazon

- E-commerce search problems
- Product matching algorithms
- Review and rating text analysis
- Inventory string processing

Apple

- User interface text handling
- Device-specific optimizations
- Accessibility considerations
- Multi-language support

Microsoft

- Office suite text processing
- Search algorithms
- Document parsing
- Language processing

Practice Strategy:

1. **Week 1-2:** Master basic string operations and simple algorithms
2. **Week 3-4:** Focus on pattern matching and advanced manipulation
3. **Week 5-6:** Tackle dynamic programming and complex transformation problems
4. **Week 7-8:** Practice system design problems involving strings
5. **Week 9-10:** Mock interviews and time-constrained problem solving

Key Data Structures for String Problems:

- **Arrays/Lists** - Character storage and manipulation
- **Hash Maps** - Frequency counting and fast lookups
- **Stacks** - Nested structure handling
- **Queues** - BFS in string problems
- **Tries** - Prefix matching and word storage
- **Suffix Arrays** - Advanced string searching

This comprehensive collection covers all essential string algorithms and patterns needed for FAANG interviews. Focus on understanding the underlying principles and practice implementing solutions

efficiently.