DSA Quick Notes 2026

Arrays, Strings, Trees, Graphs, DP, Blind-75 & Common Interview Patterns

Master the patterns that solve 80% of coding interview problems

Last Updated: January 2026

Arrays | Strings | Trees | Graphs | Dynamic Programming | Patterns

Arrays

Random Access, Fixed/Resizable

Key Operations

- **Access:** O(1) - Direct indexing
- **Search:** O(n) - Linear scan
- **Insertion:** O(n) - Shifting elements
- **Deletion:** O(n) - Shifting elements

Common Patterns

Sliding Window | Two Pointers | Divide & Conquer | Prefix Sum | Cyclic Sort

Blind-75 Problems

- Two Sum
- Best Time to Buy/Sell Stock
- Contains Duplicate
- Product of Array Except Self
- Maximum Subarray

```
// Sliding Window Example [1, 3, 2, 6, -1, 4, 1, 8, 2], k=5 Window 1: [1, 3, 2, 6, -1] → Avg: 2.2 Window 2: [3, 2, 6, -1, 4] → Avg: 2.8 Window 3: [2, 6, -1, 4, 1] → Avg: 2.4
```

Strings

Immutable sequences

Key Concepts

- Strings are immutable in many languages
- Character encoding matters (ASCII, Unicode)
- StringBuilder for efficient concatenation
- Palindrome, Anagram, Substring problems

Common Patterns

Two Pointers | Sliding Window | Hash Map Counting | Backtracking | Trie

Blind-75 Problems

- Valid Palindrome
- Longest Substring Without Repeating
- String to Integer (atoi)
- Longest Palindromic Substring
- Group Anagrams

```
// Anagram Check s1 = "listen", s2 = "silent" Frequency Map for s1: l:1, i:1, s:1, t:1, e:1, n:1 Frequency Map for s2: s:1, i:1, l:1, e:1, n:1, t:1 Maps are identical → They are anagrams
```

Trees

Hierarchical Data Structures

Tree Types

- **Binary Tree:** 0-2 children per node
- **BST:** Left < Root < Right property
- **AVL/Red-Black:** Self-balancing trees
- **Heap:** Complete binary tree, heap property
- **Trie:** Prefix tree for strings

Traversal Patterns

BFS (Level Order) | DFS (Pre/In/Post) | Morris Traversal

Blind-75 Problems

- Maximum Depth of Binary Tree
- Validate Binary Search Tree
- Binary Tree Level Order Traversal
- Serialize and Deserialize Binary Tree
- Implement Trie (Prefix Tree)

```
// Binary Tree Traversal Tree: 1 / \ 2 3 / \ / 4 5 6 Pre-order: 1 → 2 → 4 → 5 → 3 → 6 (Root, Left, Right) In-order: 4 → 2 → 5 → 1 → 6 → 3 (Left, Root, Right) Post-order: 4 → 5 → 2 → 6 → 3 → 1 (Left, Right, Root)
```

Graphs

Nodes & Edges, V & E

Representations

- **Adjacency List:** O(V+E) space, good for sparse
- **Adjacency Matrix:** O(V^2) space, good for dense
- **Edge List:** Simple but slower queries

Algorithm Guide

Algorithm	Use Case	Time
BFS	Shortest path (unweighted)	O(V+E)
DFS	Connectivity, cycles	O(V+E)
Dijkstra	SSSP (non-negative)	O(E log V)
Topological Sort	DAG ordering	O(V+E)

Blind-75 Problems

- Clone Graph
- Course Schedule
- Number of Islands
- Alien Dictionary
- Graph Valid Tree

Dynamic Programming

Memoization & Tabulation

When to Use DP

- Optimal substructure
- Overlapping subproblems
- Count, minimize, maximize problems
- "How many ways" questions

DP Patterns

0/1 Knapsack | Longest Common Subseq | Fibonacci Pattern | Palindromic Subseq | Matrix Chain Multiplication

Framework

1. Define state dp[i] or dp[i][j]
2. Define recurrence relation
3. Base cases initialization
4. Fill table in correct order
5. Return answer

Blind-75 Problems

- Climbing Stairs
- Coin Change
- Longest Increasing Subsequence
- Word Break
- House Robber

```
// Fibonacci DP dp[0] = 0, dp[1] = 1 dp[i] = dp[i-1] + dp[i-2] Tabulation: Index: 0 1 2 3 4 5 6 7 Value: 0 1 1 2 3 5 8 13
```

Common Patterns

Solve 80% of interview problems

Top 6 Patterns

- **Sliding Window:** Fixed/variable window, subarray problems
- **Two Pointers:** Sorted arrays, pair searching
- **Fast & Slow:** Cycle detection, middle element
- **BFS/DFS:** Tree/Graph traversal, shortest path
- **Backtracking:** Permutations, combinations, subsets
- **Divide & Conquer:** Merge sort, binary search, matrix ops

Pattern Identification Guide

Problem Type	Likely Pattern
"Subarray with sum/condition"	Sliding Window
"Sorted array", "pair sum"	Two Pointers
"All permutations/combinations"	Backtracking
"Shortest path", "minimum steps"	BFS
"Count ways", "minimum/maximum"	Dynamic Programming

```
// Fast & Slow Pattern (Floyd's) Linked List: 1 → 2 → 3 → 4 → 5 → 3 (cycle) Slow: 1 → 2 → 3 → 4 → 5 → 3 → 4 → 5... Fast: 1 → 3 → 5 → 4 → 3 → 5 → 4... They meet at node 4 → Cycle detected!
```

Interview Strategy & Tips (2026)

Communication First

Think out loud. Interviewers care more about your thought process than the final answer. Ask clarifying questions before jumping to solutions.

Time Management

Spend 5-10 minutes understanding, 15-20 minutes solving, 5-10 minutes testing. If stuck after 10 minutes, ask for a hint.

Code Quality

Write clean, readable code with meaningful variable names. Include comments for complex logic. Check edge cases explicitly.

Optimize Progressively

Start with brute force, then optimize. Explain time/space complexity at each step. "Make it work, make it right, make it fast."

Blind-75 Priority Checklist

Master these 15 problems first (covers all patterns)

- [x] Two Sum (Arrays/Hash)
- [x] Best Time to Buy/Sell Stock (Arrays)
- [x] Contains Duplicate (Arrays/Hash)
- [x] Product of Array Except Self (Arrays)
- [x] Maximum Subarray (Arrays/DP)
- [x] Valid Palindrome (Strings)
- [x] Longest Substring Without Repeating (Strings/Sliding)
- [x] Validate BST (Trees)
- [x] Invert Binary Tree (Trees)
- [x] Number of Islands (Graphs/DFS)
- [x] Climbing Stairs (DP)
- [x] Coin Change (DP)
- [x] LRU Cache (Design)
- [x] Merge Intervals (Arrays/Sorting)
- [x] Longest Increasing Subsequence (DP)

Complete these 15, then expand to full Blind-75. Quality > Quantity.