ADOBE UNIVERSITY HACKATHON 2026

Cheat Sheet 2 — Algorithms

Searching, sorting, graphs, recursion, techniques. Know WHAT each does and WHEN to use it.

A. Searching

Algorithm	How it works	When to use
Linear search	Check each element one by one	Unsorted / small data
Binary search	Halve a sorted range each step	Sorted array, fast lookup
Interpolation	Estimate position by value	Sorted + uniformly spread

Binary search on 1,000 elements ≈ 10 comparisons ($\log_2 1000$). On 1,000,000 ≈ 20.

B. Sorting — key properties

Algorithm	Idea	Stable / In-place
Bubble	Swap adjacent out-of-order pairs	Stable / In-place
Selection	Pick min, place at front	Not stable / In-place
Insertion	Insert each into sorted left part	Stable / In-place
Merge	Divide, sort halves, merge	Stable / NOT in-place
Quick	Partition around a pivot	Not stable / In-place
Heap	Build heap, extract max repeatedly	Not stable / In-place

Nearly-sorted data ⟶ insertion sort (O(n) best). Guaranteed O(n log n) + stable ⟶ merge sort. Small integer range ⟶ counting sort.

C. Graph algorithms

Algorithm	Uses	One-line purpose
BFS	Queue	Level-by-level; shortest path in unweighted graph
DFS	Stack / recursion	Go deep; cycle detection, topological sort
Dijkstra	Min-heap (priority queue)	Shortest path, non-negative weights
Bellman-Ford	Edge relaxation × (V-1)	Shortest path, handles negative weights
Kruskal	Union-Find + sort edges	Minimum spanning tree
Prim	Min-heap	Minimum spanning tree
Topological sort	DFS / in-degree (Kahn)	Order a DAG (dependencies)

Tree facts: a tree with n nodes has n−1 edges. A binary tree has at most 2^l nodes at level l, and $2^{h+1}-1$ nodes in a perfect tree of height h.

D. Tree traversals

Traversal	Order	Key use
Inorder	Left → Root → Right	BST → sorted output
Preorder	Root → Left → Right	Copy / serialize a tree
Postorder	Left → Right → Root	Delete / free a tree
Level-order	Breadth first (by level)	BFS, shortest unweighted

E. Recursion & Dynamic Programming

Recursion needs a base case (stops infinite calls) + a recursive case (shrinks the problem).

Term	Meaning
Memoization	Top-down: recurse + cache results
Tabulation	Bottom-up: fill a table iteratively
Overlapping subproblems	Same sub-answers reused → DP helps
Optimal substructure	Best answer built from best sub-answers

Classic DP: Fibonacci, Longest Common Subsequence, 0/1 Knapsack, Coin Change, Edit Distance.

Naive Fibonacci = $O(2^n)$; with memoization = $O(n)$. Tower of Hanoi n disks = 2^n-1 moves.

F. Core techniques

Technique	Spot it when...	Example
Two pointers	Sorted array, pair/target	Pair sum, remove duplicates
Sliding window	Subarray/substring of size k	Max sum window, longest substring
Hashing / frequency	Counting, seen-before checks	Two Sum, anagrams, top-K
Greedy	Local best → global best	Activity selection, Huffman
Divide & conquer	Split, solve, combine	Merge sort, binary search

G. Bit manipulation quick reference

Expression	Result / meaning
n & 1 == 0	n is even
n & (n-1) == 0	n is a power of two
x << 1	multiply by 2
x >> 1	integer divide by 2
a ^ a	0 (XOR with self)
a ^ 0	a
swap via XOR	a^=b; b^=a; a^=b
count set bits	Brian Kernighan: n &= (n-1) loop