NSCT Data Structures & Algorithms Cheat Sheet

By the NSCT Exam Editorial Team · Updated 2026-11-24 · Facts checked against official sources on 2026-11-24

This sheet holds the complexity tables, tree formulas and algorithm comparisons behind most Data Structures & Algorithms questions. DSA carries 10% of the NSCT per HEC's Areas of Competencies, and 2026 candidates averaged 40.2% accuracy in it, one of the four weakest areas, per the HEC NSCT report and APP. Print it (or save it as a PDF from your browser) and keep it next to your practice.

Key definitions

Term	One-line meaning
O(g) / Ω(g) / Θ(g)	Upper bound / lower bound / tight bound (both)
Best, worst, average case	Input scenarios; any of them can be stated with O, Ω or Θ
Amortised cost	Average cost per operation over a sequence (doubling-array append O(1))
ADT	What operations do (stack: push, pop), not how
Stable sort	Equal keys keep their original order
In-place	O(1) extra space (some texts allow O(log n) stack)
Height of a tree	Longest root-to-leaf path in edges (CLRS); some texts count nodes
Complete binary tree	All levels full except the last, filled left to right
Load factor α	n ÷ m (keys ÷ slots)
Optimal substructure	An optimal answer contains optimal answers to subproblems
Overlapping subproblems	The same subproblem recurs; DP stores it

Growth order

$O(1) < O(\log n) < O(\sqrt{n}) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$

Sequential loops add; nested loops multiply; a counter that doubles or halves gives O(log n). The log base does not matter in Big-O.

Data structure operations (typical)

Structure	Access	Search	Insert	Delete
Array	O(1)	O(n)	O(n) (end: O(1) amortised)	O(n)
Singly linked list	O(n)	O(n)	O(1) at head	O(1) at head
Stack / queue	n/a	O(n)	O(1)	O(1)
Hash table	n/a	O(1) expected, O(n) worst	O(1) expected	O(1) expected
BST (unbalanced)	n/a	O(h): O(log n) to O(n)	O(h)	O(h)
AVL / red-black	n/a	O(log n)	O(log n)	O(log n)
Binary heap	Min/max O(1)	O(n)	O(log n)	Extract O(log n)
Trie (key length L)	n/a	O(L)	O(L)	O(L)

Build heap: O(n). Union-find with rank and path compression: O(α(n)) amortised.

Sorting

Algorithm	Best	Average	Worst	Extra space	Stable
Bubble (with flag)	O(n)	$O(n^2)$	$O(n^2)$	O(1)	Yes
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)	No

Algorithm	Best	Average	Worst	Extra space	Stable
Insertion	O(n)	$O(n^2)$	$O(n^2)$	O(1)	Yes
Merge	O(n log n)	O(n log n)	O(n log n)	O(n)	Yes
Quick	O(n log n)	O(n log n)	$O(n^2)$	O(log n) average	No
Heap	O(n log n)	O(n log n)	O(n log n)	O(1)	No
Counting (key range k)	O(n + k)	O(n + k)	O(n + k)	O(n + k)	Yes

Comparison sorts need Ω(n log n) in the worst case. Bubble sort swaps = number of inversions. Selection sort always makes n(n − 1) ÷ 2 comparisons.

Searching

Algorithm	Requirement	Time
Linear	None	O(n); average successful (n + 1) ÷ 2 comparisons
Binary	Sorted, random access	O(log n); worst $\lfloor \log_2 n \rfloor + 1$ probes (1,000 items → 10)
Jump	Sorted	O(√n)
Interpolation	Sorted, uniform keys	O(log log n) average, O(n) worst
Exponential	Sorted	O(log i) for target at index i

Tree and graph formulas

Fact	Formula	Worked check
Max nodes, binary tree of height h (edges)	2^(h + 1) − 1	h = 4 → 31
Min height for n nodes	$\lfloor \log_2 n \rfloor$	n = 20 → 4
Leaves in a binary tree	Two-child nodes + 1	
Edges in a tree	n − 1	
Heap array (0-based)	Children 2i + 1, 2i + 2; parent ⌊(i − 1) ÷ 2⌋	
Distinct BSTs with n keys	Catalan: C(2n, n) ÷ (n + 1)	n = 3 → 5
Max edges, simple undirected graph	n(n − 1) ÷ 2	
Sum of degrees	2E (handshaking lemma)	
MST edges	V − 1	
Adjacency matrix / list space	$\Theta(V^2)$ / Θ(V + E)	

Traversal	Order
Preorder	Node, left, right
Inorder	Left, node, right (sorted for a BST)
Postorder	Left, right, node
Level order	By depth, using a queue

Inorder plus preorder (or postorder) rebuilds a unique binary tree; preorder plus postorder does not.

Graph algorithms

Algorithm	Solves	Negative edges?	Time
BFS	Shortest paths, unweighted	n/a	O(V + E)
DFS	Reachability, cycles, topological order	n/a	O(V + E)
Dijkstra (binary heap)	Single source	No	O((V + E) log V)
Bellman-Ford	Single source, detects negative cycles	Yes	O(VE)

Algorithm	Solves	Negative edges?	Time
Floyd-Warshall	All pairs	Yes, no negative cycles	$O(V^3)$
Kruskal	MST (sort edges, union-find)	n/a	O(E log E)
Prim (binary heap)	MST (grow one tree)	n/a	O(E log V)
Kahn's topological sort	DAG ordering	n/a	O(V + E)

Recurrences and the master theorem

Recurrence	Solution	Example
T(n) = T(n/2) + 1	Θ(log n)	Binary search
T(n) = 2T(n/2) + n	Θ(n log n)	Merge sort
T(n) = 2T(n/2) + 1	Θ(n)	Tree traversal
T(n) = T(n − 1) + 1	Θ(n)	Linear recursion
T(n) = T(n − 1) + n	$\Theta(n^2)$	Worst-case quicksort
T(n) = 2T(n − 1) + 1	$\Theta(2^n)$	Towers of Hanoi

Master theorem for T(n) = aT(n/b) + f(n): compare f(n) with n^(log_b a). Smaller (polynomially) → Θ(n^(log_b a)); equal → multiply by log n; larger (with the regularity condition) → Θ(f(n)). It does not apply to T(n − 1) forms.

Hashing

Method	Clustering
Linear probing	Primary clustering
Quadratic probing	Secondary clustering
Double hashing	Avoids both
Separate chaining	No clustering; α can exceed 1

Use a prime table size not close to a power of 2 for the division method. Deleting under open addressing needs a tombstone.

Design techniques

Technique	Classic examples
Divide and conquer	Merge sort, quicksort, binary search
Greedy	Huffman, Kruskal, Prim, Dijkstra, activity selection, fractional knapsack
Dynamic programming	0/1 knapsack, LCS, edit distance, Floyd-Warshall, Bellman-Ford
Backtracking	N-Queens, Sudoku, subset sum

X vs Y quick distinctions

Pair	Difference
Stack vs queue	LIFO vs FIFO
Array vs linked list	O(1) access vs O(1) insert at a known node
BFS vs DFS	Queue vs stack or recursion
Merge vs quick sort	Stable, O(n) space, always n log n vs in place, $O(n^2)$ worst
Dijkstra vs Bellman-Ford	No negative edges, faster vs handles negatives
Prim vs Kruskal	One growing tree vs merging forest
Greedy vs DP	Choice never revisited vs stored subproblem answers

Pair	Difference
Heap vs BST	Parent–child order vs whole-subtree order
Chaining vs open addressing	Lists per slot vs probing inside the table
KMP vs Rabin-Karp	O(n + m) worst via prefix function vs rolling hash, O(nm) worst

Classic traps

1. Equating worst case with O and best case with Ω.
2. Assuming BST search is always O(log n); an unbalanced BST is O(n).
3. Binary search on a linked list; reaching the middle is O(n).
4. Calling heap sort stable because it is always O(n log n).
5. Using Dijkstra with a negative edge.
6. Applying the master theorem to T(n − 1) recurrences.
7. Assuming linked-list insertion is O(1) before you have found the position.
8. Forgetting to wrap from slot m − 1 to slot 0 when probing.
9. Mixing height in edges with height in nodes.
10. Using greedy for 0/1 knapsack, or greedy coin change for coins 1, 3, 4 (amount 6: greedy 3 coins, optimal 2).
11. Treating a Rabin-Karp hash match as a confirmed match without checking characters.
12. Leaving out the recursion stack when stating space complexity.

Keep going

- Full notes with worked MCQs: DSA revision notes
- All 12 topics: DSA MCQ hub
- Deep practice: Big-O practice, sorting comparison, tree traversals, graph traversal traces and dynamic programming

Next step

Trace a timed set in a subject-wise mock test, then take a full NSCT mock test.