Placement Drives Criteria & Percentage Breakage

40 %	10 %	20 %	20 %	10 %
Binary Search	Recursion Backtracking	Trees Graphs	Arrays Strings Heaps Stacks Queues Linked Lists	Greedy Algos Bit Manipulation Dynamic Programming

- Square Root
- 1st, last, Total Occurrences of an element in an array
- Missing Element in an array
- Peak in a Mountain
  - Aggressive Cows
  - Search In a 2D Matrix

Leet Code / Coding Ninjas

Binary Search → Sorted Array

int n = 36 → Square Root = 6

0 — 36 (s, e)

m = s+e/2 = 0+36/2 = 18

m×m = 18×18 = 324 > 36

e = m-1 = 18-1 = 17

0 — 17

m = 17/2 = 8

8×8 = 64 > 36

e = m-1 = 8-1 = 7

0 — 7

m = 7/2 = 3

ans = mid = 3

3×3 = 9 < 36

s = m+1 ⇒ 4 — 7

m = 11/2 = 5

5×5 = 25 < 36

s = m+1 = 6 — 7

m = 13/2 = 6 (mid)

6×6 = 36 → ans

* Bubble Sort Algorithm: Asc / Des n = 4, Rounds = n-1 = 3

7 6 4 3

Round 1 / Pass 1 / Iteration 1: 7 6 4 3 → 6 7 4 3 → 6 4 7 3 → 6 4 3 7

Round 2: 6 4 3 7 → 4 6 3 7 → 4 3 6 7

Round 3: 4 3 6 7 → 3 4 6 7 → Sorted

for (iterations) → 1 to n O(n)

for (swap) → 1 to n O(n)

O(n×n) = O(n²)

j = 0 ; j < n-1

Selection Sort (Swap)

i = 0 to n-1 minIndex = i

j = i+1 to n-1 minIndex = j

5 1 3 6 9 2

0 1 2 3 4 5

1 5 3 6 9 2 ✓

1 2 3 6 9 5 ✓

1 2 3 5 9 6 ✓

1 2 3 5 6 9 → Sorted.

minIndex = i

j = i+1

if (arr[j] < arr[minIndex]) { minIndex = j } (swap)

TC: O(n²)

for { if (minIndex != i) swap (arr[i], arr[minIndex]) } for

Insertion Sort: (Shift)

9 1 2 0 4 3 6

1 9 2 0 4 3 6

1 2 9 0 4 3 6

0 1 2 9 4 3 6

0 1 2 4 9 3 6

0 1 2 3 4 9 6

0 1 2 3 4 6 9

arr[j] arr[j+1]

int temp = ✗

while( )

j--

j == 0 = -1

6