What is Big-O Notation?

Worksheet

Big-O notation, written O(f(n)), gives the upper bound on how an algorithm's cost scales with input size n. Common classes include O(1), O(log n), O(n), O(n log n) and O(n), from fastest to slowest growth.

Questions

1. What does O(n) mean?

A) Constant time
B) Runtime grows linearly with input size
C) Runtime is always 1 second
D) Runtime grows exponentially

2. Binary search on a sorted array has what time complexity?

A) O(n)
B) O(n)
C) O(log n)
D) O(1)

3. Which grows fastest as n increases?

A) O(log n)
B) O(n)
C) O(n)
D) O(1)

4. Why is Big-O usually about the worst case?

A) It's easier to compute
B) It guarantees a performance ceiling regardless of input arrangement
C) Best case never happens
D) It ignores input size entirely

5. A loop runs once for every element in a list of 8 items. What's its Big-O?

6. Binary search on a sorted list of 1,024 items - how many steps at worst?

7. Nested loops, both running n times over a list of n = 6 items - how many operations?

8. Define: What does Big-O notation measure?

9. Define: Order the common Big-O classes from fastest to slowest.

10. Define: What is O(1)?

Answer Key

1. B) Runtime grows linearly with input size - O(n) means the number of operations scales directly and proportionally with n.

2. C) O(log n) - Binary search halves the search space each step, giving logarithmic growth.

3. C) O(n) - O(n) grows quadratically, outpacing linear and logarithmic growth for large n.

4. B) It guarantees a performance ceiling regardless of input arrangement - Worst-case analysis guarantees the algorithm won't perform worse than that bound, which is the safest planning assumption.

5. 1 pass, 8 comparisons, 1 operation per element T(n) = n O(n) For n = 8, roughly 8 operations

6. Each step halves the search space: 1024 512 256 ... 1 Number of halvings = log(1024) = 10 O(log n) about 10 comparisons, not 1024

7. Outer loop: 6 iterations Inner loop: 6 iterations each time Total = 6 6 = 36 operations T(n) = n O(n)

8. How an algorithm's time or space cost grows as input size n increases, focusing on the worst case.

9. O(1) < O(log n) < O(n) < O(n log n) < O(n) < O(2).

10. Constant time - the operation takes the same time no matter how large the input is (e.g., accessing an array by index).