CS12: Introduction to Big O Notation

Understanding Algorithm Efficiency

Mr. Gullo

Learning Objectives

By the end of this lesson, you will be able to:

- Define algorithm efficiency in your own words
- Identify and explain common Big O notations
- Analyze simple algorithms to determine their time complexity
- Compare different algorithms based on their efficiency

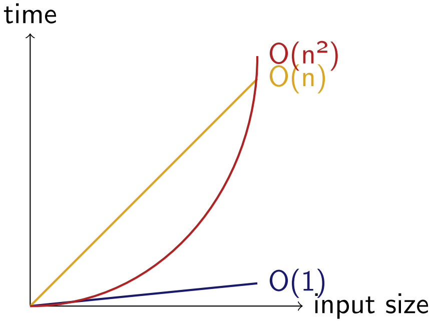
What is Big O Notation?

Definition

Big O notation measures how an algorithm's running time or space requirements grow as the input size increases.

- Think of it as a way to measure an algorithm's speed
- Helps us compare different solutions
- Focuses on the worst-case scenario
- Ignores smaller details and focuses on the big picture

How Different Algorithms Grow



As input grows...

Understanding Through Real Examples

O(1) - Constant Time

- Finding a book on your desk
- Looking up array element by index

O(n) - Linear Time

- Searching through a line of books
- Finding maximum in unsorted array

O(n^2) - Quadratic Time

- Comparing every book with others
- Bubble sort algorithm

O(log n) - Logarithmic Time

- Finding word in dictionary
- Binary search

I Do: Analyzing Linear Search

Problem

Let's analyze this linear search algorithm:

```
int linearSearch(int arr[], int n, int x) {
    for(int i = 0; i < n; i++) {
        if(arr[i] == x) {
            return i;  // Found it!
        }
    }
    return -1;  // Not found
}
```

- Time Complexity: O(n)
- Why? In worst case, we check every element

We Do: Let's Analyze Together

What's the time complexity?

```
void printPairs(int arr[], int n) {
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            cout << arr[i] << ","
                 << arr[j] << endl;
        }
    }
}
```

We Do: Let's Analyze Together

What's the time complexity?

```
void printPairs(int arr[], int n) {
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            cout << arr[i] << ","
                 << arr[j] << endl;
        }
    }
}
```

- Let's count the operations...
- Outer loop runs n times
- For each outer loop, inner loop runs n times
- Total operations: $n \times n = n^2$

You Do: Practice Time!

Analyze These Operations

Determine the Big O notation for:

1. Getting the first element of an array
2. Finding the maximum value in an unsorted array
3. Checking if a number is even or odd

You Do: Practice Time!

Analyze These Operations

Determine the Big O notation for:

1. Getting the first element of an array
2. Finding the maximum value in an unsorted array
3. Checking if a number is even or odd

Solutions

1. O(1) - Direct access, no matter the size
2. O(n) - Must check every element once
3. O(1) - Single operation, size independent

Key Takeaways

Remember These Points

- Big O notation helps us measure efficiency
- Most common notations: O(1), O(n), $O(n^2)$
- Consider how performance changes with input size
- Different problems require different solutions

Practice Makes Perfect

Try analyzing algorithms you write in your own code!