Day2_ADSA_Sanket_Shalukar

Wednesday, September 17, 2025 10:12 AM

Topics that are in the day 2

- Recursion
- Arrays

Base Condition

Definition:

- The condition that stops the recursion
- Purpose: Prevents infinite function calls
- When it occurs: When the problem becomes simple enough to solve directly
- Result: Returns a value without making another recursive call

```java
// Example: Base condition in factorial
if (n == 0 || n == 1) {
return 1; // Base case: 0! = 1, 1! = 1
}
```

Recursive Case

Definition:

- The part where function calls itself with modified parameters
- Purpose: Moves toward the base condition by reducing problem size
- Key: Each call must bring us closer to the base case

```java
// Example: Recursive case in factorial
return n * factorial(n - 1); // Calls itself with smaller input
```

Iterative :

Terminates when the loop condition becomes false.

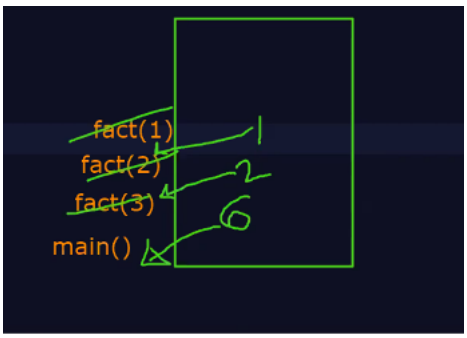
Characteristics:

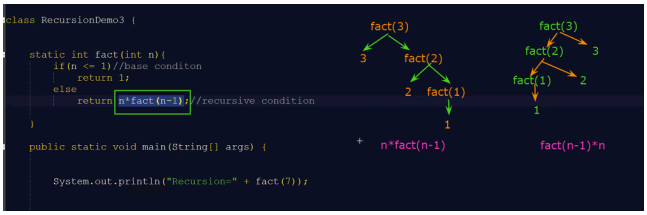
- Termination: When loop condition becomes false
- Memory: Uses constant memory (no function call stack)
- Speed: Generally faster
- Control: Explicit control with loops

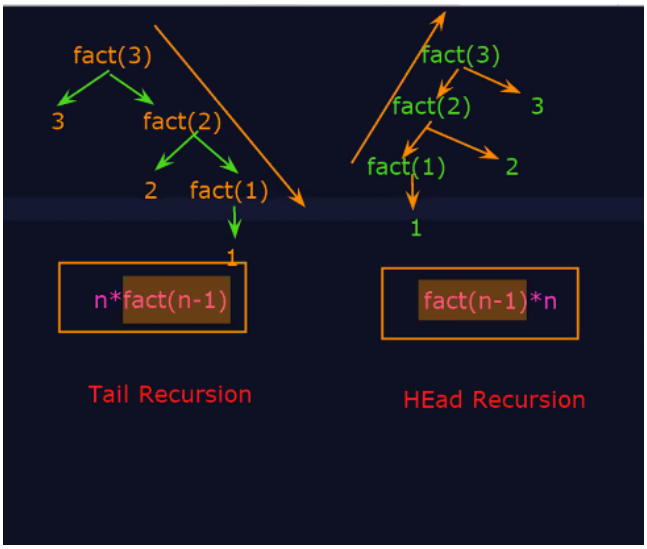
Recursion

Recursion is a programming technique where a function calls itself directly or indirectly to solve a problem by breaking it down into smaller, similar subproblems.

Key Principle: Solve a big problem by solving smaller versions of the same problem.



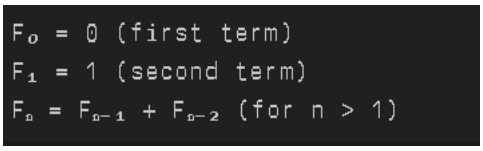


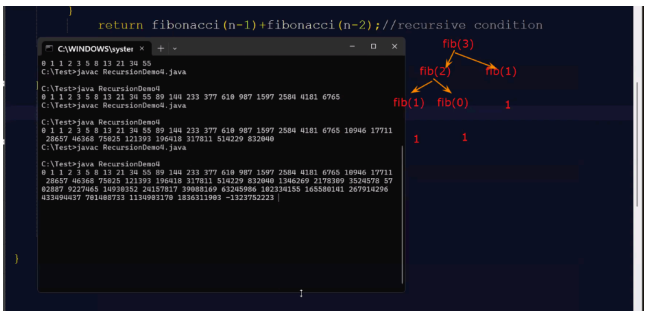


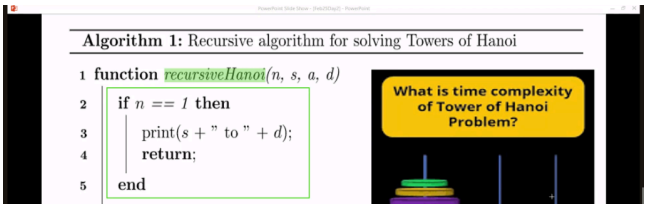
Fibonacci Series Notes

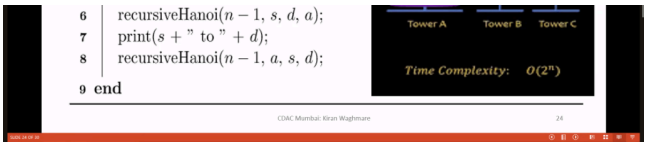
The Fibonacci Series is a sequence of numbers where each number is the sum of the two preceding numbers. It typically starts with 0 and 1.

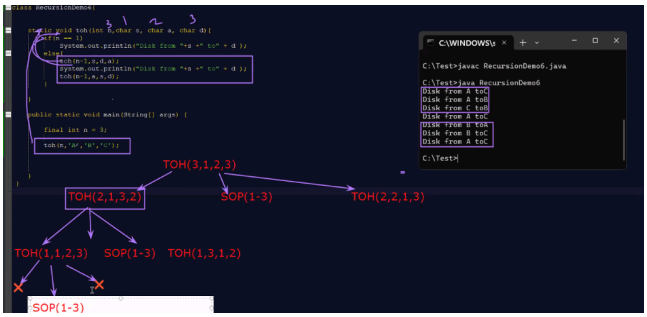
Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987...

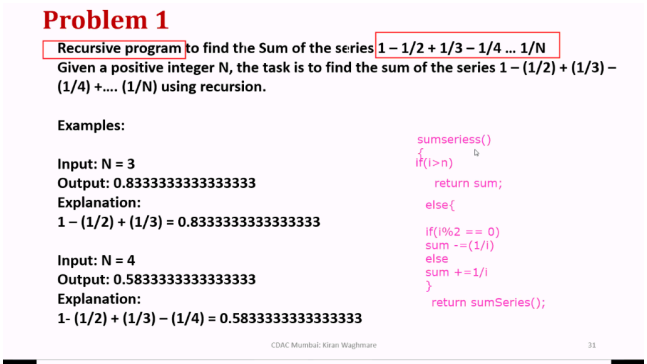


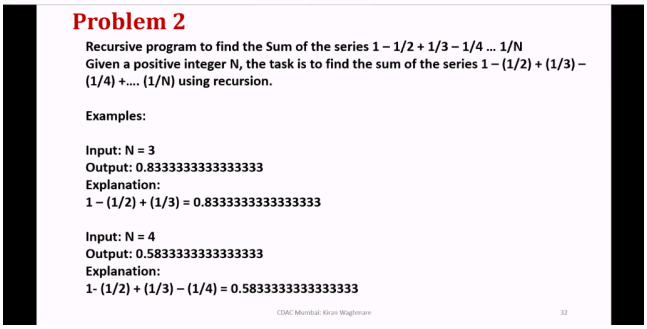


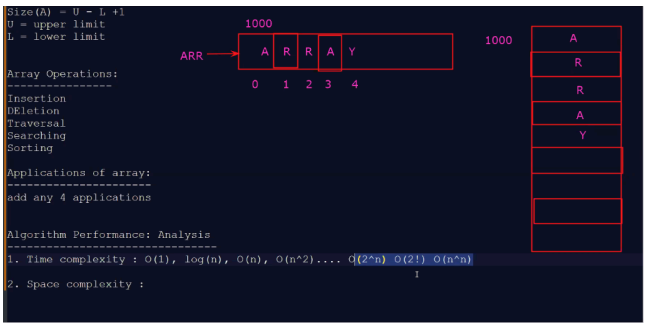












Array Operations

1. **Insertion**

Definition: Adding new elements to an array at specific positions.

Types of Insertion:

- At the end: Add element after the last element
- At the beginning: Add element at index 0 (requires shifting)
- At specific position: Add element at any given index

2. **Deletion**

Definition:

Removing elements from an array at specific positions.

Types of Deletion:

- From end: Remove last element
- From beginning: Remove first element (requires shifting)

- From specific position: Remove element at given index

3. **Traversal**

Definition: Visiting each element of the array exactly once.

4. **Searching**

Definition: Finding the location of a specific element in an array.

Types of Searching:

- Linear Search: Check each element sequentially
- Binary Search: For sorted arrays, divide and conquer approach

5. **Sorting**

Definition: Arranging array elements in ascending or descending order.

Common Sorting Algorithms:

- Bubble Sort: Compare adjacent elements
- Selection Sort: Find minimum and place at beginning
- Insertion Sort: Insert each element in correct position
- Merge Sort: Divide and conquer approach
- Quick Sort: Partition-based sorting

Major Applications of Arrays

1. Database Management Systems

- Use: Store records, indexing, query processing
- Example: Employee database with ID, name, salary arrays

2. Image & Graphics Processing

- Use: Store pixel data, color values, image manipulation
- Example: 2D arrays representing image pixels in photo editing

3. Scientific Computing

- Use: Matrix operations, mathematical calculations, simulations
- Example: Weather prediction models, statistical analysis

4. Gaming & Game Development

- Use: Game boards, maps, player statistics
- Example: Chess board (8×8 array), Tic-tac-toe (3×3 array)

Algorithm Performances Analysis

Time Complexity : 0(1), log(n), O(n), O(2^n)...O(2^n) O(2!) O(n^n)
Spice Complecity : 0(1), log(n), O(n), O(2^n)...O(2^n) O(2!) O(n^n)

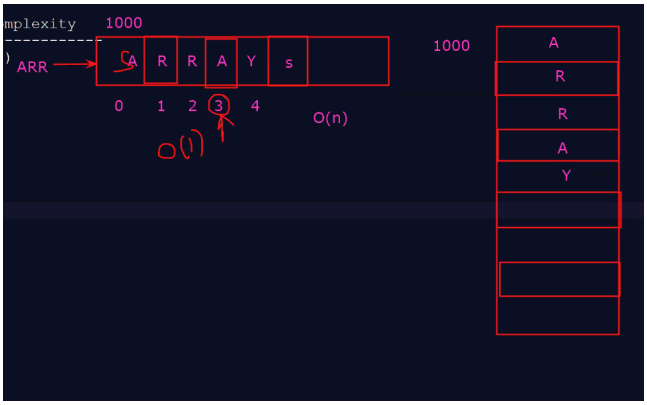
Complexity	Performance	Example	Growth for n=1000
O(1)	Excellent	Array access	1 operation
O(log n)	Very Good	Binary search	~10 operations
O(n)	Good	Linear search	1,000 operations
O(n log n)	Acceptable	Merge sort	~10,000 operations
O(n²)	Poor	Bubble sort	1,000,000 operations
O(2^n)	Very Poor	Recursive Fibonacci	~10^301 operations
O(n!)	Extremely Poor	All permutations	~10^2567 operations
O(n^n)	Worst Possible	Poor recursive design	Astronomical

Array Operations: Time Complexity

Insertion
Deletion
Traversal
Searching
Sorting

Operation	Best Case	Average Case	Worst Case	Space
Access/Read	O(1)	O(1)	O(1)	O(1)
Traversal	O(n)	O(n)	O(n)	O(1)

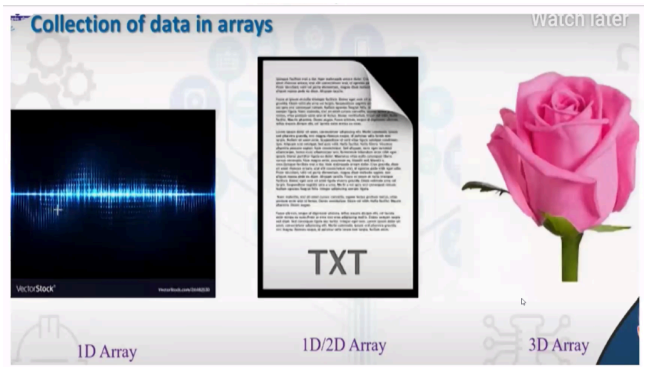
Insert at End	O(1)	O(1)	O(1)	O(1)
Insert at Beginning	O(n)	O(n)	O(n)	O(1)
Insert at Middle	O(n)	O(n)	O(n)	O(1)
Delete from End	O(1)	O(1)	O(1)	O(1)
Delete from Beginning	O(n)	O(n)	O(n)	O(1)
Delete from Middle	O(n)	O(n)	O(n)	O(1)
Linear Search	O(1)	O(n)	O(n)	O(1)
Binary Search	O(1)	O(log n)	O(log n)	O(1)



Searching Algorithm

Array Opreations Time Complexity

Accessing an element : 0(1)
Search an element



Binary Search

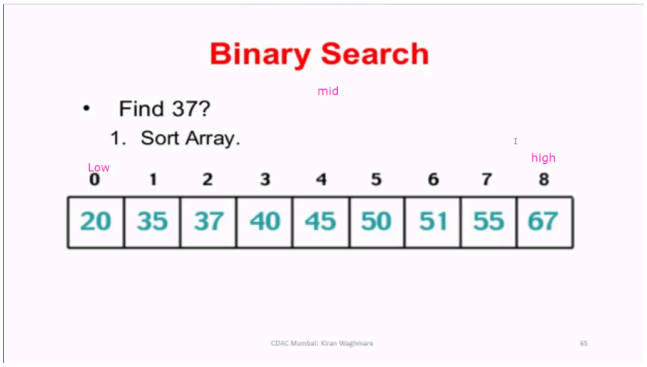
Definition :

Binary search divides the sorted array in half repeatedly, comparing the target with the middle element to eliminate half of the search space each time.

```
    int left = 0, right = arr.length - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == target) {
            return mid;  // Found
        }
        if (arr[mid] < target) {
            left = mid + 1;  // Search right half
        } else {
            right = mid - 1;  // Search left half
        }
    }
    return -1;  // Not found
}
```



Array Operations: Time Complexity

Accessing an element: O(1)

Search an element: O(n)

Insertion of an element : O(n)

Deletion of an element: O(n)

Linear Search: Time Complexity: O(n)

Linear Search

Definition :

Linear search sequentially checks each element in an array from beginning to end until the target element is found or the array is exhausted.