ABHYASMITRA.IN

COMPREHENSIVE UNIVERSITY STUDY NOTES

DATA STRUCTURES

Unit III — Linear Data Structures: Stacks, Recursion & Queues

---

Topics Covered in this Unit

Stack Abstract Data Type (ADT): LIFO Principle, Sequential Array & Linked List Representations, Boundary Conditions

Stack Operations & State Transitions: push(), pop(), peek(), isEmpty(), isFull() with Complete Complexity Analysis

Applications of Stack: Expression Types (Infix, Prefix, Postfix), Operator Precedence & Associativity Rules

Algorithms & Trace Tables: Infix to Postfix Conversion, Infix to Prefix Conversion, Postfix Expression Evaluation

Parenthesis Matching & Syntax Validation: Stack-based Delimiter Balancing Algorithm & Traces

Recursion & Activation Records: Call Stack Dynamics, Stack Frames, Iteration vs. Recursion, Tower of Hanoi Problem

Queue Abstract Data Type (ADT): FIFO Principle, Sequential Representation, Front & Rear Pointer Mechanics

Linear Queue Limitations: False Overflow Phenomenon & Inefficient Memory Utilization

Circular Queue: Modulo Arithmetic Mechanics, Boundary State Formulas, and Elimination of False Overflow

Double-Ended Queue (Deque): Input-Restricted & Output-Restricted Deques, Operations, and Use Cases

Priority Queue: Min/Max Priority Queues, Implementation Strategies (Arrays, Linked Lists, Heaps), Real-World Applications

High-Yield University Exam Question Bank with Detailed Model Answers & Unit Summary

Compiled in a structured, easy-to-understand, textbook style format
Specially curated for B.Tech / B.E. / BCA / MCA / B.Sc Computer Science & Engineering Students

COURSE SYLLABUS & MAPPING

[UNIT III SYLLABUS — PRESCRIBED UNIVERSITY CURRICULUM]

This section contains the official syllabus for Unit III. Verify with your university curriculum mapping.

Unit III Syllabus Breakdown:

Unit III Linear Data Structure :Stacks, Queues: Stack: Introduction of stack, stack Abstract Data Type, Representation of Stacks Using Sequential Organization, stack operations, Applications of Stack- Expression Evaluation and Conversion. Recursion- concept. Queue: Introduction of Queue, Queue as Abstract Data Type, Representation of Queue using Sequential organization. Queue Operations. Circular Queue and its advantages, Deque-introduction, Priority Queue.

• Course Code: _____ • Semester / Year: _____

TABLE OF CONTENTS

3.1 Stack Abstract Data Type & Sequential Representation

Definition: Stack & LIFO Principle

A Stack is a linear data structure following the Last-In, First-Out (LIFO) or First-In, Last-Out (FILO) discipline, where all insertions (Push) and deletions (Pop) occur exclusively at a single designated end called the TOP of the stack.

Stack Abstract Data Type (ADT) Specification

The Stack ADT consists of elements of a given type and the following primitive operations:

- **1. push(item):** Inserts item onto top of stack. Precondition: Stack is not full (overflow check).
- **2. pop():** Removes and returns top item. Precondition: Stack is not empty (underflow check).
- **3. peek() / top():** Returns top item without removing it.
- **4. isEmpty():** Returns TRUE if stack has zero elements, FALSE otherwise.
- **5. isFull():** Returns TRUE if stack has reached maximum allocated capacity, FALSE otherwise.

STACK MEMORY & POINTER STATE TRANSITIONS

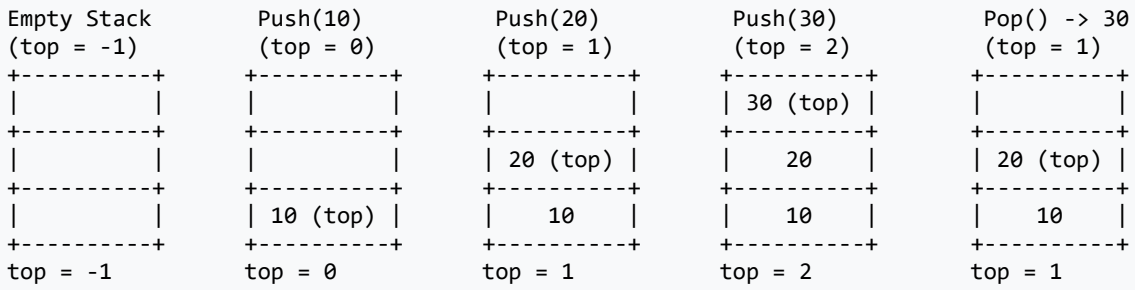


Figure: Sequential array stack state transitions with 'top' pointer updates (O(1) operations)

Sequential vs. Linked Representation of Stack

Feature / Parameter	Sequential (Array) Representation	Linked List Representation
Memory Allocation	Contiguous static array of fixed maximum size MAX.	Dynamic non-contiguous heap nodes linked via pointers.
Stack Overflow	Occurs when top == MAX - 1 (Fixed limit).	Never occurs unless system heap memory is exhausted.

Feature / Parameter	Sequential (Array) Representation	Linked List Representation
Stack Underflow	Occurs when top == -1.	Occurs when top == NULL.
Time Complexity	Push: O(1), Pop: O(1), Peek: O(1).	Push: O(1) (at head), Pop: O(1) (from head).
Space Overhead	Fixed array memory allocated upfront.	Pointer overhead (next pointer per node).

3.2 Applications of Stack: Expression Notations & Conversions

Arithmetic Expression Notations

- **1. Infix Notation:** Operators appear between operands (e.g., A + B * C). Requires operator precedence and parentheses to eliminate ambiguity.
- **2. Postfix Notation (Reverse Polish Notation - RPN):** Operators appear after operands (e.g., A B C * +). Parenthesis-free and evaluated strictly left-to-right.
- **3. Prefix Notation (Polish Notation):** Operators appear before operands (e.g., + A * B C). Parenthesis-free and evaluated right-to-left.

Operator Precedence & Associativity Table

Operator	Description	Precedence Rank	Associativity
^ (or $)	Exponentiation (Power)	3 (Highest)	Right-to-Left (e.g., 2^3^2 = 2^(3^2))
*, /, %	Multiplication, Division, Modulus	2 (Medium)	Left-to-Right
+, -	Addition, Subtraction	1 (Lowest)	Left-to-Right
(	Opening Parenthesis	0 (Inside stack)	Left-to-Right

Algorithm: Infix to Postfix Conversion (Shunting-Yard)

Scan infix string from left to right token by token:

- If token is an Operand: Append directly to postfix output string.
- If token is '(': Push onto operator stack.
- If token is ')': Pop and append operators from stack to output until '(' is encountered; discard '('.

- If token is an Operator (op): While stack is non-empty and precedence(top) >= precedence(op) (with strict > for right-associative '^'), pop and append top to output. Then push op onto stack.
- At End of String: Pop and append all remaining operators from stack to output.

Trace Table: Convert Infix expression: (A + B) * C - D ^ E ^ F to Postfix

Symbol Scanned	*Stack Content*	*Postfix Output*
(	*(*	
A	*(*	*A*
+	*(+*	*A*
B	*(+*	*A B*
)	*[empty]*	*A B +*
***	***	*A B +*
C	***	*A B + C*
-	*-*	*A B + C **
D	*-*	*A B + C * D*
^	*- ^*	*A B + C * D*
E	*- ^*	*A B + C * D E*
^	*- ^ ^*	*A B + C * D E (Right-associative push)*
F	*- ^ ^*	*A B + C * D E F*
[End of Input]	*[empty]*	*A B + C * D E F ^ ^ -*

*Final Postfix Expression: A B + C * D E F ^ ^ -*

Algorithm: Postfix Expression Evaluation

Scan postfix expression from left to right using an operand stack:

- If token is an Operand: Push numerical value onto stack.
- If token is an Operator (op): Pop operand2 = pop(), operand1 = pop(). Compute result = operand1 op operand2. Push result back onto stack.
- At End: Pop final answer from stack.

3.3 Recursion Concept & Run-Time Call Stack

Definition: Recursion & Activation Record

Recursion is a programming technique in which a function calls itself directly or indirectly to solve a smaller instance of the same problem.

• *Base Case: The terminating condition that produces an answer directly without further recursive calls.*

• *Recursive Step: The rule that reduces problem size towards the base case.*

• *Activation Record (Stack Frame): A block of memory pushed onto the system Call Stack for every function invocation, storing parameters, local variables, and return address.*

The Tower of Hanoi Problem

Move n disks from Source peg (A) to Destination peg (C) using Auxiliary peg (B) such that only one disk is moved at a time and a larger disk is NEVER placed on top of a smaller disk.

Tower of Hanoi Algorithm & Recurrence

```
void TOH(int n, char src, char dest, char aux) {
   if (n == 1) {
      printf("Move disk 1 from %c to %c\n", src, dest);
      return;
   }
   TOH(n - 1, src, aux, dest);   // Move top n-1 disks from src to aux
   printf("Move disk %d from %c to %c\n", n, src, dest);
   TOH(n - 1, aux, dest, src);   // Move n-1 disks from aux to dest
}
```

Recurrence: $T(n) = 2T(n-1) + 1$

Solving: $T(n) = 2^n - 1$ moves. Time Complexity = $O(2^n)$, Space Complexity = $O(n)$ call stack.

3.4 Queue Abstract Data Type & Linear Queue

Definition: Queue & FIFO Principle

A Queue is a linear data structure operating under the First-In, First-Out (FIFO) principle, where insertions occur at the REAR end (Enqueue) and deletions occur at the FRONT end (Dequeue).

Drawback of Linear Queue: False Overflow

In a simple array-based linear queue with size MAX, front advances during dequeue and rear advances during enqueue. When rear reaches MAX - 1, isFull() reports TRUE, even if earlier elements have been deleted and front slots are completely empty! This wasteful condition is called False Overflow.

3.5 Circular Queue & Modular Arithmetic Mechanics

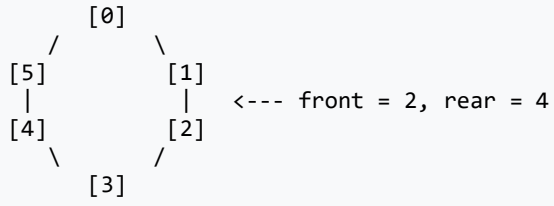
Definition: Circular Queue

A Circular Queue is an optimized queue implementation where the last position connects back to the first position in a ring, utilizing Modulo Arithmetic ((index + 1) % MAX) to reuse vacated front memory slots and eliminate false overflow entirely.

Circular Queue Boundary Conditions

- **1. Queue Empty Condition:** front == -1 (or front == rear after resetting).
- **2. Queue Full Condition:** (rear + 1) % MAX == front.
- **3. Enqueue Operation:** If full -> error. Else if empty -> front = rear = 0; else rear = (rear + 1) % MAX. Insert arr[rear] = val.
- **4. Dequeue Operation:** If empty -> error. Val = arr[front]. If front == rear -> front = rear = -1 (reset); else front = (front + 1) % MAX. Return Val.

CIRCULAR QUEUE RING VISUALIZATION (MAX = 6)




```
Array indexing wraps around: (5 + 1) % 6 = 0.
Guarantees 100% memory utilization of all allocated slots!
```

3.6 Double-Ended Queue (Deque) & Priority Queue

1. Double-Ended Queue (Deque)

A Deque (Double-Ended Queue) allows insertions and deletions at BOTH front and rear ends. It supports four primitive operations: insertFront(), insertRear(), deleteFront(), deleteRear().

- • **Input-Restricted Deque:** Insertion is allowed at ONE end only (e.g., Rear), but deletion is allowed at BOTH ends (Front and Rear).
- • **Output-Restricted Deque:** Deletion is allowed at ONE end only (e.g., Front), but insertion is allowed at BOTH ends (Front and Rear).

2. Priority Queue

Definition: Priority Queue

A Priority Queue is an Abstract Data Type where each element has an associated Priority. Elements are dequeued according to priority rather than arrival order.

• *Max-Priority Queue: Element with highest priority (largest value) is deleted first.*

• *Min-Priority Queue: Element with lowest priority (smallest value) is deleted first.*

Implementation Trade-offs for Priority Queue

Data Structure Implementation	Enqueue (Insert) Time	Dequeue (Extract-Min/Max) Time	Peek (Get-Min/Max) Time
Unordered Array / List	O(1) (Insert at end)	O(n) (Scan for highest priority)	O(n)
Ordered Array / List	O(n) (Insert in sorted order)	O(1) (Remove from front/rear)	O(1)
Binary Heap (Optimal)	O(log n) (Heapify-up)	O(log n) (Heapify-down)	O(1) (Root node)

3.7 High-Yield University Exam Question Bank

Part A: Short Answer Questions (2 to 5 Marks)

Q1. What is False Overflow in a Linear Queue, and how does a Circular Queue solve it? [3 Marks]

- **Answer:** False overflow occurs in a linear array queue when rear reaches the end (MAX-1), preventing further insertions even if elements were dequeued from the front and empty slots exist. A Circular Queue wraps around using modulo arithmetic ((rear+1)%MAX), allowing new elements to occupy the freed front slots, achieving 100% capacity utilization.

Q2. Differentiate between Input-Restricted and Output-Restricted Deques. [3 Marks]

- **Answer:** In an Input-Restricted Deque, insertions can occur at only ONE end, while deletions are allowed at BOTH ends. In an Output-Restricted Deque, deletions can occur at only ONE end, while insertions are allowed at BOTH ends.

Q3. What is an Activation Record / Stack Frame in recursive execution? [3 Marks]

- **Answer:** An Activation Record is a contiguous block of memory allocated on the system Call Stack for every active function call. It stores function arguments/parameters, local variables, intermediate expression values, and the return address to resume execution in the caller function upon return.

Part B: Long Answer Questions (10 to 15 Marks Outline Guides)

- **Q4. Write complete C functions for Stack operations (push, pop, peek, isEmpty, isFull) using arrays and linked lists. Analyze time and space complexity. [12 Marks]**
- **Q5. Convert the Infix expression: ((A + B) * C - (D - E) ^ (F + G)) to Postfix. Show the status of the operator stack and output string at every step. [12 Marks]**
- **Q6. Write an algorithm to evaluate a Postfix expression. Trace evaluation for Postfix string: 6 2 3 + * 12 4 / - 2 ^ 3 +. [10 Marks]**
- **Q7. Write the complete C algorithm for Circular Queue operations (enqueue, dequeue, display). State empty and full boundary conditions with modulo arithmetic. [12 Marks]**
- **Q8. Explain Tower of Hanoi recursion with n = 3 disks. Draw the recursion call tree and derive its time complexity recurrence relation T(n) = 2T(n-1) + 1. [10 Marks]**

3.8 Unit Summary & Key Takeaways

- Stack is a LIFO structure supporting O(1) push, pop, and peek operations at the TOP end.
- Stack applications include expression conversion (Infix to Postfix/Prefix), postfix evaluation, delimiter matching, and runtime call stack management.
- Recursion solves problems by self-calling smaller subproblems, relying on stack frames in system memory to maintain state until the base case is reached.
- Queue is a FIFO structure where elements enter at the REAR and leave from the FRONT.

- Linear queues suffer from false overflow; Circular queues eliminate this by using modulo arithmetic index wrapping ((i + 1) % MAX).
- Deques permit insertions and deletions at both ends (input-restricted vs output-restricted).
- Priority queues dequeue elements based on highest/lowest priority, implemented with optimal O(log n) efficiency via Binary Heaps.