UNIT II · LESSON 02

Stacks & Queues

stack ADT · expression conversion & evaluation · queue · circular queue · deque · priority queue

20 questions in Unit II
Lists · Trees · Sorting · Hashing · Graphs

LIFO STACK ADT

- **Last In, First Out**; one end called **top**
- Operations: `push`, `pop`, `peek/top`, `isEmpty`, `isFull` — all **O(1)**
- Array: `top = -1` empty; `top == MAX-1` full
- **Overflow**: push on full; **underflow**: pop on empty

```
void push(int x) {
    if (top == MAX - 1)
        { puts("Overflow"); return; }
    s[++top] = x;
}
int pop(void) {
    if (top == -1)
        { puts("Underflow"); return -1; }
    return s[top--];
}
```

- Linked stack: push/pop at the **head** of the list; no overflow until memory ends

USES STACK APPLICATIONS

- Function calls & **recursion** (activation records)
- **Expression** conversion (infix → postfix/prefix) and evaluation
- **Balanced parentheses** checking
- Undo/redo, browser back button
- DFS of graphs, backtracking (maze, N-queens)
- Reversing a string, decimal → binary conversion
- Tower of Hanoi, quicksort (iterative)

INFIX · PREFIX · POSTFIX NOTATIONS

Infix	Prefix (Polish)	Postfix (Reverse Polish)
A + B	+AB	AB+
A + B * C	+A*BC	ABC*+
(A + B) * C	*+ABC	AB+C*
A * B + C / D	+*AB/CD	AB*CD/+
A ^ B ^ C	^A^BC	ABC^^ (right assoc.)

INFIX · PREFIX · POSTFIX NOTATIONS (CONTD.)

Infix	Prefix (Polish)	Postfix (Reverse Polish)
(A+B)* (C−D)	*+AB−CD	AB+CD−*

- Postfix/prefix need **no parentheses** and no precedence rules to evaluate
- Precedence: `^` (R→L) > `* / %` > `+ −`

OPERAND STACK POSTFIX EVALUATION

- Operand → push; operator → pop **B** then **A**, push **A op B**
- Example `6 2 3 + − 3 8 2 / + * 2 ^ 3 +`:

Step	Stack
6 2 3 +	6 5
−	1
3 8 2 /	1 3 4
+	1 7
*****	7
2 ^	49
3 +	**52**

- Prefix evaluation: scan **right** → **left**, pop A then B, push A op B

DOUBLE-ENDED DEQUE

- Insert & delete at **both** ends
- **Input-restricted**: insert at one end, delete at both
- **Output-restricted**: delete at one end, insert at both
- Deque can act as both stack and queue

ALGORITHM INFIX → POSTFIX

- Scan left → right
- **Operand** → output
- `(` → push
- `)` → pop to output until `(`; discard both
- **Operator** → pop while top has **higher precedence** (or equal & left-assoc.), then push
- At end, pop everything

Symbol	Stack	Output
A		A
+	+	A
B	+	AB
*****	+*	AB
C	+*	ABC
−	−	ABC*+
D	−	ABC*+D
end		**ABC*+D−**

Infix → prefix: reverse the expression, swap `(` and `)`, convert to postfix, reverse the result.

FIFO QUEUE ADT

- **First In, First Out**; insert at **rear**, delete at **front**
- Operations: `enqueue`, `dequeue`, `front`, `isEmpty`, `isFull` — O(1)
- Linear array queue wastes space: after deletions, rear = MAX−1 looks "full"
- Linked queue: enqueue at tail, dequeue at head (keep both pointers)

MOD CIRCULAR QUEUE

```c
/* front = rear = -1 initially */
void enq(int x) {
  if ((rear + 1) % MAX == front)
      { puts("Full"); return; }
  if (front == -1) front = 0;
  rear = (rear + 1) % MAX;  q[rear] = x;
}
int deq(void) {
  int x = q[front];
  if (front == rear) front = rear = -1;
  else front = (front + 1) % MAX;
  return x;
}
```

- Full: `(rear + 1) % MAX == front`; one convention keeps **one slot empty** → capacity MAX−1
- Count = `(rear - front + MAX) % MAX` (empty-slot convention)

BY PRIORITY PRIORITY QUEUE

- Element with **highest (or lowest) priority** removed first

Implementation	Insert	Delete-min
Unsorted array/list	O(1)	O(n)
Sorted array/list	O(n)	O(1)
Binary heap	**O(log n)**	**O(log n)**
BST (balanced)	O(log n)	O(log n)

- Uses: CPU scheduling, Dijkstra, Prim, Huffman coding, event simulation

Q → A RAPID RECALL (CONTD.)

Stack needed for infix → postfix	operator stack
Pops to reverse n items using stack	n

USES QUEUE APPLICATIONS

- CPU / disk scheduling, printer spooling
- **BFS** of graphs, level-order traversal of trees
- Buffers (keyboard, I/O, network packets)
- Call-centre waiting lines, simulation

Q → A RAPID RECALL

Stack principle	LIFO
Queue principle	FIFO
Postfix of `A*(B+C)/D`	`ABC+*D/`
Prefix of `A*(B+C)/D`	`/*A+BCD`
Value of `5 3 + 8 2 - *`	48
Stack needed for postfix → evaluate	operand stack

Q → A RAPID RECALL (CONTD.)

Queue using two stacks: dequeue cost	O(1) amortised
Stack using two queues: push or pop cost	O(n) for one of them

TRAP EXAM TRAPS

- `pop` before checking empty → underflow
- Circular queue "full" test is `(rear+1) % MAX == front`, not `rear == MAX-1`
- For `^` (right-assoc.), **do not** pop an equal-precedence `^`
- Postfix: the **first** popped value is the **right** operand (`A - B` → pop B first)
- Linear queue may report full even when front slots are free
- Number of stack permutations of 1…n (valid pop orders) = Catalan number C_n