UNIT–2 NOTES

STACKS AND QUEUES (Data Structures)

---

1. STACK

Introduction

Stack is a **linear data structure** in which insertion and deletion are performed from only one end.

This end is called:

```
TOP
```

Stack works on:

LIFO Principle

```
Last In First Out
```

Meaning:
Jo element sabse last me insert hoga,
wo sabse pehle remove hoga.

Real Life Examples

- Plate stack
- Browser back button
- Undo operation
- Recursion
- Call stack

Stack Operations

1. PUSH

Used to insert element into stack.

Example

```
Stack:
10
20
30

Push 40

Stack:
10
20
30
40
```

---

2. POP

Used to delete top element.

Example

```
Stack:
10
20
30

Pop()

Removed = 30
```

---

3. PEEK / TOP

Returns top element without deleting.

---

4. isEmpty()

Checks whether stack is empty.

---

5. isFull()

Checks whether stack is full.

Stack as ADT (Abstract Data Type)

ADT means:
only operations are defined,
implementation hidden hota hai.

Stack ADT Operations

- push()
- pop()
- peek()
- isEmpty()
- isFull()

Implementation of Stack

Stacks can be implemented using:

1. Arrays

2. Linked List

Stack Using Array

Declaration

```
int stack[100];
int top = -1;
```

PUSH Algorithm

```
1. Check overflow
2. top = top + 1
3. stack[top] = value
```

POP Algorithm

```
1. Check underflow
2. value = stack[top]
3. top = top - 1
```

Advantages

- Simple
- Fast access

Disadvantages

- Fixed size
- Memory wastage possible

Stack Using Linked List

Each node contains:

- data
- next pointer

Top points to first node.

Advantages

- Dynamic size
- No overflow until memory full

Disadvantages

- Extra memory for pointer

Multiple Stacks

Multiple stacks can be implemented in a single array.

Example:

```
Stack 1 → left to right
Stack 2 → right to left
```

This improves memory utilization.

Applications of Stack

---

1. Infix to Postfix Conversion

Infix Expression

Operator between operands.

Example:

```
A + B
```

Postfix Expression

Operator after operands.

Example:

```
AB+
```

Why Convert?

Postfix evaluation is easier for computer.

Algorithm for Infix to Postfix

```
1. Scan expression left to right
2. If operand → add to output
3. If '(' → push to stack
4. If operator:
      pop higher precedence operators
```

```
      then push current operator
5. If ')' → pop until '('
6. Pop remaining operators
```

Example

Expression

```
A + B * C
```

Conversion

```
ABC*+
```

Because:

Multiplication has higher priority.

Operator Precedence

Operator	Priority
^	Highest
* /	Medium
+ -	Low

---

2. Evaluation of Postfix Expression

Example

```
23*5+
```

Meaning:

```
(2*3)+5
```

Algorithm

```
1. Scan expression
2. If operand → push
3. If operator:
       pop two operands
       perform operation
       push result
4. Final value = answer
```

Example

```
23*5+
```

```
2*3 = 6
6+5 = 11
```

Answer:

```
11
```

---

3. Recursion

Recursion means:

function calling itself.

Example

Factorial:

```
int fact(int n)
{
    if(n==0)
        return 1;

    return n * fact(n-1);
}
```

How Stack is Used in Recursion?

Each recursive call is stored in stack memory.

This is called:

```
Call Stack
```

Advantages of Recursion

- Simple code
- Easy for tree problems

Disadvantages

- Extra memory
- Stack overflow possible

---

2. QUEUE

Introduction

Queue is a linear data structure.

Insertion occurs from:

```
REAR
```

Deletion occurs from:

```
FRONT
```

FIFO Principle

```
First In First Out
```

Meaning:
Jo element pehle insert hoga,
wo pehle remove hoga.

Real Life Example

- Ticket counter
- Printer queue
- CPU scheduling

Queue Operations

1. ENQUEUE

Insert element.

---

2. DEQUEUE

Delete element.

---

3. PEEK

View front element.

---

4. isEmpty()

Check empty queue.

---

5. isFull()

Check full queue.

Queue as ADT

Operations:

- enqueue()
- dequeue()
- peek()
- isEmpty()
- isFull()

Queue Implementation

1. Array

2. Linked List

Circular Queue

In normal queue:

memory wastage occurs.

Circular queue connects last position to first position.

Advantage

Efficient memory utilization.

Condition

Queue Full

```
(front == rear + 1)
```

Applications

- CPU scheduling
- Keyboard buffering
- Traffic systems

DQUEUE (Double Ended Queue)

Insertion and deletion possible from both ends.

Types of DQUEUE

1. Input Restricted DQUEUE

Insertion only at one end.

---

2. Output Restricted DQUEUE

Deletion only at one end.

Priority Queue

Each element has priority.

Higher priority element served first.

Example

Hospital emergency system.

Queue Simulation

Queue behavior can be simulated using:

- Arrays
- Linked lists

Used in:

- Operating systems

- Networking
- Printer scheduling

Applications of Queue

1. CPU Scheduling

2. Printer Queue

3. BFS Traversal

4. Ticket Reservation

5. Data Buffers

Difference Between Stack and Queue

Stack	Queue
LIFO	FIFO
One end operation	Two end operation
PUSH/POP	ENQUEUE/DEQUEUE
Top used	Front and Rear used
Recursion	Scheduling