NAMES: IDUFASHE Angela Mars

REG NUMBER: 223015288

Assignment two

STACK AND QUEUE DATA STRUCTURE

1.STACK DATA STRUCTURE

What is a Stack?

It's a linear data structure that follows the Last-In, First-Out (LIFO) principle. This means the last element added to the stack is the first one to be removed.

Forexample Imagine a stack of plates. You can only add or remove plates from the top. This is the basic idea behind a stack data structure.

Here's a breakdown;

*   LIFO: Think of it like a pile of books. You put the last book on top, and when you take a book off, you take the one from the top.

*   Operations: The primary operations on a stack are:

    *   Push: Adds an element to the top of the stack.

    *   Pop: Removes the top element from the stack.

    *   Peek (or Top): Allows you to view the top element without removing it.

    *   IsEmpty: Checks if the stack is empty.

*   Implementation: Stacks can be implemented using arrays or linked lists. Arrays provide a simple implementation, while linked lists offer dynamic sizing.

Real-World Examples:

    *   Function Calls: When a function is called, it's added to the stack. When the function is finished, it's removed.

* Undo/Redo: Many applications use stacks to implement undo and redo functionality.

* Expression Evaluation: Stacks are used to evaluate arithmetic expressions.

Stack Applications and Advantages

Advantages:

* Simple to Implement: Stacks are relatively easy to implement and understand.

* Efficient: Push and pop operations are typically very fast (O(1) time complexity).

* Memory Management: Stacks can be helpful in managing memory in certain situations.

Applications:

* Expression Evaluation: Stacks are used to convert infix expressions to postfix or prefix notation and evaluate them.

* Backtracking: In algorithms like depth-first search (DFS), stacks are used to keep track of visited nodes.

* Syntax Parsing: Compilers and interpreters use stacks to parse programming language syntax.

* Browser History: Web browsers use stacks to manage the history of visited web pages (back and forward buttons).

* Recursion: Stacks are fundamental to the implementation of recursive functions. Each recursive call adds a new frame to the stack.

* Considerations:

* Fixed Size (Array Implementation): If you use an array to implement a stack, you might run into a "stack overflow" if you try to add more elements than the array can hold.

* Limited Access: You can only access the top element of a stack. You can't directly access elements in the middle.

TYPES OF STACK DATA STRUCTURE

1. Array-Based Stack:

* Implementation: Uses a fixed-size array to store stack elements.

* Pros: Simple to implement and efficient for basic operations.

* Cons: Limited size (requires predefining the maximum capacity).

2. Linked List-Based Stack:

* Implementation: Uses a linked list (nodes) to store elements. Each node contains data and a pointer to the next node.

* Pros: Dynamic size (can grow or shrink as needed), no fixed capacity.

* Cons: Slightly more complex to implement than array-based stacks due to managing nodes and pointers.

3. Static Stack:

* Implementation: A stack whose size is fixed at compile time.

  * Pros: Simple and efficient if the maximum size is known beforehand.

  * Cons: Inflexible; cannot resize during runtime.

4. Dynamic Stack:

* Implementation: A stack that can grow or shrink during runtime.

  * Pros: Flexible; adjusts to the number of elements.

  * Cons: Requires memory management to resize the stack.

In conclusion, stacks are a fundamental data structure with a wide range of applications. Their LIFO nature makes them ideal for managing data where the order of operations is crucial.

2.QUEUE DATA STRUCTURE

What is a Queue?

It's a linear data structure that follows the First-In, First-Out (FIFO) principle. This means the first element added to the queue is the first one to be removed.

For example

Think of a queue as a line of people waiting to buy movie tickets. The first person in line gets served first. That's the basic concept behind a queue data structure.

Here's a breakdown:

* FIFO: Imagine a line at a grocery store. The first person in line is the first to be served.

* Operations: The primary operations on a queue are:

  * Enqueue: Adds an element to the rear (end) of the queue.

  * Dequeue: Removes the element from the front of the queue.

  * Peek (or Front): Allows you to view the front element without removing it.

  * IsEmpty: Checks if the queue is empty.

Implementation: Queues can be implemented using arrays or linked lists. Arrays can be efficient, but you might need to handle circular queues to avoid wasting space. Linked lists offer dynamic sizing.

Real-World Examples:

- Print Queue: When you send a document to a printer, it goes into a queue to be printed.
- Call Center: Customer calls are placed in a queue to be answered by the next available agent.
- Operating System Task Scheduling: Operating systems often use queues to manage processes waiting to use the CPU.

Queue Applications and Advantages

Advantages:

- Fairness: FIFO ensures elements are processed in the order they were added, making it fair.
- Order Preservation: Queues maintain the order of elements.
- Simple to Understand: The concept of a queue is easy to grasp.

- Applications:
  - Breadth-First Search (BFS): BFS algorithms use queues to explore nodes level by level.
  - Task Scheduling: Operating systems use queues to manage processes.
  - Message Queues: Used in distributed systems for asynchronous communication.
  - Buffer Management: Queues can buffer data between processes.
  - Simulations: Queues are used in simulations to model real-world scenarios.

Types of Queues:

- Simple Queue: Standard FIFO queue.
- Circular Queue: Uses a fixed-size array and reuses the space by wrapping around.
- Priority Queue: Elements are assigned priorities, and the highest priority element is dequeued first.
- Double-Ended Queue (Deque): Elements can be added or removed from both ends.

In conclusion, queues are a fundamental data structure that's essential for managing data in a FIFO manner. They are used in a wide variety of applications where fairness and order of processing are important.