TOPIC

DATA STRUCTURE AND ALGORITHM

Content

BY DO NGOC TRUNG
BH00901

Part 1

Overview of Student Management System

- A Student Management System (SMS) is a comprehensive software application designed to manage student data effectively.

Key Aspects:

- Centralized database for storing all student information securely.
- User-friendly interface for seamless navigation and data access.
- Supports various functionalities tailored to the needs of educators, students, and administrators.

Part 2: Abstract Data Type (ADT) Overview

- Definition of ADT:
  - An Abstract Data Type (ADT) is a theoretical concept used in computer science to define a data type by its behavior from the user’s perspective, emphasizing operations rather than implementations.

- Importance of ADTs:
  - Allows developers to focus on the functionality of data structures without being concerned with the implementation details.
  - Promotes clean, modular, and reusable code.

Objectives of Student Management System

- Centralized Data Management: Provides a single platform for storing and accessing information.
- Improved Communication: Enhances interaction among students, parents, and educators.
- Enhanced Efficiency: Streamlines administrative tasks to allow more focus on education.

Common Examples of ADTs

Stack:

Data structure that operates on a Last-In-First-Out (LIFO) principle.

Queue:

Data structure that operates on a First-In-First-Out (FIFO) principle.

List:

Collection of items that can be accessed by their index, with operations for adding, removing, and accessing elements.

Introduction to Stack

Definition:

A Stack is an ADT that follows the Last-In-First-Out (LIFO) principle, where the most recently added element is the first to be removed.

Visual Representation:

Include a diagram showing a stack with operations illustrated (push, pop, peek).

Stack Operations:

Push: Adds an item to the top of the stack.
Pop: Removes the item from the top of the stack and returns it.
Peek: Returns the item at the top without removing it.
isEmpty: Checks if the stack is empty, returning a boolean value

Applications of Stack in SMS

Tracking Changes:

Keeps a history of modifications made to student records (e.g., grades, course enrollments), allowing administrators to audit changes.

Implementing Undo Functionality:

Allows users to revert recent actions (e.g., un-enrolling a student from a course) through a simple pop operation.

Example Use Case:

In an SMS, a stack can be used to manage changes to student grades, where each change can be pushed onto the stack and popped off when an undo is required.

Queue Operations

Definition

- A Queue is an ADT that operates on a First-In-First-Out (FIFO) basis, where the first element added is the first one to be removed.
- Visual Representation: Include a diagram depicting a queue with enqueue and dequeue operations.

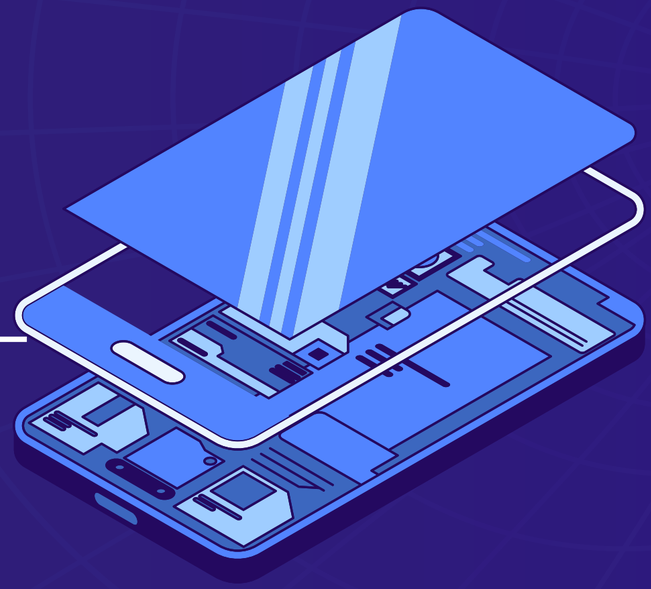
Introduction to Queue

Queue Operations

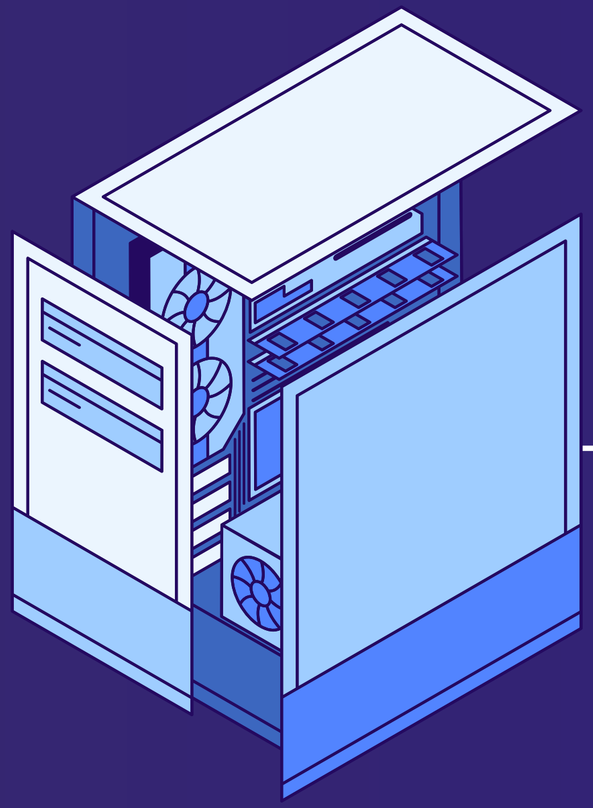
- Enqueue: Adds an item to the back of the queue.
- Dequeue: Removes the item from the front of the queue and returns it.
- Peek: Returns the item at the front without removing it.
- isEmpty: Checks if the queue is empty.

Applications of Queue in SMS

- Handling Requests: Manages student requests (e.g., course registration) in the order they are received, ensuring fairness.
- Event Scheduling: Organizes notifications or events based on priority and order of occurrence, such as sending reminders for upcoming exams or deadlines.
- Example Use Case: A queue can manage course registration requests, processing each request in the order it was submitted.



Part 3: Comparing Stack and Queue



Functional Differences Between Stack and Queue

- Stack Characteristics:
  - Order: Last-In-First-Out (LIFO)
  - Use Case: Ideal for situations requiring reverse order processing, such as backtracking in course prerequisites.
- Queue Characteristics:
  - Order: First-In-First-Out (FIFO)
  - Use Case: Best for processing tasks in the order they arrive, like handling student inquiries or support requests.

Use Cases Comparison

- Stack Use Cases: Backtracking Algorithms: Helps navigate course prerequisites in reverse order, ensuring all conditions are met.
- Queue Use Cases:
- Managing Enrollment Processes: Ensures that student enrollments are handled in the order they are received, enhancing organization and fairness.

Part 4: Ways to Implement Stack and Queue

Array-Based Implementation of Stack

- Description: Utilizes a fixed-size array to store elements, providing direct access through indices.
- Pros: Fast access times due to direct index-based retrieval.
- Cons: Limited size; resizing requires copying to a new array, which can be inefficient.

Linked List-Based Implementation of Stack

- Description: Employs a dynamic linked list structure where each node points to the next, allowing for flexible sizing.
- Pros: Grows dynamically as elements are added, eliminating size constraints.
- Cons: Slightly higher memory overhead due to additional pointers in each node.

Circular Queue Implementation

Description:

A circular queue connects the last position back to the first, optimizing space usage in a fixed-size array.

Use Case:

Particularly useful in scenarios where the queue needs to wrap around, such as managing a fixed number of processes or requests.

Visual Representation:

Include a diagram illustrating how a circular queue operates.23

Part 5: Sorting Algorithms

Introduction to Sorting Algorithms

Definition:

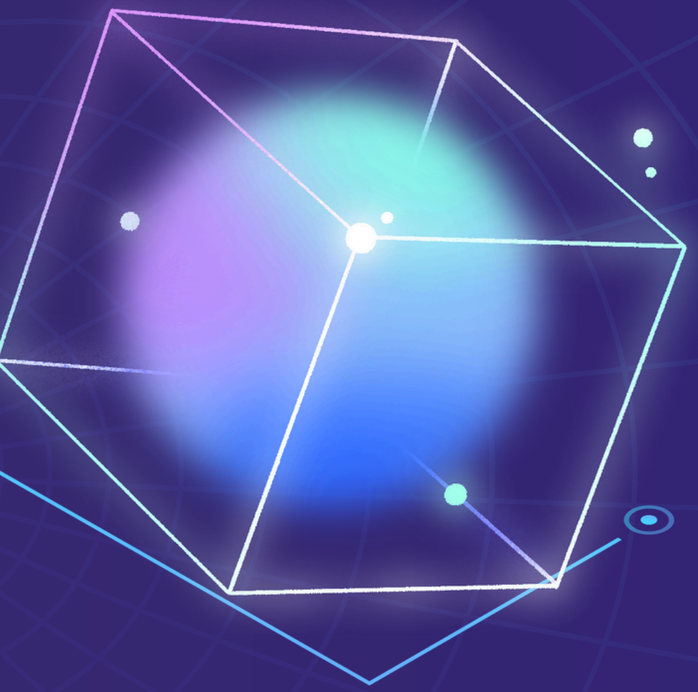
Sorting Algorithms are methods used to arrange elements in a list or array in a specified order (e.g., ascending or descending).

Importance in SMS

Efficient sorting is crucial for organizing student data, such as sorting by grades for report cards and organizing attendance records chronologically.

OVERVIEW OF COMMON SORTING ALGORITHMS

Bubble Sort:	Selection Sort:	Quick Sort:
A simple comparison-based algorithm that repeatedly steps through the list, swapping adjacent elements if they are in the wrong order. Not efficient for large datasets.	Selects the smallest (or largest) element from the unsorted portion of the array and swaps it with the first unsorted element. Suitable for small lists but inefficient for larger datasets.	A highly efficient, divide-and-conquer algorithm that selects a pivot and partitions the array, recursively applying the same logic. Best for large datasets.



PART 6: APPLICATIONS OF SORTING ALGORITHMS IN SMS

Applications of Bubble Sort

- Use Case: Suitable for small classes or when dealing with a limited number of student records due to its simplicity.
- Example: Arranging a small group of students' scores in ascending order for a simple display.
- Visual Representation: Include a small dataset example illustrating the bubble sort process.

Applications of Selection Sort

- Use Case: The best choice for sorting large datasets, such as full class rosters, attendance logs, or comprehensive grade reports.
- Example: Rapidly sorting an entire school's student records by GPA to identify top-performing students.
- Visual Representation: Include a visual flowchart illustrating the quick sort process on a large dataset.

Applications of Quick Sort

THANK YOU!

+123-456-7890

dotrung422004@gmail.com

www.reallygreatsite.com