**Exercise 5: Task Management System**

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

In that case, a linked list data structure would be a suitable choice for implementing the task management system. Here's why:

1. Efficient insertion and deletion: Linked lists allow for efficient insertion and deletion of tasks at any position, which is useful when tasks need to be added or removed dynamically.

2. Dynamic size: Linked lists can grow or shrink dynamically, making them suitable for applications where the number of tasks changes frequently.

3. Good memory usage: Linked lists can make good use of memory, especially when dealing with large datasets.

Here's a possible implementation of the task management system using a linked list:

public class TaskManagementSystem {

private Node head;

public void addTask(Task task) {

Node newNode = new Node(task);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public void deleteTask(int taskId) {

if (head == null) {

return;

}

if (head.task.getTaskId() == taskId) {

head = head.next;

return;

}

Node current = head;

while (current.next != null) {

if (current.next.task.getTaskId() == taskId) {

current.next = current.next.next;

return;

}

current = current.next;

}

}

public void traverseTasks() {

Node current = head;

while (current != null) {

System.out.println(current.task);

current = current.next;

}

}

private class Node {

Task task;

Node next;

public Node(Task task) {

this.task = task;

this.next = null;

}

}

}

This implementation provides methods for adding, deleting, and traversing tasks efficiently. The time complexity of these operations is O(n), where n is the number of tasks in the system.

//Java code

public class Task {

private int taskId;

private String taskName;

private String status;

public Task(int taskId, String taskName, String status) {

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

// Getters and setters

}

public class SinglyLinkedList {

private Node head;

public void addTask(Task task) {

Node newNode = new Node(task);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public Task searchTask(int taskId) {

Node current = head;

while (current != null) {

if (current.task.getTaskId() == taskId) {

return current.task;

}

current = current.next;

}

return null;

}

public void traverseTasks() {

Node current = head;

while (current != null) {

System.out.println(current.task);

current = current.next;

}

}

public void deleteTask(int taskId) {

if (head == null) {

return;

}

if (head.task.getTaskId() == taskId) {

head = head.next;

return;

}

Node current = head;

while (current.next != null) {

if (current.next.task.getTaskId() == taskId) {

current.next = current.next.next;

return;

}

current = current.next;

}

}

private class Node {

Task task;

Node next;

public Node(Task task) {

this.task = task;

this.next = null;

}

}

}

Analysis

Analyze the time complexity of each operation.

- Add Task: O(n) - linear time complexity

- Search Task: O(n) - linear time complexity

- Traverse Tasks: O(n) - linear time complexity

- Delete Task: O(n) - linear time complexity

Discuss the advantages of linked lists over arrays for dynamic data

Advantages of Linked Lists over Arrays

- Dynamic size: Linked lists can grow or shrink dynamically, making them suitable for applications with changing data.

- Efficient insertion and deletion: Linked lists allow for efficient insertion and deletion of elements at any position.

- Good memory usage: Linked lists can make good use of memory, especially when dealing with large datasets.

However, linked lists also have some disadvantages, such as:

- Slow search: Searching for an element in a linked list can be slow.

- More complex implementation: Linked lists are more complex to implement than arrays.