**Design and Complexity Analysis for Restaurant Reservation System**

The Restaurant Reservation System is designed to efficiently manage reservations, cancellations, and waitlists in a restaurant. The choice of appropriate data structures and algorithms is crucial for ensuring the system's performance and scalability.

Data Structures

* Binary Search Tree (BST) for Table Management

A Binary Search Tree is chosen to manage tables efficiently. Each node in the tree represents a table, and the tree is structured based on table capacity. The choice of BST allows for quick lookups of the smallest available table that can accommodate a given party size.

* Insertion

- The **insertTable** method in **TableNode** class ensures that tables are inserted in a way that maintains the ordering of capacities in the tree.

- Time Complexity:O(log n) on average, where n is the number of tables.

* Table Size Lookup

-The **findAvailableTable** and **findSmallestTable** methods in the **RestaurantReservationSystem** class efficiently find the smallest available table for a given party size.

- Time Complexity: O(log n) on average, where n is the number of tables.

* Hash Table for Reservation Lookup

A hash table is chosen to allow quick lookup of reservations based on customer names. The **ReservationManagementTable** class manages the reservations using a hash table.

* Hash Function

- The hash method in **ReservationManagementTable** class provides an index for efficient retrieval.

- Time Complexity:O(1) on average.

* Insertion and Update

- The **put** method in **ReservationManagementTable** ensures that reservations are efficiently inserted or updated.

- Time Complexity: O(1) on average.

* Reservation Lookup

- The **get** method in **ReservationManagementTable** quickly retrieves a reservation based on the customer's name.

- Time Complexity:O(1) on average.

* Queue for Waitlist

A simple queue (RestQueue) is employed to manage the waitlist. This ensures a first-come-first-served order for customers waiting for a table.

* Enqueue

- The `enqueue` method in `RestQueue` efficiently adds a customer to the end of the queue.

- Time Complexity:O(1).

* Dequeue

- The **dequeue** method in `RestQueue` efficiently removes and returns the customer at the front of the queue.

- Time Complexity:O(1).

Algorithms

* Rescheduling Algorithm

When rescheduling a reservation, the system checks for the availability of tables to accommodate the updated reservation. The algorithm finds the smallest available table for the new party size.

* Rescheduling Algorithm

1. Retrieve the existing reservation.

2. Find the smallest available table using the `findAvailableTable` algorithm.

3. If an available table is found, update the reservation with the new details.

Time Complexity

- Worst-Case:O(log n), where n is the number of tables in the BST.

The choice of a Binary Search Tree, Hash Table, and Queue for managing tables, reservations, and waitlists, respectively, ensures efficient and scalable operations for the Restaurant Reservation System. The selected data structures and algorithms provide quick lookups, insertions, and updates, meeting the requirements of a real-world restaurant environment.