Assignment 06

Efficient Truck Management Using Binomial Heap in Logistics

1. Suitable Data Structure to Solve the Problem

In this implementation, a **Binomial Heap** is used as the primary data structure to manage the logistics of a trucking system. A **Binomial Heap** is an efficient priority queue that supports fast insertion, merging, and deletion operations. This choice is particularly useful in logistics, where trucks with different urgency levels must be efficiently managed and dispatched based on their priority.

The Binomial Heap consists of **binomial trees**, which provide logarithmic height and allow quick merging of heaps. The heap property is maintained by ensuring that a parent node has a lower urgency value than its children, prioritizing the most urgent trucks.

2. Operations and Their Complexity

The **LogisticCompany** class implements the following operations:

Insertion

- **Operation**: `insertTruck(int truckId, int distance, int urgency)`
- **Description**: Inserts a new truck into the heap by creating a single-node heap and merging it with the existing heap.
- **Time Complexity**: O(logn)O(\log n)
- **Space Complexity**: O(1)O(1)

Finding the Minimum Priority Truck

- **Operation**: `getMinPriorityTruck()`
- **Description**: Scans the root list of the binomial heap to find the truck with the smallest urgency value.
- **Time Complexity**: O(logn)O(\log n)
- **Space Complexity**: O(1)O(1)

Extracting the Minimum Priority Truck

- **Operation**: `extractMin()`
- **Description**: Removes the truck with the smallest urgency value, reorders its child nodes, and merges them back into the heap.

- **Time Complexity**: O(logn)O(\log n)
- **Space Complexity**: O(logn)O(\log n) (for recursive merging)

Merging Two Heaps

- **Operation**: `unionHeap(truck *h1, truck *h2)`
- **Description**: Merges two binomial heaps into one by linking binomial trees of the same degree while maintaining the heap property.
- **Time Complexity**: O(logn)O(\log n)
- **Space Complexity**: O(1)O(1)

Decrease Key (Updating Urgency Level)

- **Operation**: `decreaseKey(int targetKey, int newKey)`
- **Description**: Decreases the urgency value of a specific truck and ensures the heap property is maintained by swapping with its parent if necessary.
- **Time Complexity**: O(logn)O(\log n)
- **Space Complexity**: O(1)O(1)

Deleting a Truck

- **Operation**: `deleteKey(int targetKey)`
- **Description**: Decreases the truck's urgency to negative infinity and extracts it from the heap.
- **Time Complexity**: O(logn)O(\log n)
- **Space Complexity**: O(logn)O(\log n)

Searching for a Truck

- **Operation**: `findNode(int targetKey)`
- **Description**: Recursively searches for a truck with a specific urgency value.
- **Time Complexity**: O(n)O(n)
- **Space Complexity**: O(1)O(1)

3. Conclusion

The **Binomial Heap** is a suitable data structure for managing a fleet of trucks with different urgency levels. Its **logarithmic time complexity** for insertion, extraction, and merging makes it an **efficient priority queue** for dynamic scheduling problems.

The implementation ensures **optimal truck dispatching**, reducing delays and improving logistics efficiency. The provided operations allow for efficient truck management, enabling insertion, searching, and removal of trucks based on priority levels.

Overall, the use of a binomial heap significantly improves the performance of truck dispatching and priority management in logistics systems, making it a well-suited choice for this application.
