Priority Queue Implementation and Applications

Design Choices

1. **Data Structure:** A binary heap was chosen to represent the priority queue due to its logarithmic time complexity for insertion, extraction, and critical modification operations. A list was used for implementation, as heap operations such as parent-child relationship management are efficient with arrays.
2. **Task Representation:** The Task class includes task_id, priority, arrival_time, and deadline attributes. The priority determines task scheduling, and the comparison operators (<) enable tasks to be sorted according to priority.
3. **Heap Property:** A min-heap was chosen, meaning tasks with the lowest priority value are extracted first. This suits scheduling algorithms prioritizing tasks with the shortest deadline or highest urgency.

Time Complexity Analysis

Insert Operation (insert):

- When inserting a task, we add it to the end of the heap and then restore the heap property by bubbling up the task. The height of a binary heap is O(logn), and the bubbling-up process takes O(logn).
- **Time Complexity**: O(logn), where n is the number of tasks.

Extract Min Operation (extract_min):

- The task with the lowest priority is always at the root (index 0). When removing it, we replace it with the last task in the heap and restore the heap property by bubbling down. Like insertion, this takes O(logn).
- **Time Complexity**: O(logn).

Decrease/Increase Key (decrease_key and increase_key):

Both operations modify the priority of a task and either bubble it up (for decrease) or bubble it down (for increase) to restore the heap property. These operations also take O(logn).

- **Time Complexity**: O(logn).

Is Empty (is_empty):

- Checking if the priority queue is empty takes constant time, as it only checks the length of the heap list.
- **Time Complexity**: O(1).

Scheduling Result

In the provided simulation, tasks are scheduled based on their priority. Tasks with the lowest priority values are processed first. This simple scheduling algorithm can be extended to more complex systems that consider additional factors like deadlines or arrival times.