Solution to Problem P5.2.4: Correctness and Runtime Complexity of PersonEnters

Problem Overview

We are tasked with explaining why the `PersonEnters` algorithm is correct and has a runtime complexity of $\Theta(\log M)$, where M is the maximum capacity of the museum.

Solution Explanation

1. Correctness of the Algorithm

- **Min-Heap Representation:** The min-heap (wla) holds the leave times of all the visitors currently in the museum. Since we add each visitor's leave time to the heap when they enter the museum, and we remove visitors whose leave times are less than the current time (i.e., visitors who have already left), the heap always contains only the leave times of visitors still inside the museum.
- **Invariant:** At any given time, the heap correctly maintains the set of leave times of all visitors who are still in the museum. The size of the heap, therefore, directly reflects the number of visitors inside the museum at that moment.
- **Visitor Handling:** For each new visitor i entering at time x_i, we:
  a) Remove any visitors whose leave time is less than x_i, ensuring that only the visitors who are still in the museum remain in the heap.
  b) Add the new visitor's leave time y_i to the heap.
  c) The number of visitors in the museum at time x_i is the size of the heap after performing these steps.

2. Time Complexity Argument

We need to show that the operation `PersonEnters(x_i, y_i)` runs in $\Theta(\log M)$, where M is the maximum occupancy of the museum.

- **Heap Operations:** The key operations in the algorithm involve maintaining the heap:

  a) **Add Operation:** Inserting a new leave time y_i into the min-heap takes $O(\log M)$ time. The heap contains at most M elements (since M is the maximum capacity of the museum).

  b) **Remove (DelMin) Operations:** For each visitor entering the museum, we remove all visitors who have already left by repeatedly removing the minimum element (leave time) from the heap. Each removal takes $O(\log M)$ time, and we remove only the visitors whose leave time is less than x_i.

- **Worst-Case Analysis:** After the i-th person enters, we have performed i insertions and at most $i-1$ removals on a heap with at most M elements. Each of these heap operations (insertion and deletion) takes $O(\log M)$ time.

  Hence, the total cost for i visitors is:

  $$O(i \log M)$$

  - The number of visitors in the museum at any given time is bounded by M, so the heap operations take $O(\log M)$ per visitor.

- **Amortized Cost per Person:** To calculate the amortized cost, we spread the total cost over the number of people i. The total cost for i people is $O(i \log M)$, so the amortized cost per person is:

  $$O(\log M)$$

  Therefore, the amortized time complexity for each person entering the museum is $O(\log M)$, which meets the problem's requirement.

Conclusion

- **Correctness:** The algorithm is correct because it efficiently maintains the set of visitors in the museum at any given time using a min-heap that stores the leave times.

- **Time Complexity:** The total time complexity for i visitors is $O(i \log M)$, and the amortized time per person is $O(\log M)$, which matches the required $\Theta(\log M)$ time complexity.

Thus, the `PersonEnters` algorithm is both correct and has the desired runtime complexity of $\Theta(\log M)$.