Solution to Problem P5.2.3: PersonEnters Operation in $O(\log M)$

Problem Restatement

We are tasked with designing a data structure that supports an operation `PersonEnters(x_i, y_i)`. This operation should:

- Compute the number of visitors in the museum at time x_i when person i enters.
- Operate in at most $O(\log M)$, where M is the maximum number of visitors allowed in the museum.

Solution Overview

The solution proposes using a **min-heap** (`wla`) to efficiently manage visitors' leave times. The operation `PersonEnters(x_i, y_i)` performs similar steps to the for-loop in the `MaxVisitors` algorithm, which includes:

1. **Removing visitors who have left:** Before adding the new visitor, we need to remove all visitors who have already left the museum (i.e., whose leave times are earlier than x_i).
2. **Adding the new visitor:** Once the earlier visitors have been removed, the new visitor's leave time y_i is added to the heap.
3. **Returning the current occupancy:** The size of the heap at this point gives the current number of visitors in the museum.

Explanation of the Solution

1. Using the Min-Heap

- The min-heap (`wla`) stores the leave times of visitors currently inside the museum.

- The property of the min-heap ensures that the smallest leave time is always at the top. This allows us to efficiently remove visitors who have already left by repeatedly calling `DELMIN(wla)` until the minimum leave time in the heap is greater than or equal to x_i (the current person's entry time).

2. Steps in PersonEnters(x_i, y_i)

- **Step 1:** Remove all visitors whose leave times are less than x_i. This is done by repeatedly extracting the minimum element from the heap (`DELMIN(wla)`).
- **Step 2:** Add the new visitor's leave time y_i to the heap (`ADD(wla, y_i)`).
- **Step 3:** Return the size of the heap (`SIZE(wla)`) to indicate how many visitors are currently inside the museum.

3. Why the Algorithm Works in $O(\log M)$

- **Heap Operations:** Both insertion and deletion in a heap take $O(\log M)$, where M is the maximum capacity of the museum (i.e., the maximum number of visitors that can be inside the museum at any time).
- **Step 1 (Removing Visitors):** The min-heap allows us to efficiently remove visitors whose leave times are earlier than x_i. This takes $O(\log M)$ for each removal, and since we only remove visitors who have already left, the number of removals will be bounded by M.
- **Step 2 (Adding New Visitor):** Adding a new leave time to the heap takes $O(\log M)$ time.
- **Step 3 (Returning Size):** Returning the size of the heap takes constant time $O(1)$.

Therefore, the entire `PersonEnters(x_i, y_i)` operation runs in $O(\log M)$.

Example

Consider a museum with the following sequence of visitors:

- Person 1 enters at time 1 and leaves at time 5.
- Person 2 enters at time 2 and leaves at time 6.
- Person 3 enters at time 4 and leaves at time 8.
- Person 4 enters at time 7 and leaves at time 9.

Let's walk through the operations:

- **PersonEnters(1, 5)**:
  - No visitors to remove (heap is empty).
  - Add leave time 5.
  - The heap contains [5].
  - The size of the heap is 1.
- **PersonEnters(2, 6)**:
  - No visitors to remove (leave time 5 ¿ 2).
  - Add leave time 6.
  - The heap contains [5, 6].
  - The size of the heap is 2.
- **PersonEnters(4, 8)**:
  - No visitors to remove (leave time 5 ¿ 4).
  - Add leave time 8.
  - The heap contains [5, 6, 8].
  - The size of the heap is 3.
- **PersonEnters(7, 9)**:
  - Remove Person 1 (leave time 5 ¡ 7) and Person 2 (leave time 6 ¡ 7).
  - Add leave time 9.
  - The heap contains [8, 9].
  - The size of the heap is 2.

Conclusion

The min-heap efficiently maintains the leave times of visitors in the museum, allowing us to remove visitors who have already left and add new visitors, all within $O(\log M)$ time. This satisfies the problem requirement for the `PersonEnters(x_i, y_i)` operation to run in $O(\log M)$. By returning the size of the heap, we can determine the current number of visitors in the museum at any given time.