Explanation of Algorithm MaxVisitors

Correctness of the Algorithm

The algorithm **MaxVisitors** is correct because it properly maintains and updates the set of visitors currently inside the building and uses a min-heap to ensure that visitors are efficiently removed once they have left. The following points argue for its correctness:

1. **Visitor Tracking with Min-Heap:**
    - The min-heap *wla* stores the **leave times** of visitors currently inside the building. This ensures that only those visitors who have not yet left are being tracked.
    - Each time a new visitor enters, the algorithm checks the heap to remove any visitors whose leave time is before the new visitor's entry time. This is done using the `DELMIN(wla)` operation, which removes the visitor with the smallest leave time from the heap. This guarantees that only current visitors are considered for overlap at any given time.
2. **Efficient Removal of Left Visitors:**
    - By using a **min-heap**, the algorithm efficiently tracks the smallest leave time, ensuring that visitors are removed from the heap in the correct order. The `DELMIN(wla)` operation guarantees that the earliest leaving visitor is removed before processing the current visitor.
3. **Accurate Tracking of Maximum Overlap:**
    - After updating the heap for the current visitor, the algorithm updates the maximum number of visitors present at any time by checking the size of the heap. The heap's size at any point represents the number of visitors currently inside the building. The algorithm keeps track of the maximum size of the heap, which directly corresponds to the maximum number of overlapping visitors.
4. **Termination and Final Result:**

- The algorithm processes all visitors and their entry/leave times once, ensuring that it terminates after visiting each element in L. After processing all visitors, the variable m holds the maximum number of visitors present at any time.

Because the algorithm ensures that visitors who have left are removed from the heap, and it keeps track of the current visitors at all times, it correctly calculates the maximum number of overlapping visitors.

Time Complexity of the Algorithm

To show that the algorithm runs in $O(N \log N)$, where N is the number of visitors, we need to analyze the operations and their associated costs:

1. **Processing Each Visitor:**
   - The algorithm iterates through the list L, which contains N visitor intervals. For each visitor (x, y), the algorithm performs several operations:
     - It removes any visitors from the heap who have already left, which involves repeated calls to `DELMIN(wla)` while the smallest leave time is less than x. Each removal operation takes $O(\log N)$ time, as it adjusts the heap.
     - It adds the current visitor's leave time to the heap using `ADD(wla, y)`, which also takes $O(\log N)$ time.
2. **Number of Heap Operations:**
   - Each visitor enters and eventually leaves the building, which corresponds to exactly one insertion (`ADD`) and one removal (`DELMIN`) in the min-heap. Thus, for each visitor, at most two heap operations occur.
3. **Heap Size:**
   - Since the heap stores the leave times of all visitors who are currently inside the building, the heap size is at most N, where N is the total number of visitors. Insertion and removal from a min-heap both take $O(\log N)$ time in the worst case.
4. **Overall Complexity:**
   - For each visitor, the algorithm performs a constant number of heap operations, and each heap operation takes $O(\log N)$. Therefore, the total time for processing all N visitors is $O(N \log N)$.

Conclusion

- **Correctness:** The algorithm correctly calculates the maximum number of overlapping visitors at any given time by using a min-heap to track visitors who are still inside, and it efficiently updates the number of visitors based on their leave times.
- **Time Complexity:** The algorithm processes each visitor in $O(\log N)$ time due to the heap operations, and since there are N visitors, the total runtime complexity is $O(N \log N)$.

Thus, the algorithm is both correct and has a runtime complexity of $O(N \log N)$.