Documentation

Problem Description:

The Skyline Problem involves determining the outer contour of a city's skyline formed by a collection of rectangular buildings. Each building is defined by three values: the left x-coordinate (where the building starts), the right x-coordinate (where the building ends), and the height of the building. The goal is to find the key points where the height of the skyline changes and return them in an ordered list, representing the silhouette formed by the buildings.

The skyline should be represented as a series of key points, each indicating a change in height at a particular x-coordinate. The last key point in the result always has a height of 0, marking the end of the skyline after all buildings are processed. Consecutive points in the output cannot have the same height, meaning that overlapping buildings of equal height should be merged into a single segment.

Key Concepts:

The key points of the skyline are the points where the height of the skyline changes. These changes occur at the start and end of buildings. Hence, for each building, two events are important:

1. The starting point where the building begins and its height becomes part of the skyline.
2. The ending point where the building ends and its height no longer contributes to the skyline.

The challenge lies in efficiently determining when the height changes and ensuring that overlapping buildings of different heights are correctly accounted for.

Algorithm Approach:

The most efficient approach to solving this problem is to use a sweep line algorithm with a priority queue (max-heap). This approach processes the buildings as a series of events and maintains a record of active buildings at each point in time.

1. Event Creation:

- For each building, two key events are generated:
  - A start event at the left x-coordinate where the building starts. This event contributes to the building's height to the skyline.
  - An end event at the right x-coordinate where the building ends. At this point, the building's height is removed from the skyline.

2. Sorting Events:

- These events are sorted primarily by x-coordinate. If two events share the same x-coordinate, start events are processed before end events to ensure that new buildings are added before old ones are removed. This ensures the skyline is correctly updated when multiple buildings overlap.

3. Using a Max-Heap:

- A max-heap (priority queue) is used to store the heights of the buildings that are currently part of the skyline. The heap helps keep track of the maximum height at any given x-coordinate. When a building starts, its height is added to the heap; when a building ends, its height is removed. The maximum value in the heap represents the current height of the skyline.

4. Handling Height Changes:

- As the algorithm processes events from left to right, it checks for changes in the maximum height of the heap. If the maximum height changes at a given x-coordinate, a key point is added to the result, indicating the change in the skyline. For example, when a taller building starts, the skyline rises, and when a building ends or a shorter building takes over, the skyline drops.

5. Termination:

- After all the events are processed, the last key point must have a height of 0, marking the termination of the skyline where the rightmost building ends.

Efficiency:

- This approach ensures that the problem is solved efficiently with a time complexity of O(n log n), where n is the number of buildings. The events are sorted in O(n log n), and the priority queue operations (insertions and deletions) also occur in O(log n), ensuring that the algorithm can handle large inputs efficiently. This makes it suitable for the input constraints, where up to 10,000 buildings may need to be processed.

Edge Cases:

- *The algorithm accounts for several edge cases, such as:*
  - Multiple buildings starting and ending at the same x-coordinate, requiring careful handling to ensure proper updates to the skyline.
  - Ensuring that there are no consecutive horizontal lines of the same height in the final result by merging such segments into a single key point.
  - Buildings that overlap or are nested within each other, requiring correct updates to the skyline as one building ends and another continues.

Example:

- For an input of buildings like [[2, 9, 10], [3, 7, 15], [5, 12, 12], [15, 20, 10], [19, 24, 8]], the algorithm would output the skyline points as [[2, 10], [3, 15], [7, 12], [12, 0], [15, 10], [20, 8], [24, 0]]. This output represents the key points where the height of the skyline changes as you move from left to right across the city.

Conclusion:

- The Skyline Problem is a classic computational geometry challenge that can be efficiently solved using a sweep line algorithm with a priority queue. By processing building start and end events in a carefully sorted manner and using a max-heap to maintain active building heights, the algorithm can efficiently determine the key points of the skyline. This solution is optimal for large inputs and ensures that the skyline is correctly represented with no unnecessary horizontal lines.