Largest Rectangular Area in Histogram LeetCode

Given an array representing the heights of bars in a histogram, find the largest rectangle area that can be formed within the histogram.

Example

Consider the following histogram:

Histogram: [2, 1, 5, 6, 2, 3]

Output

The largest rectangle area that can be formed within the histogram is 10.

Approach 1: Find Largest Rectangle Area in Histogram using Brute Force approach

In this approach, we consider each bar as a potential starting point of the rectangle. For each starting point, we calculate the area of the potential rectangle by finding the minimum height within the bars and calculating the breadth. We keep track of the maximum area seen so far and return it as the result.

Steps:

1. Initialize a variable **area** to store the maximum area and set it to 0.
2. Iterate through each bar in the histogram:
    - For each bar, iterate through the subsequent bars to calculate the potential rectangle's area.
    - Find the minimum height within the current potential rectangle.
    - Calculate the breadth of the rectangle.
    - Update the maximum area if the current rectangle's area is greater.
3. Return the maximum area as the result.

Time Complexity:

- The approach involves nested loops, where the outer loop iterates through each bar and the inner loop iterates through the subsequent bars.
- **Overall, the time complexity is approximately O(n^2), where n is the number of bars in the histogram.**

Space Complexity:

- The approach uses only a few variables to store indices and the maximum area.
- **Space Complexity: O(1).**

Approach 2: Find Largest Rectangle Area in Histogram using Stack

In this approach, we use a stack to efficiently find the previous smaller and next smaller elements for each bar in the histogram. We traverse the histogram twice to find the indices of the previous smaller and next smaller elements. Then, we use these indices to calculate the potential rectangle area for each bar.

Steps:

1. Traverse the histogram to find the indices of the next smaller element for each bar using a stack.
2. Traverse the histogram again to find the indices of the previous smaller element for each bar using a stack.
3. Initialize a variable **area** to store the maximum area and set it to **INT_MIN**.
4. For each bar, calculate the area of the potential rectangle using the heights, indices of next smaller and previous smaller elements.
5. Update the maximum area if the current rectangle's area is greater.
6. Return the maximum area as the result.

Time Complexity:

- The approach involves traversing the histogram twice to find indices of previous and next smaller elements.
- The stack operations and the calculations for area are constant time.
- **Time Complexity**: **O(n), where n is the number of bars in the histogram.**

Space Complexity:

- The approach uses additional space for two arrays to store indices and a few variables.
- **Space Complexity**: **O(n), where n is the number of bars in the histogram.**