<https://leetcode.ca/all/1762.html>

There are n buildings in a line. You are given an integer array heights of size n that represents the heights of the buildings in the line.

The ocean is to the right of the buildings. A building has an ocean view if the building can see the ocean without obstructions. Formally, a building has an ocean view if all the buildings to its right have a smaller height.

Return a list of indices (0-indexed) of buildings that have an ocean view, sorted in increasing order.

**Example 1:**

Input: heights = [4,2,3,1]

Output: [0,2,3]

Explanation: Building 1 (0-indexed) does not have an ocean view because building 2 is taller.

**Example 2:**

Input: heights = [4,3,2,1]

Output: [0,1,2,3]

Explanation: All the buildings have an ocean view.

**Example 3:**

Input: heights = [1,3,2,4]

Output: [3]

Explanation: Only building 3 has an ocean view.

**Example 4:**

Input: heights = [2,2,2,2]

Output: [3]

Explanation: Buildings cannot see the ocean if there are buildings of the same height to its right.

**Constraints:**

1 <= heights.length <= 10^5

1 <= heights[i] <= 10^9

**Attempt 1: 2024-06-01**

**Solution 1: Decreasing Monotonic Stack (10 min)**

public class Solution {

public int[] findBuildings(int[] heights) {

Stack<Integer> stack = new Stack<>();

for(int i = 0; i < heights.length; i++) {

while(!stack.isEmpty() && heights[stack.peek()] <= heights[i]) {

stack.pop();

}

stack.push(i);

}

int n = stack.size();

int[] result = new int[n];

int i = n - 1;

while(i >= 0) {

result[i] = stack.pop();

i--;

}

return result;

}

public static void main(String[] args) {

Solution so = new Solution();

int[] heights = new int[]{4,2,3,1};

int[] result = so.findBuildings(heights);

System.out.println(result);

}

}

Time complexity: O(n) — We iterate through the input array once

Space complexity: O(n) — The worst-case scenario is that all the buildings have ocean views,

so the monotonic stack holds all indexes of the input array

**Refer to**

<https://walkccc.me/LeetCode/problems/1762/#__tabbed_1_2>

class Solution {

public int[] findBuildings(int[] heights) {

Deque<Integer> stack = new ArrayDeque<>();

for (int i = 0; i < heights.length; ++i) {

while (!stack.isEmpty() && heights[stack.peek()] <= heights[i])

stack.pop();

stack.push(i);

}

int[] ans = new int[stack.size()];

for (int i = 0; i < ans.length; ++i)

ans[ans.length - 1 - i] = stack.pop();

return ans;

}

}