Largest Rectangle

Skyline Real Estate Developers is planning to demolish a number of old, unoccupied buildings and construct a shopping mall in their place. Your task is to find the largest solid area in which the mall can be constructed.

There are a number of buildings in a certain two-dimensional landscape. Each building has a height, given by $h[i] \text{ where } i \in [1,\ n]$. If you join k adjacent buildings, they will form a solid rectangle of area $k \times min(h[i], h[i+1], \ldots, h_{[}i+k-1])$.

Example

$h = [3, 2, 3]$

A rectangle of height $h = 2$ and length $k = 3$ can be constructed within the boundaries. The area formed is $h \cdot k = 2 \cdot 3 = 6$.

Function Description

Complete the function `largestRectangle` int the editor below. It should return an integer representing the largest rectangle that can be formed within the bounds of consecutive buildings.

largestRectangle has the following parameter(s):

- *int h[n]:* the building heights

Returns

- *long:* the area of the largest rectangle that can be formed within the bounds of consecutive buildings

Input Format

The first line contains n, the number of buildings.
The second line contains n space-separated integers, each the height of a building.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq h[i] \leq 10^6$

Sample Input

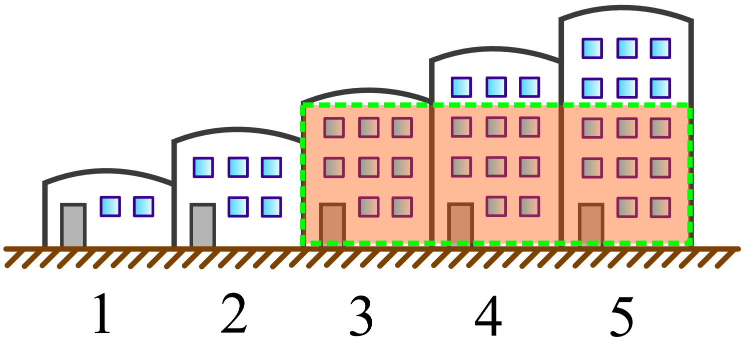
```
STDIN       Function
-----       --------
5           h[] size n = 5
1 2 3 4 5   h = [1, 2, 3, 4, 5]
```

Sample Output

```
9
```

Explanation

An illustration of the test case follows.



Contest ends in 2 months

Submissions: 45

Max Score: 10

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

More

Java 8

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;

class Result {

    /*
     * Complete the 'largestRectangle' function below.
     *
     * The function is expected to return a LONG_INTEGER.
     * The function accepts INTEGER_ARRAY h as parameter.
     */

    public static long largestRectangle(List<Integer> h) {
    // Write your code here
        long maximum = 0;
     for(int i = 0; i < h.size(); i++) {
            long count = 1;
            Integer curr = h.get(i);
        for (int j = i + 1; j < h.size(); j++) {  //checking right side of curr
                if (curr > h.get(j))
                    break;
                count++;
            }
        for (int j = i -1; j >= 0; j--) {   //checking left side of curr
                if (curr > h.get(j))
                    break;
                count++;
            }
            long sum = count * curr;
            if (sum > maximum) {
                maximum = sum;
            }
        }
    return maximum;
    }



}

public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT_PATH")));

        int n = Integer.parseInt(bufferedReader.readLine().trim());

        List<Integer> h = Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))
            .map(Integer::parseInt)
            .collect(toList());

        long result = Result.largestRectangle(h);

        bufferedWriter.write(String.valueOf(result));
        bufferedWriter.newLine();

        bufferedReader.close();
        bufferedWriter.close();
    }
}

```

Line: 46 Col: 5

Upload Code as File □ Test against custom input Run Code Submit Code

Testcase 0 ✓ | Testcase 1 ✓ | Testcase 2 ✓

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
1 2 3 4 5
```

Your Output (stdout)

```
9
```

Expected Output

```
9
```

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |