Our Solution(s) Run Code

Solution 1

```java
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Program {
  // O(n) time | O(1) space - where n is the length of the input
  public static int longestPeak(int[] array) {
    int longestPeakLength = 0;
    int i = 1;
    while (i < array.length - 1) {
      boolean isPeak = array[i - 1] < array[i] && array[i] > ar
      if (!isPeak) {
        i += 1;
        continue;
      }

      int leftIdx = i - 2;
      while (leftIdx >= 0 && array[leftIdx] < array[leftIdx + 1
        leftIdx -= 1;
      }

      int rightIdx = i + 2;
      while (rightIdx < array.length && array[rightIdx] < array[
        rightIdx += 1;
      }
      int currentPeakLength = rightIdx - leftIdx - 1;
      if (currentPeakLength > longestPeakLength) {
        longestPeakLength = currentPeakLength;
      }
      i = rightIdx;
    }
    return longestPeakLength;
  }
}

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```java
class Program {
  public static int longestPeak(int[] array) {
    // Write your code here.
    return -1;
  }
}

```

Our Tests

Custom Output Submit Code

```
@Test
public void Test1Case() {
  var input = new int[] {},
  var expected = 0,
  var actual = Program.LongestPeak(input),
  Utils.assertTrue(expected == actual),
}

@Test
public void Test2Case() {
  var input = new int[] {1, 3, 2},
  var expected = 3,
  var actual = Program.LongestPeak(input),
  Utils.assertTrue(expected == actual),
}

@Test
public void Test3Case() {
  var input = new int[] {1, 2, 3, 4, 5, 1},
  var expected = 6,
  var actual = Program.LongestPeak(input),
```

Run or submit code when you're ready.