```
1  // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3  // O(nlogn) time | O(n) space
4  function longestIncreasingSubsequence(array) {
5    const sequences = new Array(array.length);
6    const indices = new Array(array.length + 1);
7    let length = 0;
8    for (let i = 0; i < array.length; i++) {
9      const num = array[i];
10     const newLength = binarySearch(1, length, indices, array, num);
11     sequences[i] = indices[newLength - 1];
12     indices[newLength] = i;
13     length = Math.max(length, newLength);
14   }
15   return buildSequence(array, sequences, indices[length]);
16 }
17
18 function binarySearch(startIdx, endIdx, indices, array, num) {
19   if (startIdx > endIdx) return startIdx;
20   const middleIdx = Math.floor((startIdx + endIdx) / 2);
21   if (array[indices[middleIdx]] < num) {
22     startIdx = middleIdx + 1;
23   } else {
24     endIdx = middleIdx - 1;
25   }
26   return binarySearch(startIdx, endIdx, indices, array, num);
27 }
28
29 function buildSequence(array, sequences, currentIdx) {
30   const sequence = [];
31   while (currentIdx !== undefined) {
32     sequence.unshift(array[currentIdx]);
33     currentIdx = sequences[currentIdx];
34   }
35   return sequence;
36 }
37
38 exports.longestIncreasingSubsequence = longestIncreasingSubsequence;
39
```