

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 // O(n^2) time | O(n) space
4 function maxSumIncreasingSubsequence(array) {
5   const sequences = new Array(array.length);
6   const sums = array.map(num => num);
7   let maxSumIdx = 0;
8   for (let i = 0; i < array.length; i++) {
9     const currentNum = array[i];
10    for (let j = 0; j < i; j++) {
11      const otherNum = array[j];
12      if (otherNum < currentNum && sums[j] + currentNum >= sums[i]) {
13        sums[i] = sums[j] + currentNum;
14        sequences[i] = j;
15      }
16    }
17    if (sums[i] >= sums[maxSumIdx]) maxSumIdx = i;
18  }
19  return [sums[maxSumIdx], buildSequence(array, sequences, maxSumIdx)]
20 }
21
22 function buildSequence(array, sequences, currentIdx) {
23   const sequence = [];
24   while (currentIdx !== undefined) {
25     sequence.unshift(array[currentIdx]);
26     currentIdx = sequences[currentIdx];
27   }
28   return sequence;
29 }
30
31 exports.maxSumIncreasingSubsequence = maxSumIncreasingSubsequence;
32
```

Solution 1   Solution 2   Solution 3

```
1 function maxSumIncreasingSubsequence(array) {
2   // Write your code here.
3 }
4
5 // Do not edit the line below.
6 exports.maxSumIncreasingSubsequence = maxSumIncreasingSubsequence;
7
```

Our Tests

Custom Output

Submit Code

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 // O(n^2) time | O(n) space
4 function maxSumIncreasingSubsequence(array) {
5   const sequences = new Array(array.length);
6   const sums = array.map(num => num);
7   let maxSumIdx = 0;
8   for (let i = 0; i < array.length; i++) {
9     const currentNum = array[i];
10    for (let j = 0; j < i; j++) {
11      const otherNum = array[j];
12      if (otherNum < currentNum && sums[j] + currentNum >= sums[i]) {
13        sums[i] = sums[j] + currentNum;
14        sequences[i] = j;
15      }
16    }
17    if (sums[i] >= sums[maxSumIdx]) maxSumIdx = i;
18  }
19  return [sums[maxSumIdx], buildSequence(array, sequences, maxSumIdx)]
20 }
21
22 function buildSequence(array, sequences, currentIdx) {
23   const sequence = [];
24   while (currentIdx !== undefined) {
25     sequence.unshift(array[currentIdx]);
26     currentIdx = sequences[currentIdx];
27   }
28   return sequence;
29 }
30
31 exports.maxSumIncreasingSubsequence = maxSumIncreasingSubsequence;
32
```

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 // O(n^2) time | O(n) space
4 function maxSumIncreasingSubsequence(array) {
5   const sequences = new Array(array.length);
6   const sums = array.map(num => num);
7   let maxSumIdx = 0;
8   for (let i = 0; i < array.length; i++) {
9     const currentNum = array[i];
10    for (let j = 0; j < i; j++) {
11      const otherNum = array[j];
12      if (otherNum < currentNum && sums[j] + currentNum >= sums[i]) {
13        sums[i] = sums[j] + currentNum;
14        sequences[i] = j;
15      }
16    }
17    if (sums[i] >= sums[maxSumIdx]) maxSumIdx = i;
18  }
19  return [sums[maxSumIdx], buildSequence(array, sequences, maxSumIdx)]
20 }
21
22 function buildSequence(array, sequences, currentIdx) {
23   const sequence = [];
24   while (currentIdx !== undefined) {
25     sequence.unshift(array[currentIdx]);
26     currentIdx = sequences[currentIdx];
27   }
28   return sequence;
29 }
30
31 exports.maxSumIncreasingSubsequence = maxSumIncreasingSubsequence;
32
```

```
10 # Run Test Case 40 - Success (1/1)
11 # Run expect program definition test program (1/1)
12 #
13
14 # Run Test Case 41 - Success (1/1)
15 # Run expect program definition test program (1/1)
16 #
17
18 # Run Test Case 42 - Success (1/1)
19 # Run expect program definition test program (1/1)
20 #
21
22 # Run Test Case 43 - Success (1/1)
23 # Run expect program definition test program (1/1)
24 #
25
26 # Run Test Case 44 - Success (1/1)
27 # Run expect program definition test program (1/1)
28 #
29
30 # Run Test Case 45 - Success (1/1)
31 # Run expect program definition test program (1/1)
32 #
33
```

Run or submit code when you're ready.