Our Solution(s) Run Code

Solution 1

```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

# O(n^2) time | O(n) space
def maxSumIncreasingSubsequence(array):
    sequences = [None for x in array]
    sums = [num for num in array]
    maxSumIdx = 0
    for i in range(len(array)):
        currentNum = array[i]
        for j in range(0, i):
            otherNum = array[j]
            if otherNum < currentNum and sums[j] + currentNum >= sums
                sums[i] = sums[j] + currentNum
                sequences[i] = j
        if sums[i] >= sums[maxSumIdx]:
            maxSumIdx = i
    return [sums[maxSumIdx], buildSequence(array, sequences, maxSumId


def buildSequence(array, sequences, currentIdx):
    sequence = []
    while currentIdx is not None:
        sequence.append(array[currentIdx])
        currentIdx = sequences[currentIdx]
    return list(reversed(sequence))

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```python
def maxSumIncreasingSubsequence(array):
    # Write your code here.
    pass

```

Our Tests

Custom Output Submit Code

```
    def test_case_2(self):
        self.assertEqual(Solution().longestSubsequence([1,3],1), 1)

    def test_case_3(self):
        self.assertEqual(Solution().longestSubsequence([1,3,5,7],1), 1)

    def test_case_4(self):
        self.assertEqual(Solution().longestSubsequence([1,5,7,8,9,10,6,8],4), 5)

    def test_case_5(self):
        self.assertEqual(Solution().longestSubsequence([3,2,1,5,4],1), 2)

    def test_case_6(self):
        self.assertEqual(Solution().longestSubsequence([1,2,3,4],-1), 0)
```

Run or submit code when you're ready.