Max-Sum-Subarray

May 14, 2020

0.0.1 Problem Statement

You have been given an array containg numbers. Find and return the largest sum in a contiguous subarray within the input array.

Example 1: * `arr= [1, 2, 3, -4, 6]` * The largest sum is `8`, which is the sum of all elements of the array.

Example 2: * `arr = [1, 2, -5, -4, 1, 6]` * The largest sum is `7`, which is the sum of the last two elements of the array.

```
In [1]: def max_sum_subarray(arr):
            """
            :param - arr - input array
            return - number - largest sum in contiguous subarry within arr
            """
        #     Wrong Solution
            total = 0
            maxTotal = 0
            for item in arr:
                if item > 0:
                    total += 0
                    if total > max:
                        max = total
                else:
                    total = 0
            return max
            pass
```

Hide Solution

```
In [ ]: # Solution
        '''
        The Idea:
        1. We have to find the sum of "contiguous" subarray, therefore we must not change the or
        2. Let `current_sum` denotes the sum of a subarray, and `max_sum` denotes the maximum va
        3. LOOP STARTS: For each element of the array, update the `current_sum` with the MAXIMUM
         - The element added to the `current_sum` (denotes the addition of the element to the cu
         - The element itself  (denotes the starting of a new subarray)
         - Update (overwrite) `max_sum`, if it is lower than the updated `current_sum`
        4. Return `max_sum`
        '''

        def max_sum_subarray(arr):

            current_sum = arr[0] # `current_sum` denotes the sum of a subarray
            max_sum = arr[0]     # `max_sum` denotes the maximum value of `current_sum` ever

            # Loop from VALUE at index position 1 till the end of the array
            for element in arr[1:]:

                '''
                # Compare (current_sum + element) vs (element)
                # If (current_sum + element) is higher, it denotes the addition of the element t
                # If (element) alone is higher, it denotes the starting of a new subarray
                '''
                current_sum = max(current_sum + element, element)

                # Update (overwrite) `max_sum`, if it is lower than the updated `current_sum`
                max_sum = max(current_sum, max_sum)

            return max_sum
```

```
In [9]: def test_function(test_case):
            arr = test_case[0]
            solution = test_case[1]

            output = max_sum_subarray(arr)
            if output == solution:
                print("Pass")
            else:
                print("Fail")
```

```
In [10]: arr= [1, 2, 3, -4, 6]
         solution= 8 # sum of array

         test_case = [arr, solution]
         test_function(test_case)
```

```
Pass
```

```
In [11]: arr = [1, 2, -5, -4, 1, 6]
         solution = 7   # sum of last two elements

         test_case = [arr, solution]
         test_function(test_case)
```

```
Pass
```

```
In [16]: arr = [-12, 15, -13, 14, -1, 2, 1, -5, 4]
         solution = 18  # sum of subarray = [15, -13, 14, -1, 2, 1]

         test_case = [arr, solution]
         test_function(test_case)
```

```
Pass
```