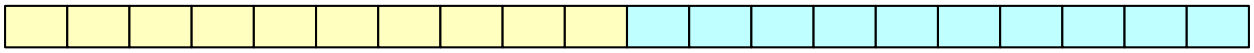
Homework 5

WARNING: -100 points and an Academic Misconduct Procedure if you copy from the web

Let $x_1, x_2, \ldots, x_n$ be a list of positive or negative integers. Give an $O(n)$ divide and conquer algorithm to determine the largest possible sum of a sub-sequence of consecutive items in the list. **Implement in Python. For simplicity's sake, your algorithm should return the largest sum of a contiguous sub-sequence.**

Example 1: 10, -20, 3, 4, 5, -1, -1, 12, -3, 1 Largest list has 3+4+5+-1+-1+12 = 22.

Hints: Obviously, you need to use recursion which needs a base case. When you divide the array into two halves, the best sub-sequence might be



(a) located entirely in the left half
(b) located entirely in the right half
(c) span both sides.
In the latter case, the left side of the longest sequence consists of two parts. The left one is the sub-sequence with largest sum including the end and the right one is the sub-sequence with largest sum including the beginning of the array. So, what you need to do is to calculate with recursion:
(a) The sub-sequence with largest sum in the array
(b) The sub-sequence with largest sum in the array starting at the beginning of the array.
(c) The sub-sequence with largest sum in the array ending at the end of the array.

Example

Let's look at an example for an array with 16 elements. We divided the array into a left and a right half.

4	-20	12	5	4	-10	2	5	-5	3	2	1	1	-10	-4	5

Recursively, the left array has:
- Largest sum sub-sequence starting at the beginning: [4, -20, 12, 5, 4] with sum 5
- Largest sum sub-sequence ending at the end: [12, 5, 4, -10, 2, 5] with sum 18
- Largest sum sub-sequence: [12, 5, 4] with sum 21.

The right array has:
- Largest sum sub-sequence starting at the beginning: [-5,3,2,1,1] with sum 2
- Largest sum sub-sequence ending at the end: [5] with sum 5
- Largest sum sub-sequence: [3,2,1,1] with sum 7.

To calculate the largest-sum sub-sequence starting at the beginning, we look at the following possibilities:
(1) We use the largest sum sub-sequence starting at the beginning of the left array: [4, -20, 12, 5, 4] with sum 5
(2) We use the complete left array (with a sum of 2) plus the largest-sum sub-sequence of the right array: [4, -20, 12, 5, 4, -10, 2, 5] + [-5, 3, 2, 1, 1] with a combined sum of 2 + 2.
Obviously, the first choice is better.

To calculate the largest-sum sub-sequence of the combined array ending at the end:
(1) We use the largest-sum sub-sequence of the right (green) array ending at the end, which is [5] with sum 5.
(2) Or we use the complete green array with sum -7 and the largest-sum sub-sequence ending at the end of the red array, with sum 18 for a total of 11.
Obviously, we pick alternative (2).

To calculate the largest-sum sub-sequence of the combined array, we get to pick between:
(1) the largest-sum sub-sequence of the left array with sum 21
(2) the largest-sum sub-sequence of the right array with sum 7
(3) the combination of the largest-sum sub-sequence ending with the end of the red array (sum 18) and the largest-sum sub-sequence beginning at the beginning of the green array (sum 2) for a total of 20.
We pick alternative (1).

We can amend the algorithm to return for every sub-array also the sum of its elements.

Run-time

For the divide phase, we just divide the array into two halves. For the conquer phase, we add the sums of the two halves, and then use addition to calculate five values and select three of them. This is done in constant time. Thus, we have

$$T(n) = 2T(n/2) + c.$$

According to the Master Theorem, this has runtime $\Theta(n)$.