# Chapter 4 solution

August 26, 2023

4.1-1 FIND-MAXIMUM-SUBARRAY returns the largest negative number in A.

4.1-2 Pseudocode for brute-force solution for maximum-subarray problem.

```
brute_force_maximum_subarray(A):
    best_sum = -inifinity
    best_low = -1
    best_high = -1
    for int i = 0 ... A.size - 1:
        sum = A[i]
        if sum > best_sum:
            best_sum = sum
            best_low = i
            best_high = i
        for int j = i+1 ... A.size - 1:
            sum += A[j]
            if sum > best_sum:
                best_sum = sum
                best_low = i
                best_high = j
    return best_low, best_high, best_sum
```

4.1-3 The crossover point is somewhere between 850 and 875. I don't think there is much point to find out exactly what n is since the performance measurements has too many noises when it comes to tiny differences.

Replacing the base case of recursive algorithm makes the crossover point lower. In other words, the modified recursive algorithms runs faster on smaller problem size n. See maximum_subarray.cpp for more details.

4.1-4 Change the initial sum of sub array search to 0 and a special value pair for low and high indicating it's an empty sub array.

4.1-5 Suppose we know the maximum sub array of A[0] to A[j] and we know the maximum sub array that ends at A[j] of A[0] to A[j]. For j+1, compute the following in constant time

- maximum sub array of A[0] to A[j]
- maximum sub array ending at A[j] + A[j+1]
- A[j + 1]

From these, we can easily compute the maximum sub array of A[0] to A[j+1] and the maximum sub array of A[0] to A[j+1] that ends at A[j+1]. Use induction and iterate until we get the maximum sub array of A.