Divide and Conquer Theoretical Analysis

Psuedocode

Note: Originally based on pseudocode found in Introduction to Algorithms, 3rd Edition.

```
FIND-MAX-SUBARRAY(A, low, high)
if high = low
    return low, high, A[low]
else
    mid = floor((low+high)/2)
    left-low, left-high, left-sum = FIND-MAX-SUBARRAY(A, low, mid)
    right-low, right-high, right-sum = FIND-MAX-SUBARRAY(A, mid+1, high)
    left-sum = 0
    max-left = -inf
    max-subarray-sum = 0
    for i = mid <- low
        sum = sum + A[i]
        if sum > left-sum
            left-sum = sum
            max-left = i
    right-sum = 0
    max-right = -inf
    max-subarray-sum = 0
    for i = mid+1 -> high
        sum = sum + A[i]
        if sum > right-sum
            right-sum = sum
            max-right = i
    cross-low, cross-high, cross-sum = max-left, max-right, left-sum+right-sum
    if left-sum >= right-sum and left-sum >= cross-sum
        return left-low, left-high, left-sum
    else if right-sum >= left-sum and right-sum >= cross-sum
        return right-low, right-high, right-sum
    else
        return cross-low, cross-high, cross-sum
```

Theoretical Runtime Analysis

The divide and conquer algorithm for finding a maximum subarray can be divided into three parts:

1. Setup, control statements, calculations, comparisons, and base case handling
2. Recursive subproblems
3. Maximum subarray crossing midpoint subproblem

The first section includes the lines that occur in Θ(1) time. These constant time tasks include the work done on the following lines:

1. Lines 1-2: Handling the T(1) base case
2. Line 4: Calculating the midpoint of the array or subarray
3. Lines 7-9, 11-14: Initialization, comparison, and calculations for the crossing sub-problem
4. Lines 15-17, 19-22: Initialization, comparison, and calculations for the crossing sub-problem
5. Lines 24-29: Comparisons and calculations for the combine step
6. Misc: Control statements (e.g. if/else if/else/return)

The second section includes slines related to the recursive sub-problems. There are two subproblems of n/2 size. For purposes of this analysis we will consider n as a power of two, such that n/2 is always an integer. Adding the constant time work done by the lines in the previous section gives us a recurrence relation of $T(n) = 2T(n/2) + \Theta(1)$. The recursive tasks include the work done on the following lines:

1. Line 5: Recursive subproblem handling the left subarray
2. Line 6: Recursive subproblem handling the right subarray

The final section includes the lines related to the subproblem which finds a maximum subarray crossing the midpoint of A[low..high]. This work occurs in the `for` loops which are initialized on lines 10 and 18. Each of these loops processes an n/2 subarray, for a total of n elements processed. Therefore, the work done during the crossing sub-problem occurs in Θ(n) time. The linear time tasks include the work done in the following lines:

3. Line 10: Loop handing the left subarray for the crossing subproblem
4. Line 18: Loop handing the right subarray for the crossing subproblem

At this point the recurrence relation is $T(n) = 2T(n/2) + \Theta(n) + \Theta(1)$. Using the master theorem to solve this recurrence indicates that the total runtime for the divide and conquer approach is $\Theta(n \lg n)$.