Divide and Conquer Technique

Week-3

DSA-II Lab

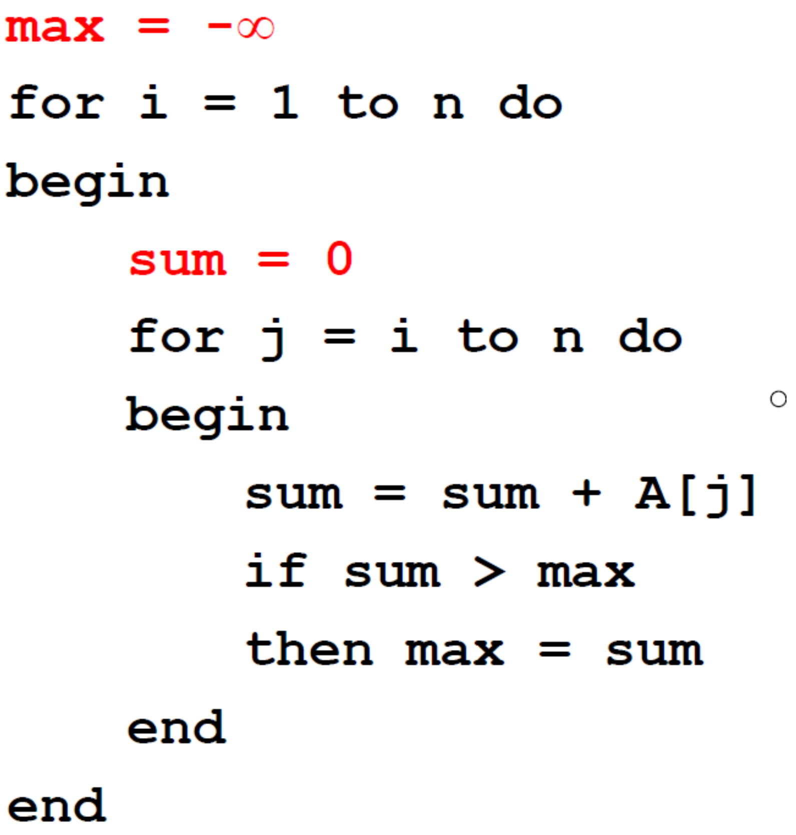
The Divide-and-Conquer algorithm:

```
 procedure Rmaxmin ($i$, $j$, $fmax$, $fmin$);     // $i$, $j$ are index #, $fmax$,
begin                                              // $fmin$ are output parameters
      case:
            $i = j$:        $fmax \leftarrow fmin \leftarrow A[i]$;
            $i = j - 1$:    if $A[i] < A[j]$ then      $fmax \leftarrow A[j]$;
                                                       $fmin \leftarrow A[i]$;
                            else     $fmax \leftarrow A[i]$;
                                     $fmin \leftarrow A[j]$;

            else:           $mid \leftarrow (i + j)/2$;

                            call Rmaxmin ($i$, $mid$, $gmax$, $gmin$);
                            call Rmaxmin ($mid+1$, $j$, $hmax$, $hmin$);
                            $fmax \leftarrow$ MAX ($gmax$, $hmax$);
                            $fmin \leftarrow$ MIN ($gmin$, $hmin$);
      end
    end;
```

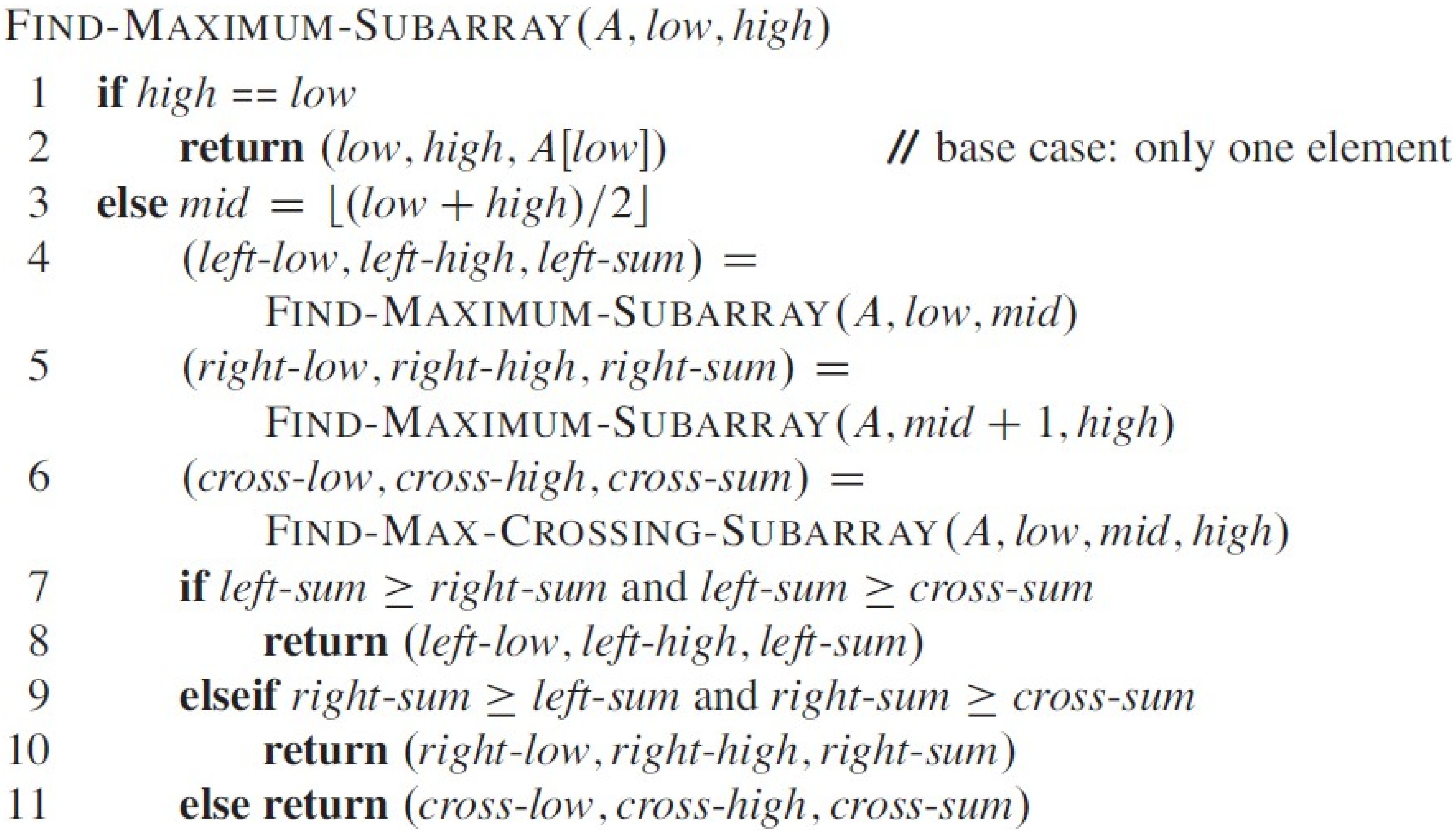
Find Maximum Subarray



```
max = -∞
for i = 1 to n do
begin
    sum = 0
    for j = i to n do
    begin
        sum = sum + A[j]
        if sum > max
        then max = sum
    end
end
```

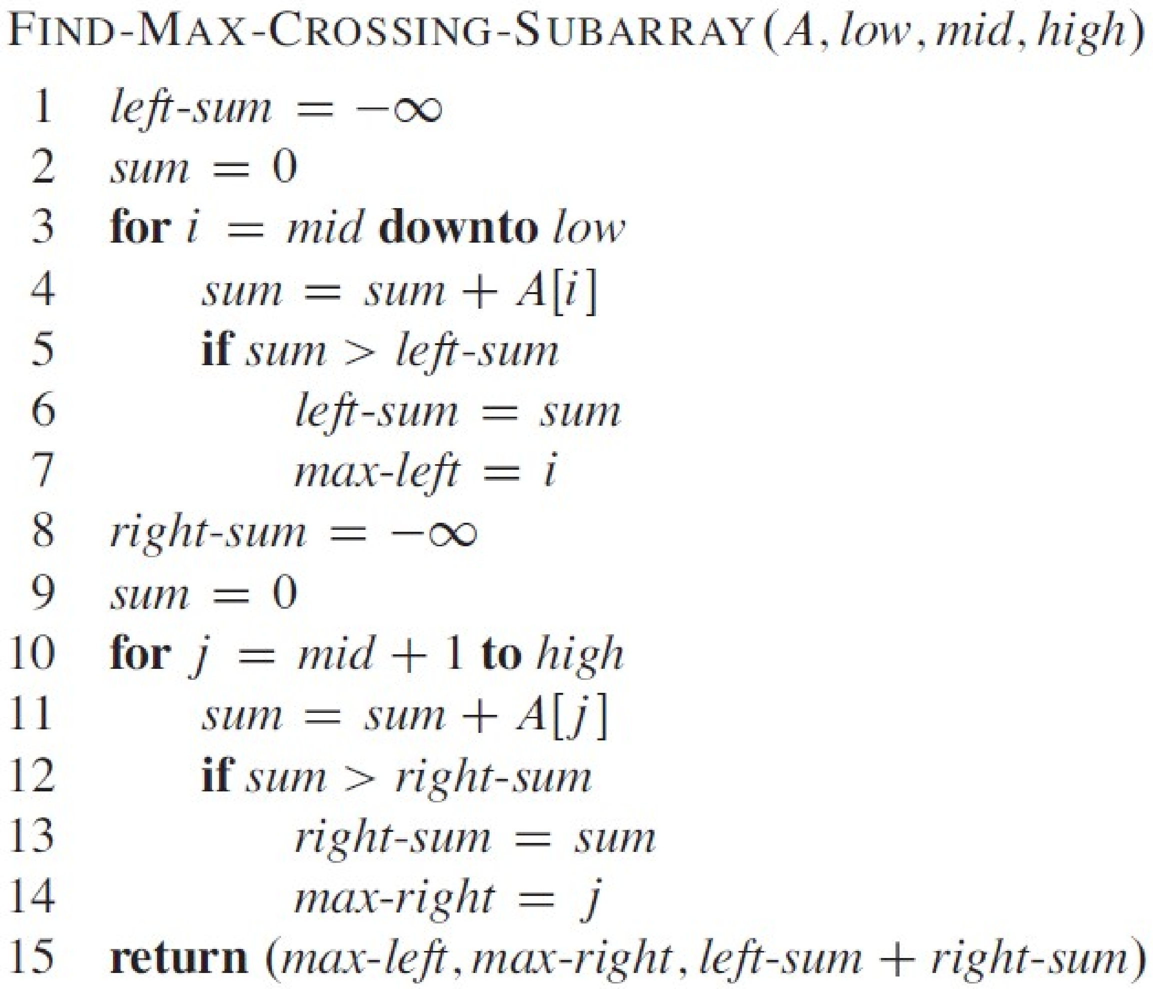
Brute Force Approach

Find Maximum Subarray



```
FIND-MAXIMUM-SUBARRAY($A, low, high$)
if $high == low$
    return ($low, high, A[low]$)        // base case: only one element
else $mid = \lfloor (low + high)/2 \rfloor$
    ($left\text{-}low, left\text{-}high, left\text{-}sum$) =
        FIND-MAXIMUM-SUBARRAY($A, low, mid$)
    ($right\text{-}low, right\text{-}high, right\text{-}sum$) =
        FIND-MAXIMUM-SUBARRAY($A, mid + 1, high$)
    ($cross\text{-}low, cross\text{-}high, cross\text{-}sum$) =
        FIND-MAX-CROSSING-SUBARRAY($A, low, mid, high$)
    if $left\text{-}sum \geq right\text{-}sum$ and $left\text{-}sum \geq cross\text{-}sum$
        return ($left\text{-}low, left\text{-}high, left\text{-}sum$)
    elseif $right\text{-}sum \geq left\text{-}sum$ and $right\text{-}sum \geq cross\text{-}sum$
        return ($right\text{-}low, right\text{-}high, right\text{-}sum$)
    else return ($cross\text{-}low, cross\text{-}high, cross\text{-}sum$)
```

Find Maximum Subarray



```
FIND-MAX-CROSSING-SUBARRAY($A, low, mid, high$)
left-sum $= -\infty$
sum $= 0$
for $i = mid$ downto $low$
    sum $=$ sum $+ A[i]$
    if sum $>$ left-sum
        left-sum $=$ sum
        max-left $= i$
right-sum $= -\infty$
sum $= 0$
for $j = mid + 1$ to $high$
    sum $=$ sum $+ A[j]$
    if sum $>$ right-sum
        right-sum $=$ sum
        max-right $= j$
return (max-left, max-right, left-sum $+$ right-sum)
```

Practices(Solve using divide and conquer technique)

1. Count total even numbers in an array using divide and conquer technique.
2. Count total vowels in a string using divide and conquer technique.
3. Find the largest pair in a given array.
4. Given an array of size n, return *the majority element*.The majority element is the element that appears more than ⌊n/2⌋ times. You may assume that the majority element always exists in the array.
5. You have an array of integers where some elements may be negative, zero, or positive. Your goal is to find the contiguous subarray (subarray with consecutive elements) with the maximum sum of non-negative numbers. Note that, the subarray shouldn’t have any negative number.
6. You are given a list of students who have name, id and cgpa. Your task is to find the student with highest cgpa. [you can’t sort the list]

Practices(Solve using divide and conquer technique)

7. Given an integer array nums, count *the number of smaller elements to the right of* nums[i].

Example 1:
Input: nums = [5,2,6,1]
Output: [2,1,1,0]
Explanation: To the right of 5 there are **2** smaller elements (2 and 1). To the right of 2 there is only **1** smaller element (1). To the right of 6 there is **1** smaller element (1). To the right of 1 there is **0** smaller element.

Example 2:
Input: nums = [-1] **Output:** [0]