Pseudocode Project 1 Algorithm 3

Use divide and conquer approach. Divide array into halves until a subarray of one is reached. Then calculate maxSum for three cases. Case 1: maxSum is in leftSubArray. Case 2: maxSum is int rightSubArray. Case 3: maxSum is a subarray that spans the left and right subarrays. Base case the subarray maxSum equals the value of the index.

MSS maxSumSubArray( array [], start, end, side ){

mid = (end – start)/2

//base case break recursion

if(start > end){

return

}ese{

leftMSS = maxSumSubArray( array [], start, mid-1, left)

rightMSS = maxSumSubArray( array [], mid, end,, right)

}

//found leaf of tree

if(start == end){

set the max sum the start and the end of the sub array

//calculate the span to see if it contains the max sum sub array

} else{

rightSpan = leftSpan = neg infinity;

sum = 0

spanStart = spanEnd = 0

for(int x = end; x > start; --x){

calculate the left span max sum

}

for(int x = start; x < end; ++x){

calculate the right span max sum

}

//determine where the max sum sub array is and return it

if (leftSpan + rightSpan) > leftMSS.mSSA && (leftSpan + rightSpan) > leftMSS.mSSA){

MSS.mSSA = if (leftSpan + rightSpan)

MSS.start = spanStart; MSS.end = spanEnd

}elseif(leftMSS.mSSA > rightMSS.mSSA){

return leftMSS

}else{

return rightMSS

}

}

return MSS

}