

Jake Tantoraki

## Problem 1

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n^2)$$

Input testarray1[], testarray2[]

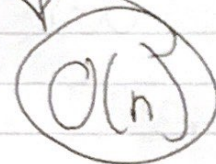
$$1 > \log_2 1$$

findmaxdiff(A[], start, end)

$$O(n^2)$$

if (start >= end)  
return -1 // 1

mid = start + (end - start) / 2 // 1  
leftDiff = findmaxdiff(A, start, mid)  
rightDiff = findmaxdiff(A, mid+1, end)


$$O(n)$$

minLeft = minF(A, start, mid) // 1  
maxRight = maxF(A, mid, end) // 1

tdiff = maxRight - minLeft // 1

return Math.max(tdiff, Math.max(leftDiff, rightDiff))

## Problem 2

checksum(s1[], s2[], x)

s = 0

e = s1.length - 1

result = false

for (i = 0; i < s1.length; i++) // O(n)

if (s1[s] + s2[e] = x)  
result = true; // 1

else if ( $s1[e] + s2[e] > x$ )  
     $e = e - 1$  //1

else  
     $s = s + 1$  //1

$$O(n) + O(1) = O(n)$$

Problem 3

commonElements( $s1[]$ ,  $s2[]$ )

$i = 0$  //1

$j = 0$  //1

while ( $i < s1.length$  &  $j < s2.length$ ) //2n

if ( $s1[i] == s2[j]$ )

    System.out.println( $s1[i]$ , " ")

$i++$

$j++$

else if ( $s1[i] < s2[j]$ )

$i++$

else

$j++$

$$O(n) + O(1) = O(n)$$