

# Class Exercises: Divide and Conquer 2

Analysis and Design of Algorithms

5 de mayo de 2024

**Ejercicio 1.** Run the maximum subarray algorithm on the following array:  $\langle 2, -2, 3, 5, -3, 0, 3, -8, 9 \rangle$

**Ejercicio 2.** Run the Karatsuba algorithm on the following numbers: 16541533 and 41142534. You must show all the involved steps.

**Ejercicio 3.** Run the Strassen algorithm on the following matrices:

$$A = \begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix},$$

$$B = \begin{pmatrix} 2 & 4 \\ 8 & 6 \end{pmatrix}.$$

You must show all the involved steps.

**Ejercicio 4.** Consider the following problem.

Input: An array  $A[1..n]$  of integers.

Output: The number of significant inversions, where a *significant inversion* is an ordered pair  $(i, j)$  such that  $i < j$  and  $A[i] > 2A[j]$ .

Design a  $\Theta(n \lg n)$  algorithm for the above problem. Write the recurrence for the worst-case execution time of the algorithm and solve it using the master theorem.

**Ejercicio 5.** Consider the following problem.

Input: Two vectors  $A[1..n]$  and  $B[1..n]$  with pairwise distinct elements, sorted in increasing order.

Output: The median of the set of elements that are in  $A$  or  $B$ .

That is, the element  $v$  such that there are exactly  $n - 1$  elements less than  $v$  in  $A$  or  $B$ .

For example, if  $A = [10, 30, 50, 70]$ ,  $B = [20, 40, 60, 80]$ , the corresponding median is 40, since  $n = 4$  and there are 3 elements less than 40. Design a divide and conquer algorithm with complexity  $\Theta(\lg n)$  for the median problem. In this exercise, you can assume that  $n$  is a power of 2. Write the recurrence for the worst-case execution time of the algorithm and solve it using the master theorem.

**Ejercicio 6.** We say that an array  $A[1..n]$  is *unimodal* if there exists an index  $p$ , called the *peak*, such that  $A[1..p]$  is an increasing sequence, and  $A[p..n]$  is a decreasing sequence. Design a divide and conquer algorithm that takes a unimodal array and finds the peak of  $A$ . Your algorithm should have a worst-case complexity of  $\Theta(\lg n)$ . Write the pseudocode of the algorithm. Write the recurrence for the worst-case execution time of the algorithm and solve it using the master theorem.

**Ejercicio 7.** Consider the following search problem.

Input: An array  $A[1..n]$  of integers.

Output: The value  $A[i]$  such that there are more than  $n/2$  numbers equal to  $A[i]$ . If there is no such value, return  $-1$ . For example, if  $A = [2, 4, 2, 4, 2, 2, 1, 4, 2]$ , the algorithm should return the value 2.

Design a divide and conquer algorithm with complexity  $\Theta(n \lg n)$  for the above problem. Explain the idea of your algorithm with an example. Write the pseudocode of the algorithm. Write the recurrence for the worst-case execution time of the algorithm and solve it using the master theorem. Note: You cannot use any pre-existing routines, such as sorting.

**Ejercicio 8.** Given an array  $A[1..n]$ , a  $k$ -rotation of  $A$  is an array  $B[1..n]$  such that

$$B(k) = \begin{cases} A[i+k] & i+k \leq n \\ A[(i+k) \bmod n] & \text{otherwise} \end{cases}$$

For example, if  $A = [3, 6, 9, 10]$ , a 2-rotation of  $A$  is  $B = [9, 10, 3, 6]$ .

Consider the following problem. Input: A  $k$ -rotation  $B$  of an array sorted in ascending order of distinct elements. Output: The number  $k$ . For example, if  $B = [9, 10, 3, 6]$ , the algorithm should return the value 2.

Design a  $\Theta(\lg n)$  worst-case time algorithm for the problem. Write the pseudocode of the algorithm. Write a recurrence for the worst-case of this algorithm. Verify with the master theorem.

**Ejercicio 9.** Consider the following problem.

Input: A set of  $k$  sorted arrays  $A_1, A_2, \dots, A_k$ , each of them sorted in increasing order, which together have size  $n$ .

Output: An array  $B[1..n]$  with all the elements in the input sorted. For example, if  $A_1 = [1, 3], A_2 = [2, 7, 8], A_3 = [3, 7]$ , the algorithm should return  $B = [1, 2, 3, 3, 7, 7, 8]$ .

Design a divide and conquer algorithm that consumes  $\Theta(n \lg k)$  time in the worst case. Write the pseudocode of the algorithm. Write a recurrence for the worst-case of this algorithm. Verify with the master theorem.