UNIVERSIDAD DE MÁLAGA

Analysis and Design of Algorithms
Divide and Conquer
(2º – Comput. Sci., Comput. Eng. & Software Eng.)
E.T.S.I. INFORMÁTICA

Problem list

Essential Exercises

1. Given an ordered array V whose n numerical elements are all different, define an algorithm to find in time $O(\log n)$ an index i such that $0 \leqslant i < n$ and $V[i] = i$, if it exists.

2. Given two (ascendingly) ordered arrays of n elements, write an algorithm of logarithmic complexity for finding the median of the $2n$ elements.

3. An element in an array is termed a *majority element* if it appears more than $m > \lfloor n/2 \rfloor$ times, where n is the length of the array.

   (a) Define a brute-force approach to determine whether a majority element exists in an array, and indicate its complexity.

   (b) Define a more sophisticated approach based on the method of divide and conquer.
   Hint: Note that if a majority element exists, there will be at least a pair of adjacent elements with the same value. Hence, it may suffice to find those elements which are repeated in adjacent positions, and repeat the procedure for these. If the total number of elements is odd, the last element can be ignored if there is a previous candidate, or can be selected as candidate is there is none. The final candidate at the end of the process must be checked to ensure it is a majority element.

4. Given an array V of n integers, find an algorithm of complexity $O(n \log n)$ for determining the portion of the array whose sum is maximal.

5. Given an integer array V whose n elements are all different and an integer S, design an algorithm for deciding in time $O(n \log n)$ whether there exist two elements in V whose sum is S or not.

6. Let A be an array of n different positive integers. We say positions i, j $(i < j)$ constitute an inversion iff $A[i] > A[j]$. We wish to obtain a function `int numInversions (int[] A);` returning the total number of inversions in an array, e.g., if $A = \{2, 4, 1, 3, 5\}$ then `numInversions`$(A)= 3$, corresponding to index pairs (1,3), (2,3) and (2,4).

   (a) Design a brute-force approach and determine its complexity in term of the number of comparisons performed.

   (b) Design a more efficient divide-and-conquer approach (**hint:** you can mix the elements if needed).

   (c) Find the complexity of the previous algorithm using the Master Theorem.

7. Let A be a unimodal integer array, namely an array whose elements are arranged in a (non-empty) strictly increasing sequence followed by a (non-empty) strictly decreasing sequence, e.g., $A = \{1, 5, 7, 9, 6, 2\}$. Consider a function "`int peak (int[] A);`" that given such a unimodal array returns its peak (i.e., maximum) value, e.g., `peak`$(A) = 9$ in the previous example. Feb 13

   - Design a brute-force approach and indicate its asymptotic complexity using comparisons as the basic operation.
   - Design a Divide-and-Conquer approach for this problem, more efficient than the previous brute-force approach.
   - Pose a recurrence for the computational complexity of the Divide-and-Conquer algorithm and find its asymptotic order of growth using the Master Theorem.

Additional Exercises

1. Build a divide-and-conquer approach to compute a^n given integers a, n. Indicate its complexity. Is it better than a brute force approach? If not, can it be modified so that it is better than the latter?

Jun 08 2. Implement a ternary search algorithm to determine whether an element is or not in a sorted array.

   func TernarySearch ($\downarrow e$: TElement, $\downarrow A$: TArray, $\downarrow l, r$: $\mathbb{N}$):$\mathbb{B}$

   If $L = r - l + 1$ is the length of the subarray explored, this algorithm must check positions at distance $L/3$ of the left end and distance $L/3$ of the right end, and proceed in a recursive way in the appropriate segment of the array.

   Define a recurrence for the number of comparisons in the worst case, and use the Master Theorem to find the asymptotic order of growth.

3. You have two boxes comprising n bolts and the corresponding n nuts respectively. You have to match each bolt with the nut it fits in. For this purpose you cannot compare bolts with bolts or nuts with nuts: you can only pair a bolt and a nut and decide whether the latter is too large, too small, or it fits OK. Build an $O(n \log n)$ algorithm.

4. Consider a binary search algorithm that divides an array into two parts comprising 1/3 and 2/3 of the elements. Compute its complexity and compare it to the original binary search algorithm.

5. Consider the *in place* version of Mergesort, in which no external storage is used and any element re-arrangement is done inside the the array to be sorted:

```
proc Mergesort (↓↑$A$: TArray, ↓$l, r$: $\mathbb{N}$)
variables $m$: $\mathbb{N}$
begin
    if $l<r$ then
        $m \leftarrow (l+l)/2$
        Mergesort($A$, $l$, $m$)
        Mergesort($A$, $m+1$, $r$)
        Merge($A$, $l$, $m+1$, $r$)
    endif
end

proc Merge (↓↑$A$: TArray, ↓$l, m, r$: $\mathbb{N}$)
variables $k$: $\mathbb{N}$; $tmp$: TElement
begin
    while $(l<m) \wedge (m \leqslant r)$ do
        if $A[l]>A[m]$ then
            $tmp \leftarrow A[m]$
            for $k \leftarrow m-1$ to $l$ step $-1$ do $A[k+1] \leftarrow A[k]$ endfor
            $A[l] \leftarrow tmp$
            $m \leftarrow m+1$
        endif
        $l \leftarrow i+1$
    endwhile
end
```

   Indicate its worst-case time complexity, assuming the copy of an element is the basic operation.

6. Build an algorithm to compute the median of a numerical array.
   (a) Use a brute force approach. Which is the complexity of this algorithm?
   (b) Use a divide-and-conquer approach (by exploiting the Divide procedure seen for Quicksort). Which is the complexity of this algorithm?

7. Let $A[1..n]$ be a positive integer array. We know all elements are prime except one of them, which we want to find. However, we cannot perform numerical operations on the elements. We just have a function: Jun 09

   func MorePrimes ($\downarrow A$: ARRAY $[1..n]$ OF $\mathbb{N}$, $\downarrow i, j, r, s$: $\mathbb{N}$): $\mathbb{Z}$

   that returns $+1/0/-1$ if there are more/as many/less primes in $A[i..j]$ as/than in $A[r..s]$.

   (a) Design a Divide-and-Conquer algorithm for this problem and determine its complexity using MorePrimes as the basic operation.

   (b) Solve now a variant of the problem in which all number are primes but one, or the other way around (but we do not know in advance whether is the latter or the former). We want to find the discrepant element.

8. Build a function that takes integers a and b as input parameters and returns $a \cdot b$. To do this, you can only use addition and multiplication/division by 2 as basic operations. Assume these operations to have the same cost. Jun 10

   (a) Use a brute force approach and indicate its complexity in terms of b.

   (b) Now use a Divide-and-Conquer approach and again indicate its complexity in terms of b.

Feb 14

9. Given a natural number n, we wish to determine whether n is the product of three consecutive natural numbers. For instance, $210 = 5 \cdot 6 \cdot 7$ but 12 cannot be obtained as the product of three consecutive numbers.

   (a) Design a *Divide&Conquer* algorithm with time complexity $O(\log n)$ to solve this problem.

   (b) Justify –posing a recurrence for the time complexity of the algorithm and solving it using the Master Theorem– that your algorithm has indeed $O(\log n)$ complexity.

10. An order-n Kasparovian matrix K_n is a matrix defined as:

$$K_0 = [1] \qquad K_n = \left[\begin{array}{c|c} K_{n-1} & -K_{n-1} \\ \hline -K_{n-1} & K_{n-1} \end{array}\right] \tag{1}$$

Sep 14

   For example, the first three matrices in the sequence are:

$$K_0 = [1] \qquad K_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad K_2 = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{2}$$

   (a) Find a Divide&Conquer approach (more efficient than brute force) to compute the product $K_n \times K_n$ for a given n.

   (b) Define a recurrence for the cost of the previous algorithm and find its asymptotic complexity using the Master Theorem. Consider that each arithmetic operation is a basic operation.

11. The following table shows the matches between $n = 8$ teams in a league: Feb 15

		League days						
	1	2	3	4	5	6	7	8
	2	1	4	3	6	5	8	7
	3	4	1	2	7	8	5	6
teams	4	3	2	1	8	7	6	5
	5	6	7	8	1	2	3	4
	6	5	8	7	2	1	4	3
	7	8	5	6	3	4	1	2
	8	7	6	5	4	3	2	1

Notice how the upper left box contains a league schedule involving only teams $1 \dots 4$. Also, the schedule for teams $5 \dots 8$ during the first 3 league days is the same as for teams $1 \dots 4$, only with an offset of 4. Similarly, notice how on days $5 \dots 7$ teams $1 \dots 4$ play against the same teams against which teams $5 \dots 8$ played during the first 3 days and vice versa. Finally, notice that on day 4 teams $1 \dots 4$ play against $5 \dots 8$.

Exploiting this structure, define a general Divide an Conquer approach to compute the schedule of a League of $n = 2^k$ teams.

Sep 15

12. Given a natural number n, we wish to determine whether n is the product of three consecutive natural numbers. For instance, $210 = 5 \cdot 6 \cdot 7$ but 12 cannot be obtained as the product of three consecutive numbers.
    (a) Design a *Divide&Conquer* algorithm with time complexity $O(\log n)$ to solve this problem.
    (b) Justify –posing a recurrence for the time complexity of the algorithm and solving it using the Master Theorem– that your algorithm has indeed $O(\log n)$ complexity.

Feb 16

13. Let $T(1 \dots n)$ be an array of integers representing n temperature values taken at intervals of 1 min. We wish to obtain the shortest time window with below-zero temperatures, i.e., we aim to obtain the indices a and b of the shortest interval $T(a..b)$ (that is, the interval with the least number of elements) of $T(1..n)$ such that $a \leqslant x \leqslant b \Rightarrow T(x) < 0) \wedge (a > 1 \Rightarrow T(a-1) \geqslant 0) \wedge (b < n \Rightarrow T(b+1) \geqslant 0)$. If there is no below-zero temperature recorded then a pair $(0,0)$ can be returned. For example, let $T = [1, 0, -1, -2, -1, 0, -3, -2, -2, -2, 1, 2, 3, 3]$; the solution would be $(3, 5)$.
    (a) Design a *Divide & Conquer* approach to solve this problem.
    (b) Indicate the complexity of the previous algorithm.

Sep 16

14. Assume you get an ordered sequence $a = \langle a_1, \dots, a_n \rangle$ of n different integers, where all elements a_i $(1 \leqslant i \leqslant n)$ have values $1 \leqslant a_i \leqslant m$ for some certain fixed (and known) $m > n$. Create an algorithm of complexity $O(\log n)$ to find the least integer $s \leqslant m$ not present in a.

Feb 17

15. Let A and B be two integer arrays of length m and n respectively, where $m \geqslant n$. Assuming that A is sorted in ascending order and that its elements are all different, we want to determine whether B is a subarray of A (i.e., if all elements of B appear consecutively in A):

    func IsSubarray ($\downarrow A$: ARRAY$[1 \dots m]$ OF $\mathbb{Z}$, $\downarrow B$: ARRAY$[1 \dots n]$ OF $\mathbb{Z}$): $\mathbb{B}$

    A brute-force approach would check B against each possible length-n subarray of A, and therefore its worst-case time complexity would be $O(mn)$, or $\Theta(m)$ if n is a constant.
    (a) Design a Divide-and-Conquer approach for this problem, more efficient that the brute-force approach.
    (b) Pose a recurrence for the computational complexity of the Divide-and-Conquer algorithm and find its asymptotic order of growth using the Master Theorem. Assume n is a constant.
    (c) Discuss how the algorithm would change (if at all) if there could be repeated elements in A.

Feb 17

16. Let A be an ordered array of length n containing elements $1, 2, \cdots n-1$. Hence, there is an element that appears twice. We want to find this element.
    (a) Devise a brute force approach. Which is its complexity?
    (b) Build a Divide&Conquer algorithm better than the previous brute-force algorithm.
    (c) Express the complexity of the previous D&C algorithm using a recurrence and indicate its order of complexity using the Master Theorem.

Feb 2018

17. Given an integer x, its additive inverse is $-x$, e.g. 3 is the additive inverse of -3 and vice versa.
    (a) Design a Divide & Conquer algorithm that given an ascendingly-ordered integer array determine in $O(\log n)$ if a number and its additive inverse are located in adjacent positions.
    (b) Determine (using the Master Theorem) that the algorithm has the desired complexity.

Sep 2018

18. We have an integer array which is circularly sorted, i.e., it would be sorted after a number of shifts to the right or to the left. For example, $[10, 15, 1, 5, 7, 9]$ is a circularly sorted array which would be sorted in ascending order after two shifts to the left or four shifts to the right.

    (a) Find a Divide&Conquer approach of $O(\log n)$ complexity to determine the number of shifts to the left required to have the array sorted in ascending order.

    (b) Define a recurrence for the cost of the previous algorithm and find its asymptotic complexity using the Master Theorem.