

Divide y Conquista

Pedro O. Pérez M., PhD.

Análisis y diseño de algoritmos avanzados
Tecnológico de Monterrey

pperezm@tec.mx

01-2023

① Divide y conquista

Introducción

Algunos ejemplos

Exponenciación rápida

Prefijo común más largo

Contando inversiones

Encontrar el número más cercano en el arreglo

Es una técnica que permite encontrar la solución de un problema descomponiéndolo en subproblemas más pequeños (dividir) y que tienen la misma naturaleza del problema original, es decir, son similares a este. Luego resuelve cada uno de los subproblemas recursivamente hasta llegar a problemas de solución trivial o conocida con antelación (conquistar) para, finalmente, unir las diferentes soluciones (combinar) y así conformar la solución global al problema.

Procedure 1 DIVIDE_AND_CONQUER

Input: X

if X is simple or known **then**

return $SOLUTION(X)$

else

 Decompose X into smaller problems x_1, x_2, \dots, x_n

for $i \leftarrow 1$ **to** n **do**

$y_i \leftarrow DIVIDE_AND_CONQUER(x_i)$

end for

 Combine the y_i to get the Y that is solution of X

return Y

end if

Dados dos números, x y n , calcular el resultado de x^n , haciendo uso de la técnica de dividir y conquistar.

Procedure 2 FAST_POW

Input: $x : \text{Real}, n : \text{Integer}$

if $n < 0$ **then**

return $\text{FAST_POW}(1/x, -n)$

else if $n == 0$ **then**

return 1

else if $n == 1$ **then**

return x

else if $n \bmod 2 = 0$ **then**

return $\text{FAST_POW}(x * x, n/2)$

else if $n \bmod 2 = 1$ **then**

return $x * \text{FAST_POW}(x * x, (n - 1)/2)$

end if

Dado un conjunto de cadenas, encontrar el prefijo común más largo. Por ejemplo, dadas la siguiente cadenas: “geeksforgeeks”, “geeks”, “geek”, “geezer”, el resultado esperado es: “gee”.¹

¹<https://goo.gl/rqjV76>

Procedure 3 FIND_PREFIX

Input: $A : \text{String}, B : \text{String}$

$result \leftarrow ""$

$i \leftarrow 1$

$j \leftarrow 1$

while $i < A.length$ **and** $j < B.length$ **do**

if $A[i] \neq B[j]$ **then**

break

end if

$result \leftarrow result + A[i]$

$i \leftarrow i + 1$

$j \leftarrow j + 1$

end while

return $result$

Procedure 4 COMMON_PREFIX

Input: A : Array, low : Index, $high$: Index

```
if  $low == high$  then
    return  $A[low]$ 
end if
if  $low < high$  then
     $mid \leftarrow FLOOR((high + low)/2)$ 
     $str1 \leftarrow COMMON\_PREFIX(A, low, mid)$ 
     $str2 \leftarrow COMMON\_PREFIX(A, mid + 1, high)$ 
    return  $FIND\_PREFIX(str1, str2)$ 
end if
```

Dado arreglo de números enteros distintos, A , determinar el número de inversiones que existen. Decimos que dos índices $i < j$ forman una inversión si $A[i] > A[j]$.

El número de inversiones de un arreglo indica: a qué distancia (o qué tan cerca) está el arreglo de estar ordenado. Si el arreglo ya está ordenado, el conteo de inversión es 0. Si el arreglo está ordenado en orden inverso, el conteo de inversión es el máximo.

Ejemplo: ¿cuántas inversiones hay en el siguiente arreglo: 2, 4, 1, 3, 5? Tiene tres inversiones (2, 1), (4, 2), (4, 3).²

²<https://goo.gl/DxqxBe>

Procedure 5 SORT_AND_COUNT

Input: A, B : Array, $low, high$: Index

$r \leftarrow 0$

$left \leftarrow 0$

$right \leftarrow 0$

if $(high - low + 1 = 1)$ **then**

return 0

else

$mid \leftarrow \text{FLOOR}((high + low)/2)$

$left \leftarrow \text{SORT_AND_COUNT}(A, B, low, mid)$

$right \leftarrow \text{SORT_AND_COUNT}(A, B, mid + 1, high)$

$r \leftarrow \text{MERGE_AND_COUNT}(A, B, low, mid, high)$

$\text{COPY}(A, B, low, high)$

end if

return $r + left + right$

Procedure 6 MERGE_AND_COUNT

Input: A, B : Array, $low, mid, high$: Index

...

while $left \leq mid$ AND $right \leq high$ **do**

if $A[left] < A[right]$ **then**

$B[i] \leftarrow A[left]$

$left \leftarrow left + 1$

else

$B[j] \leftarrow A[right]$

$right \leftarrow right + 1$

$count \leftarrow count + (mid - left)$

end if

$i \leftarrow i + 1$

end while

...

return $count$

Encontrar el número más cercano en el arreglo

Dado un arreglo de números enteros ordenados. Necesitamos encontrar el valor más cercano al número dado. El arreglo puede contener valores duplicados y números negativos.³

³<https://goo.gl/XTgBzX>

Procedure 7 GET_CLOSEST

Input: $val1, val2, target : Key$

if $(target - val1) > (val2 - target)$ then

return $val2$

else

return $val1$

end if

Procedure 8 FIND_CLOSEST

Input: A : Array, $target$: Key

if $target < A[1]$ then

return $A[1]$

end if

if $target > A[A.length]$ then

return $A[A.length]$

end if

$low \leftarrow 1$

$high \leftarrow A.length$

while $low < high$ do

NEXT SLIDE

end while

```
mid  $\leftarrow$  FLOOR((high + low)/2)
if target = A[mid] then
    return A[mid]
else if target < A[mid] then
    if mid > 0 and target > A[mid - 1] then
        return GET_CLOSEST(A[mid - 1], A[mid], target)
    end if
    high  $\leftarrow$  mid
else
    if mid < A.length and target < A[mid + 1] then
        return GET_CLOSEST(A[mid], A[mid + 1], target)
    end if
    low  $\leftarrow$  mid
end if
```
