

Salvador García González

Ejercicio 1)

119718

- 1) Formaliza el enunciado para definir el problema del subarreglo máximo que cruza el punto medio.

FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)

Sea $A = [A[1], A[2], \dots, A[n]]$ un arreglo de tamaño $n \geq 1$ tal que cada $A[i] \in \mathbb{R}$. para $i=1, \dots, n$. Sean $\underline{\text{low}}$ y $\underline{\text{high}}$ dos escalares tales que $1 \leq \underline{\text{low}} \leq \underline{\text{high}} \leq n$.

El problema del subarreglo máximo que cruza el punto medio busca encontrar un subarreglo $[A[\alpha], \dots, A[\beta], \dots, A[\gamma]]$ tal que $\sum_{i=\alpha}^{\gamma} A[i]$ sea la máxima sumatoria de elementos contiguos, tal que β se define como el elemento en el punto medio del subarreglo y $\underline{\text{low}} \leq \alpha \leq \beta \leq \gamma \leq \underline{\text{high}}$

FIND-MAXIMUM-SUBARRAY($A, low, high$)

```

1 if high == low
2   return (low, high, A[low])           // base case: only one element
3 else mid = [(low + high)/2]
4   (left-low, left-high, left-sum) =
      FIND-MAXIMUM-SUBARRAY(A, low, mid)
5   (right-low, right-high, right-sum) =
      FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
6   (cross-low, cross-high, cross-sum) =
      FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
7   if left-sum >= right-sum and left-sum >= cross-sum
8     return (left-low, left-high, left-sum)
9   elseif right-sum >= left-sum and right-sum >= cross-sum
10    return (right-low, right-high, right-sum)
11 else return (cross-low, cross-high, cross-sum)

```

]
 Caso 1 elemento (base) $\underline{\text{low}} = \underline{\text{high}}$
] Seleccionar punto medio $\underline{\text{mid}} = \beta$

]
 Caso que cruza el punto medio

FIND-MAX-CROSSING-SUBARRAY($A, low, mid, high$)

```

1 left-sum = -∞
2 sum = 0
3 for i = mid downto low
4   sum = sum + A[i]
5   if sum > left-sum
6     left-sum = sum
7     max-left = i
8 right-sum = -∞
9 sum = 0
10 for j = mid + 1 to high
11   sum = sum + A[j]
12   if sum > right-sum
13     right-sum = sum
14     max-right = j
15 return (max-left, max-right, left-sum + right-sum)

```

]
 Encontramos el índice α y la sumatoria
 máxima $\sum_{i=\alpha}^{\beta} A[i]$ correspondiente

]
 Encontramos el índice γ y la sumatoria
 máxima $\sum_{i=\beta}^{\gamma} A[i]$ correspondiente

Ejercicio 2)

Construye un algoritmo de tiempo lineal para el problema de Subarreglo máximo

```
Find-max-subarray-linear(A)
    i_bajo = 0
    i_alto = 0
    sum_actual = 0
    sum_max = 0
    for i=0 to A.length-1 do
        sum_actual = max(sum_actual+A[i], A[i])
        if sum_actual > sum_max
            i_alto = i
            sum_max = max(sum_actual, max)
        if A[i] ≥ sum_actual
            i_bajo = i
    return (sum_max, i_bajo, i_alto)
```

Las líneas 1 a 5 se ejecutan 1 sola vez, mientras que las líneas adentro del for (7 a 12) se ejecutan a lo más n veces, por lo que concluimos que el tiempo de ejecución de este programa es lineal

Ejercicio 3)

Demuestra como multiplicar los números complejos $a+bi$ y $c+di$ usando únicamente 3 multiplicaciones de números reales. Tu algoritmo debe tomar como entrada los 4 números reales que representan los dos complejos y dar como resultado la parte real $ac-bd$ y la imaginaria $ad+bc$.

La forma convencional de multiplicar números complejos es la siguiente:

$$= (a+bi)(c+di)$$

= $ac + adi + cb i - bd$ Tenemos 4 términos. Para lograrlo solo con

$$=(ac - bd) + (cb + ad)i \quad 3 \text{ multiplicaciones:}$$

funcion mult_comp(a,b,c,d)

$$P_1 = ac$$

$$P_2 = bd$$

$$\text{sum1} = a+b$$

$$\text{sum2} = c+d$$

$$P_3 = (\text{sum1})(\text{sum2})$$

$$\text{Real} = P_1 - P_2 \Rightarrow ac - bd$$

$$\text{imaginario} = P_3 - P_1 - P_2 \Rightarrow (a+b)(c+d) - ac - bd \\ = ac + ad + bc + bd - ac - bd = ad + bc$$

return (real,imaginario)

Ejercicio 4

Demuestra que la solución para la recurrencia

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(1) \quad \text{es } T(n) = \log n$$

Utilizamos el teorema maestro y consideramos lo siguiente:

$$a=1$$

$$b=2$$

$$\log_b(a) = \log_2(1) = 0$$

$$f(n) = \Theta(1)$$

$n^{\log_b a} = n^0 = 1 \Rightarrow$ Nos encontramos en el caso 2, ya que

$$f(n) = \Theta(n^{\log_b a}) = \Theta(1) \quad \text{entonces } T(n) = \Theta(n^{\log_b a} \log n)$$

$$= \Theta(n^0 \log n) = \Theta(\log n)$$

Ejercicio 5

Da una cota asintótica para la recurrencia

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2 \log n$$

$$a=4$$

$$f(n) = n^2 \log n$$

$$b=2$$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

Utilizamos la versión extendida del teorema maestro

Master Theorem

The Master Theorem applies to recurrences of the following form:

$$T(n) = aT(n/b) + f(n)$$

where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.

There are 3 cases:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a} \log^k n)$ with $k \geq 0$, then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ with $\epsilon > 0$, and $f(n)$ satisfies the regularity condition, then $T(n) = \Theta(f(n))$.
Regularity condition: $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n .

el caso 2 $f(n) = n^2 \log n \quad k=1$

$$f(n) = \Theta(n^{\log_b a} \log^k n) \quad \text{Por TM, } T(n) = \Theta(n^2 \log^2 n)$$