

# Febrero2016-Resuelto.pdf



alberto\_fm\_



Algorítmica y Modelos de Computación



3º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingeniería  
Universidad de Huelva



**WUOLAH Print**

Lo que faltaba en Wuolah



Imprimir



## Ejercicio 1. (2 puntos).

- Analizar el algoritmo de la Búsqueda del k-ésimo menor elemento. Dado un vector de n elementos, el problema de la selección consiste en buscar el k-ésimo menor elemento.
- Supongamos que disponemos de la siguiente definición de tipo:  
 CONST n = ...;  
 TYPE vector = ARRAY [1..N] OF INTEGER;  
 Y supongamos que primero y último indican los límites del array (initialmente primero=1 y último=n)
- Para la solución del problema utilizamos la idea del algoritmo **Partition** (utilizado en Quicksort): El vector A[p..r] se partitiona (reorganiza) en dos subvectores A[p..q] y A[q+1..r] de forma que los elementos de A[p..q] son menores iguales que el pivote (por ej: primer elemento) y los de A[q+1..r] mayores o iguales.

```
int función Partition (A:vector; primero, ultimo: int)
    piv = A[ primero ]; i = primero-1; j = ultimo+1;
    mientras j >= i hacer
        mientras A[ j ] >= piv hacer
            j = j - 1;
        fmientras
        mientras A[ j ] <= piv hacer
            i = i + 1;
        fmientras
        si i < j entonces /* A[ i ] <= A[ j ] */
            temp = A[ j ]; A[ j ] = A[ i ]; A[ i ] = temp;
        fsi
    fmientras
    devuelve j; /* retorna el índice para la división (partición) */
ffuncion Partition
```

- El algoritmo de la Búsqueda del k-ésimo menor elemento puede ser implementado:

1. Versión iterativa de la **Búsqueda del k-ésimo menor elemento** puede ser implementado:

```
función SelectIt(A : vector, primero, ultimo, k : entero)
    mientras (primero < ultimo) hacer
        q = Partition(A, primero, ultimo)
        si (k ≤ q) entonces
            ultimo = q
        sino
            primero = q + 1
        fsi
    fmientras
    devuelve A[primero]
ffuncion
```

2. Versión recursiva de la **Búsqueda del k-ésimo menor elemento**

```
función SelectRc(A : vector, primero, ultimo, k : entero)
    si (primero == ultimo) entonces
        devuelve A[primero]
    fsi
    q = Partition(A, primero, ultimo)
    i = q - primero + 1
    si (k ≤ i) entonces
        devuelve SelectRc(A, primero, q, k)
    sino
        devuelve SelectRc(A, q+1, ultimo, k-i)
    fsi
ffuncion
```

- Se pide:

- a. (0,5 puntos). Calcular la complejidad del algoritmo **iterativo** propuesto para el **caso promedio** mediante el **conteo de número de operaciones elementales**.
- b. (0,5 puntos). Calcular la complejidad del algoritmo **recursivo** propuesto para el **caso promedio** por el método de la **ecuación característica**.
- c. (0,5 puntos). Calcular la complejidad del algoritmo **recursivo** propuesto para el **caso promedio** por el **Teorema maestro**.
- d. (0,5 puntos). Comprobar si ambas versiones, iterativa y recursiva, invierten el mismo tiempo.

$$a) T(n) = 1 + \sum_{i=1}^n (1 + 1 + 2 + 1 + T_{\text{partition}}) + 2$$

$$T_{\text{partition}} = 7 + \sum_{i=1}^{\frac{n}{2}} (1 + 2 + \sum_{j=1}^{\frac{n}{2}} (5) + \sum_{j=\frac{n}{2}+1}^{n} (5) +$$

$$8 + 1 + 2) + 1 =$$

$$= 8 + \frac{1}{4} + \frac{5}{2}n + \frac{5}{2}n =$$

$$22 + n \cdot 5 \in O(n)$$

En cada iteración, el tamaño se divide a la mitad  
por tanto  $\frac{n}{2} = \frac{1}{2} \rightarrow i = \log(n)$

$$T(n) = 1 + \sum_{i=1}^{\log(n)} (1 + 1 + 2 + 1 + 22 + 5n) + 2 =$$

$$T(n) = 3 + \sum_{i=1}^{\log(n)} (2 + 5(\frac{n}{2^i})) =$$

$$= 3 + 2 \log(n) + 5n \left( -\frac{1}{n} + 2 \right) =$$

```
función SelectIt(A : vector, primero, ultimo, k : entero)
    mientras (primero < ultimo) hacer
        q = Partition(A, primero, ultimo)
        si (k ≤ q) entonces
            ultimo = q
        sino
            primero = q + 1
        fsi
    fmientras
    devuelve A[primero]
```

```
int función Partition (A:vector; primero, ultimo: int)
    piv = A[ primero ]; i = primero-1; j = ultimo+1;
    mientras j >= i hacer
        mientras A[ j ] >= piv hacer
            j = j - 1;
        fmientras
        mientras A[ j ] <= piv hacer
            i = i + 1;
        fmientras
    fmientras
```

$$= 3 + 2\log(n) + 5n \left( -\frac{1}{n} + 2^{\frac{n}{2}} \right) =$$

$$\sum_{i=3}^{\log n} \frac{1}{2^i} = \sum_{i=1}^{\log n} \left(\frac{1}{2}\right)^i = \frac{\left(\frac{1}{2}\right)^{\log n} - 1}{\frac{1}{2} - 1} =$$

$$\left(\frac{1}{2}\right)^{\log n} = \left(\frac{1}{2}\right)^{\log n} \cdot \frac{1}{2} = \frac{1}{n} \cdot \frac{1}{2} = \frac{1}{2n}$$

$$= \frac{\frac{1}{2n} - 1 - \frac{1}{2} - \frac{1-2n}{2n}}{-1/2} = \frac{\frac{1-2n}{2n}}{-1/2} = \frac{2-4n}{-2n} = -\frac{1}{n} + 2$$

```

j = j - 1; ←
fmientras i
mientras A[j] <= piv hacer
    i = i + 1; ←
fmientras i
    si i < j entonces /* A[i] <= A[j] */
        temp = A[j]; A[j] = A[i]; A[i] = temp;
    fsi
fmiemtras devuelve j; /* retorna el índice para la división (partición) */
ffuncion Partition

```

$$T(n) = 3 + 2\log(n) + 5n \left( -\frac{1}{n} + 2 \right) =$$

$$3 + 2\log(n) + -5 + 10n = 2\log(n) + 10n - 2 \in O(n)$$

b) Versión recursiva de la Búsqueda del k-ésimo menor elemento

```

funcion SelectRc(A : vector, primero, ultimo, k : entero)
    si (primero == ultimo) entonces
        devuelve A[primero]
    fsi
    q = Partition(A, primero, ultimo)
    i = q - primero + 1
    si (k ≤ i) entonces
        devuelve SelectRc(A, primero, q, k)
    sino
        devuelve SelectRc(A, q+1, ultimo, k-i)
    fsi
ffuncion

```

La ecuación recurrente será:

$$T(n) = \begin{cases} 3 & n \leq 1 \\ 2 + T_{\text{partition}} + 3 + 1 + T(n/2) & \end{cases} \Rightarrow$$

$$\Rightarrow T(n) = \begin{cases} 3 & n \leq 1 \\ 28 + 5n + T(n/2) & n > 1 \end{cases}$$

$n = 2^k \Leftrightarrow k = \log_2 n$

$$T(n) = 28 + 5n + T(n/2) \rightarrow T(n) - T(n/2) = 5n + 28 \rightarrow \text{NO HOMOGENEA}$$

$$T(2^k) - T(2^{k-1}) = 5 \cdot 2^k + 28 \rightarrow T(2^k) = \underbrace{k \cdot 2^k}_{\substack{f_1: 1 \\ f_2: 2}} + 28 \rightarrow$$

$$\rightarrow (x-1)(x-2)(x-1) = 0 \rightarrow \begin{cases} x_1: 1 & (\text{doble}) \\ x_2: 2 \end{cases}$$

NEW

# WUOLAH Print

Lo que faltaba en Wuolah



Imprimir



- Todos los apuntes que necesitas están aquí
- Al mejor precio del mercado, desde **2 cent.**
- Recoge los apuntes en tu copistería más cercana o recíbelos en tu casa
- Todas las anteriores son correctas



$$t_K = 1^K \cdot K^0 \cdot C_1 + 1^K \cdot K^1 \cdot C_{12} + 2^K \cdot K^0 \cdot C_2 \Rightarrow$$

$$T(n) = C_1 + \log_2(n) C_{12} + n C_2 \in O(n)$$

$$T(1) = 3$$

$$T(2) = T(1) + 5 + 28 = 36$$

$$T(4) = T(2) + 10 + 28 = 74$$

$$T(8) = T(4) + 40 + 28 = 142$$

$$\begin{cases} C_1 + \log_2(2) C_{12} + 2C_2 = 36 \\ C_1 + \log_2(4) C_{12} + 4C_2 = 74 \\ C_1 + \log_2(8) C_{12} + 8C_2 = 142 \end{cases}$$

$$\begin{cases} C_1 + C_{12} + 2C_2 = 36 \\ C_1 + 2C_{12} + 4C_2 = 74 \\ C_1 + 3C_{12} + 8C_2 = 142 \end{cases}$$

$$\left( \begin{array}{cccc} 1 & 1 & 2 & 36 \\ 1 & 2 & 4 & 74 \\ 1 & 3 & 8 & 142 \end{array} \right) \xrightarrow[F_2 - F_1]{F_3 - F_1} \left( \begin{array}{cccc} 1 & 1 & 2 & 36 \\ 0 & 1 & 2 & 38 \\ 0 & 2 & 6 & 106 \end{array} \right) \rightarrow$$

$$\xrightarrow[F_3 - 2F_2]{F_3 - 2F_2} \left( \begin{array}{cccc} 1 & 1 & 2 & 36 \\ 0 & 1 & 2 & 38 \\ 0 & 0 & 2 & 30 \end{array} \right) \xrightarrow[F_3 \leftarrow \frac{1}{2}F_3]{F_3 \leftarrow \frac{1}{2}F_3} \left( \begin{array}{cccc} 1 & 1 & 2 & 36 \\ 0 & 1 & 2 & 38 \\ 0 & 0 & 1 & 15 \end{array} \right)$$

$$\xrightarrow[F_1 - F_2]{F_1 - F_2} \left( \begin{array}{cccc} 1 & 0 & 0 & -2 \\ 0 & 1 & 2 & 38 \\ 0 & 0 & 1 & 15 \end{array} \right) \xrightarrow[F_2 - 2F_3]{F_2 - 2F_3} \left( \begin{array}{cccc} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 15 \end{array} \right) \begin{cases} C_{11} = -2 \\ C_{12} = 8 \\ C_2 = 15 \end{cases}$$

$$T(n) = -2 + 8\log_2(n) + 15n \in O(n).$$

c) El Teorema Maestro es:

$$T(n) \in \begin{cases} n^{\log_b a} & a > b^k \\ n^k \cdot \log^{p+1}(n) & a = b^k \\ n^k \cdot \log^p(n) & a < b^k \end{cases}$$

$\left. \begin{array}{l} a=1 \\ b=2 \\ k=1 \\ p=0 \end{array} \right\} T(n) \in O(n \cdot \log(n)) \Rightarrow \\ T(n) \in O(n);$

$$d) \text{ Recursivo: } 8\log_2(n) + 15n - 2 \in O(n) = f(x)$$

d) Recursivo:  $8 \log(n) + 15n - 2 \in O(n) = f(x)$

Iterativo:  $27 \log(n) + 10n - 2 \in O(n) = g(x)$

$f(x) - g(x) > 0 \rightarrow f(x)$  crece más rápido y, por tanto, consume más tiempo.

#### Ejercicio 2. (3 pts)

➤ Resolver el problema de la mochila para el caso en que no se permite partir los objetos (es decir, un objeto se coge entero o no se coge nada).

□ Problema de la mochila:

- Tenemos:
  - $n$  objetos, cada uno con un peso ( $p_i$ ) y un beneficio ( $b_i$ ).
  - Una mochila en la que podemos meter objetos, con una capacidad de peso máximo  $M$ .
- Objetivo: llenar la mochila con esos objetos, maximizando la suma de los beneficios (valores) transportados, y respetando la limitación dada por la capacidad máxima  $M$ .
- Se supondrá que los objetos NO se pueden partir en trozos.

➤ Se pide:

- (1.5 pts). Diseñar un algoritmo voraz para resolver el problema aunque no se garantice la solución óptima. Es necesario marcar en el código propuesto a qué corresponde cada parte en el esquema general de un algoritmo voraz (criterio, candidatos, función, ...). Si hay más de un criterio posible, elegir uno razonadamente y discutir los otros. Comprobar si el algoritmo garantiza la solución óptima en este caso (la demostración se puede hacer con un contraejemplo).
- Aplicar el algoritmo al caso:  $n = 3, M = 6, p = (2, 3, 4), b = (1, 2, 5)$ .
- (1.5 pts). Resolver el problema mediante programación dinámica. Definir la ecuación recurrente, los casos base, las tablas y el algoritmo para rellenarlas y especificar cómo se recomponen la solución final a partir de los valores de las tablas.
- Aplicar el algoritmo al caso:  $n = 3, M = 6, p = (2, 3, 4), b = (1, 2, 5)$ .

➤ NOTA: una posible ecuación recurrente es:

$$\text{Mochila}(k, m) = \begin{cases} 0 & \text{Si } k=0 \text{ ó } m=0 \\ -\infty & \text{Si } k<0 \text{ ó } m<0 \\ \max \{\text{Mochila}(k-1, m), b_k + \text{Mochila}(k-1, m-p_k)\} & \text{Otros casos} \end{cases}$$

a)

funcion MochilaV ( $p, b$ : array [1..N] of integer,  $M$ : integer)

$X = \{0, 0, 0\};$

ordenarArculos(); // Criterio: De menor a mayor cociente  $b/p$

peso  $\leftarrow 0$ ;

$i \leftarrow 1$ ;

while ( $i \leq N$ ) hacer

si ( $peso + p[i] \leq M$ ) entonces

peso  $\leftarrow peso + p[i]$

$X[i] = 1$ ;

fin si

$i \leftarrow i + 1$ ;

fin while

devolver  $X$ ;

fin función;

El criterio seleccionado ha sido ordenar los objetos de menor a mayor cociente  $beneficio/peso$ .

El algoritmo no garantiza la solución óptima. Podemos verlo con el siguiente ejemplo:

$n=3$

7

Sigamos el ejemplo:

$$\left. \begin{array}{l} m=3 \\ M=6 \\ p=(2, 3, 3) \\ b=(1, 5, 6) \\ b/p=(0.5, 1.6, 2) \end{array} \right\} \text{Ordenemos según el criterio}$$

$$p(2, 4, 2)$$

$$\rightarrow b(1, 5, 6)$$

$$\left. \begin{array}{ll} X \rightarrow \langle 1, 0, 0 \rangle & X \rightarrow \langle 1, 1, 0 \rangle \\ \text{peso} = 2 & \text{peso} = 6 \\ \text{beneficio} = 1 & \text{beneficio} = 6 \end{array} \right\} \text{No sería la solución óptima. } (X \rightarrow \langle 0, 1, 1 \rangle) \\ \text{peso} = 6; \\ \text{beneficio} = 5;$$

Aplicamos el algoritmo:

$$\begin{aligned} M=6, n=3 \\ b=(1, 2, 5) \\ p=(2, 3, 4) \\ b/p=(0.5, 0.6, 1.25) \end{aligned}$$

$$\left. \begin{array}{ll} X \rightarrow \langle 1, 0, 0 \rangle & X \rightarrow \langle 1, 1, 0 \rangle \\ \text{benef} = 1 & \text{benef} = 3 \\ \text{peso} = 2 & \text{peso} = 5 \end{array} \right\} \text{No genera la solución óptima} \\ (X \rightarrow \langle 1, 0, 1 \rangle \\ \text{peso} = 6 \\ \text{benef} = 6)$$

b) La ecuación recurrente será:

$$\text{Mochila}(k, m) = \begin{cases} 0 & k=0, m=0 \\ -\infty & k<0, m<0 \\ \max(\text{Mochila}(k-1, m), \text{Mochila}(k-1, m-p_k) + b_k) & \text{CASOS BASE} \end{cases}$$

La tabla será:

$$T[N][M];$$

El algoritmo es el siguiente:

```

función mochilaPD(N: integer; b, p: array [1..N] of integer)
    para i=1 hasta N hacer T[i][0]=0; fin para;
    para j=0 hasta M hacer T[0][j]=0; fin para;
    para i=1 hasta N hacer
        para j=1 hasta M hacer
            si ( j < p[i] )
                T[i][j]= T[i-1][j]
            sino
                T[i][j]= max(T[i-1][j], T[i-1][j-p[i]] + b[i]);
            fin si;
        fin para;
    fin para;
    
```

b<sup>n</sup>

$$T[i][j] = \max(T[i-1][j], T[i-1][j - p[i]] + b[i]);$$

f*i*

f*j* para

f*n* i

Aplicando el algoritmo al ejemplo nos queda:

<i>i</i>	0	1	2	3	4	5	6
<i>n</i> =3	0	0	0	0	0	0	0
<i>M</i> =6	1	0	0	1	1	1	1
<i>p</i> =(2,3,4)	2	0	0	1	2	2	3
<i>b</i> =(1,2,5)	3	0	0	1	2	5	5
							<u>6</u>

Para recomponer la solución accedemos a la última posición de la tabla  $T[N][M]=T[3][6]=6$ .

- lo comparamos con el anterior:  $T[2][6]=3$ .

Como son distintos, significa que hemos cogido el elemento *i*.

Añelitamos *j* restándole el peso del artículo que hemos cogido:

$$j = j - p[i]. \rightarrow j = 6 - 3$$

Decrementamos *i* (*i*=2) y volvemos a comparar:

$$T[i][j] == T[i-1][j]$$

Como  $T[2][3] != T[1][3] \rightarrow$  cogemos el objeto *i*=2 y seguimos iterando:

$$j = j - p[i] \rightarrow j = 3 - 3$$

Decrementamos *i* (*i*=1) y comparamos:

$$T[1][0] == T[0][0] \text{ por lo que no lo cogemos.}$$

El vector Solución queda de la siguiente forma:

$$X = 1, 0, 1, 1$$

El algoritmo que describe como obtener la solución es:

```
function Recomponer(M:Integer, b, p[1..n]: Integer, T[0..N][0..M])
    var x[1..N] of Integer;
        j:= M;
    para i=N hasta 1 hacer
        si T[i][j] == T[i-1][j]
            x[i]:=0;
        Sino
            x[i]:=1;
            j:= j - p[i];
        fsi
    fpara
function
```

**Ejercicio\_3. (2 pts)**

- Dado el AFND =  $\{(a, b, c), \{p, q, r, s, t, u, v\}, f, p, \{\}\}$  donde  $f$  viene dado por la siguiente tabla de transiciones:

$f$	a	b	c	$\lambda$
$\rightarrow p$				$\{q, t\}$
q		$\{r, s\}$		$\{r, s\}$
r				$\{q, u\}$
s	$\{t, p\}$		$\{u\}$	
t		$\{v\}$		$\{q\}$
u	$\{q, s\}$		$\{v\}$	$\{s\}$
* v				$\{r\}$

➤ Se pide:

- (0,25 puntos). Si son aceptadas o no por el autómata las siguientes cadenas:
  - $f(p, bbcc)$
  - $f(p, acbcac)$
  - $f(p, bcacaa)$
  - $f(p, caa)$
  - $f(p, abac)$
- (0,5 puntos). El AFD equivalente.
- (0,5 puntos). El AFD mínimo.
- (0,25 puntos). Corroborar el resultado obtenido para las palabras del apartado a con el AFD obtenido en el apartado c.
- (0,5 puntos). Obtener una expresión regular equivalente al AFD obtenido en el apartado c.

a.)

$$1) f'(p, bbcc) = \{p, q, t, r, s, u\}$$

$$f'(\{p, q, t, r, s, u\}, b) = \{r, s, v, q, u\}$$

$$f'(\{r, s, v, q, u\}, b) = \{r, s, q, u\}$$

$$f'(\{r, s, q, u\}, c) = \{u, v, s, r, q\}$$

$$f'(\{u, v, s, r, q\}, c) = \{v, u, r, s, q\}$$

$$v \in \{v, u, r, s, q\} \rightarrow \underline{\text{ACEPTADA}}$$

$$2) f'(p, acbcac) = f'(CL(p), acbcac) = \{p, q, t, r, s, u\}$$

$$f'(\{p, q, t, r, s, u\}, a) = \{t, p, q, s, r, u\}$$

$$f'(\{t, p, q, s, r, u\}, c) = \{u, v, s, r, q\}$$

$$f'(\{u, v, s, r, q\}, b) = \{r, s, q, u\}$$

$$f'(\{r, s, q, u\}, c) = \{u, v, s, r, q\}$$

$$f'(\{u, v, s, r, q\}, a) = \{t, p, q, s, r, u\}$$

$$f'(\{t, p, q, s, r, u\}, c) = \{u, v, s, r, q\}$$

$$v \in \{u, v, s, r, q\} \rightarrow \underline{\text{ACEPTADA}}$$

$$3) f'(p, bcacaa) = f'(CL(p), bcacaa) =$$

$$= f'(\{p, q, t, r, s, u\}, b) = \{r, s, v, q, u\}$$

$$f'(\{r, s, v, q, u\}, c) = \{u, v, s, r, q\}$$

$$f'(\{u, v, s, r, q\}, a) = \{t, p, q, s, r, u\}$$

$$f'(\{t, p, q, s, r, u\}, c) = \{u, v, s, r, q\}$$

$$f'(\{u, v, s, r, q\}, a) = \{t, p, q, s, r, u\}$$

$$f'(\{t, p, q, s, r, u\}, a) = \{t, p, q, s, r, u\}$$

$$v \notin \{t, p, q, s, r, u\} \rightarrow \underline{\text{RECHAZADA}}$$

$$4) f'(p, caa) = f'(CL(p), caa)$$

$$f'(\{p, q, t, r, s, u\}, c) = \{u, v, s, r, q\}$$

$$f'(\{u, v, s, r, q\}, a) = \{t, p, q, s, r, u\}$$

$$f'(1t, p, q, s, r, u, v) = \{t, p, q, s, r, u\}$$

$v \in \{t, p, q, s, r, u\} \rightarrow \text{RECHAZADA}$

$$5) f'(p, abac) = f'(CL(p), abac)$$

$$f'(\{p, q, t, r, s, u\}, a) = \{t, p, q, s, r, u\}$$

$$f'(\{t, p, q, s, r, u\}, b) = \{r, s, v, q, u\}$$

$$f'(\{r, s, v, q, u\}, a) = \{t, p, q, s, r, u\}$$

$$f'(\{t, p, q, s, r, u\}, c) = \{u, v, s, r, q\}$$

ver  $v, r, s, t, q \rightarrow \text{ACEPTADA}$

b) Para construir el AFD equivalente usaremos una tabla:

	a	b	c
$\rightarrow Q_0$	$Q_0$	$Q_1$	$Q_2$
$\overset{*}{Q}_1$	$Q_0$	$Q_2$	$Q_1$
$Q_2$	$Q_0$	$Q_2$	$Q_1$

$Q_0 = CL(p) = \{p, q, t, r, s, u\}$   
 $f'(Q_0, a) = \{t, p, q, s, r, u\} = Q_0$   
 $f'(Q_0, b) = \{r, s, v, q, u\} = \overset{*}{Q}_1$   
 $f'(Q_0, c) = \{u, v, s, r, q\} = Q_2$   
  
 $f'(Q_1, a) = \{t, p, q, s, r, u\} = Q_0$   
 $f'(Q_1, b) = \{r, s, v, q, u\} = Q_2$   
 $f'(Q_1, c) = \{u, v, s, r, q\} = Q_1$   
  
 $f'(Q_2, a) = \{t, p, q, s, r, u\} = Q_0$   
 $f'(Q_2, b) = \{r, s, v, q, u\} = Q_2$   
 $f'(Q_2, c) = \{u, v, s, r, q\} = Q_1$

c) El AFD mínimo se obtiene mediante el algoritmo Conjunto-colectante:

$$E/Q_0 = (C_0 = \{Q_0, Q_2\}, C_1 = \{Q_1\})$$

$$\begin{cases} f(Q_0, a) = C_0; & f(Q_0, b) = \underline{C_1}; & f(Q_0, c) = \underline{C_1} \\ f(Q_2, a) = C_0; & f(Q_2, b) = \underline{C_0}; & f(Q_2, c) = \underline{C_0} \end{cases}$$

Como no coinciden sus transiciones  $\rightarrow$  Creamos un nuevo conjunto.

$$E/Q_1 = (C_0 = \{Q_0\}, C_1 = \{Q_1\}, C_2 = \{Q_2\})$$

Por lo tanto, ya tenemos el AFD mínimo en el apartado anterior.

	a	b	c
$\rightarrow Q_0$	$Q_0$	$Q_1$	$Q_2$
$\overset{*}{Q}_1$	$Q_0$	$Q_2$	$Q_1$
$Q_2$	$Q_0$	$Q_0$	$Q_1$

d)

$$1) f'(Q_0, bbcc) =$$

$$f'(Q_0, b) = Q_1$$

$$f'(Q_1, b) = Q_2$$

$$f'(Q_2, c) = Q_1$$

$$f'(Q_1, c) = Q_1$$

$Q_1 \in F \rightarrow \text{ACEPTADA}$

$$2) f'(Q_0, acbcac) =$$

$$f'(Q_0, a) = Q_0$$

$$f'(Q_0, c) = Q_1$$

2)  $f'(Q_0, abc\bar{c}ac) =$

$$\begin{aligned} f'(Q_0, a) &= Q_0 \\ f'(Q_0, c) &= Q_1 \\ f'(Q_1, b) &= Q_2 \\ f'(Q_2, c) &= Q_1 \\ f'(Q_1, a) &= Q_0 \\ f'(Q_0, c) &= Q_2 \in F \rightarrow \text{ACEPTADA} \end{aligned}$$

3)  $f'(Q_0, bc\bar{a}c\bar{a}a) =$

$$\begin{aligned} f'(Q_0, b) &= Q_1 \\ f'(Q_1, c) &= Q_2 \\ f'(Q_2, a) &= Q_0 \\ f'(Q_0, c) &= Q_1 \\ f'(Q_1, a) &= Q_0 \\ f'(Q_0, a) &= Q_0 \notin F \rightarrow \text{RECHAZADA} \end{aligned}$$

4)  $f'(Q_0, caa)$

$$\begin{aligned} f'(Q_0, c) &= Q_1 \\ f'(Q_1, a) &= Q_0 \\ f'(Q_0, a) &= Q_0 \notin F \rightarrow \text{RECHAZADA} \end{aligned}$$

5)  $f'(Q_0, ab\bar{a}c)$

$$\begin{aligned} f'(Q_0, a) &= Q_0 \\ f'(Q_0, b) &= Q_1 \\ f'(Q_1, a) &= Q_0 \\ f'(Q_0, c) &= Q_1 \in F \rightarrow \text{ACEPTADA} \end{aligned}$$

e)

	a	b	c
Q <sub>0</sub>	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>1</sub>
Q <sub>1</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>2</sub>
Q <sub>2</sub>	Q <sub>0</sub>	Q <sub>0</sub>	Q <sub>1</sub>

$$\begin{cases} x_0 = ax_0 + bx_1 + cx_1 + b + c \\ x_1 = ax_0 + bx_2 + cx_1 + c \\ x_2 = ax_0 + bx_2 + cx_1 + c \end{cases}$$

$$X = aX + \beta \Leftrightarrow X = a^* \beta$$

$$\rightarrow \begin{cases} x_0 = ax_0 + (bx_1 + cx_1 + b + c) \rightarrow x_0 = a^*(bx_1 + cx_1 + b + c) \\ x_1 = cx_1 + (ax_0 + bx_2 + c) \rightarrow x_1 = c^*(ax_0 + bx_2 + c) \\ x_2 = bx_2 + (ax_0 + cx_1 + c) \rightarrow x_2 = b^*(ax_0 + cx_1 + c) \end{cases}$$

$$\begin{cases} x_0 = a^*bx_1 + a^*x_1 + a^*b + a^*c \\ x_1 = c^*ax_0 + c^*bx_2 + c^*c \\ x_2 = b^*ax_0 + b^*cx_1 + b^*c \end{cases} \rightarrow$$

$$x_1 = c^*ax_0 + c^*b(b^*ax_0 + b^*cx_1 + b^*c) + c^*c \rightarrow$$

$$\rightarrow x_1 = c^*ax_0 + c^*bb^*ax_0 + c^*bb^*cx_1 + c^*bb^*c + c^*c \rightarrow$$

$$\rightarrow x_1 = c^*bb^*cx_1 + (c^*ax_0 + c^*bb^*ax_0 + c^*bb^*c + c^*c) \rightarrow$$

$$\begin{aligned}
 \rightarrow x_1 &= (c^*bb^*c)^* (c^*ax_0 + c^*bb^*ax_0 + c^*bb^*c + c^*c) \\
 &\quad + \\
 x_0 &= a^*b((c^*bb^*c)^* (c^*ax_0 + c^*bb^*ax_0 + c^*bb^*c + c^*c)) + \\
 &\quad a^*((c^*bb^*c)^* (c^*ax_0 + c^*bb^*ax_0 + c^*bb^*c + c^*c)) + \\
 &\quad a^*b + a^*c \rightarrow \\
 x_0 &= a^*b (c^*bb^*c)^* (c^*ax_0 + c^*bb^*ax_0 + c^*bb^*c + c^*c) + \\
 &\quad a^* (c^*bb^*c)^* (c^*ax_0 + c^*bb^*ax_0 + c^*bb^*c + c^*c) + \\
 &\quad a^*b + a^*c \rightarrow \\
 x_0 &= a^*b(c^*bb^*c)^* ((c^*a + c^*bb^*a)x_0 + c^*bb^*c + c^*c) + \\
 &\quad a^*(c^*bb^*c)^* ((c^*a + c^*bb^*a)x_0 + c^*bb^*c + c^*c) + \\
 &\quad a^*b + a^*c \rightarrow \\
 x_0 &= a^*b(c^*bb^*c)^* ((c^*a + c^*bb^*a)x_0 + a^*b(c^*bb^*c)^*c^*c + \\
 &\quad a^*(c^*bb^*c)^* (c^*a + c^*bb^*a)x_0 + a^*b(c^*bb^*c)^*c^*bb^*c + a^*b(c^*bb^*c)^*c^*c + \\
 &\quad a^*b + a^*c \rightarrow \\
 x_0 &= (a^*b(c^*bb^*c)^* ((c^*a + c^*bb^*a) + a^*(c^*bb^*c)^* (c^*a + c^*bb^*a)x_0 + \\
 &\quad a^*b(c^*bb^*c)^*c^*bb^*c + a^*b(c^*bb^*c)^*c^*c + a^*b(c^*bb^*c)^*c^*bb^*c + a^*b(c^*bb^*c)^*c^*c + \\
 &\quad a^*b + a^*c \rightarrow
 \end{aligned}$$

$$\begin{aligned}
 x_0 = a^*b(c^*bb^*c)^* (c^*a + c^*bb^*a) + a^*(c^*bb^*c)^* (c^*a + c^*bb^*a)( \\
 a^*b(c^*bb^*c)^*c^*bb^*c + a^*b(c^*bb^*c)^*c^*c + a^*b(c^*bb^*c)^*c^*bb^*c + a^*b(c^*bb^*c)^*c^*c +
 \end{aligned}$$

**Ejercicio 4.** (3 pts)

- Dada la siguiente gramática:

$S \rightarrow S | (S) | AB$   
 $A \rightarrow iC$   
 $B \rightarrow S | \lambda$   
 $C \rightarrow (S) | \lambda$

➤ Se pide:

- (0.25 pts). Comprobar si es LL(1) mediante el cálculo de los conjuntos Primero y Siguiente.
- (0.25 pts). Convertir la gramática del apartado anterior en un autómata con pila que acepte el mismo lenguaje por pila vacía.
- (0.5 pts). Analizar, teniendo en cuenta el principio de preanálisis (lectura de un símbolo de la entrada con anticipación) la entrada "i - - i ( i )" según el AP especificado en el apartado b anterior.
- (0.75 pts). Implementar la tabla de análisis sintáctico y especificar el pseudocódigo de análisis sintáctico tabular.
- (0.75 pts). Construir la traza correspondiente al reconocimiento de la frase "i - - i ( i )" según el pseudocódigo especificado en el apartado d anterior.
- (0.5 pts). Especificar el pseudocódigo de análisis sintáctico dirigido por la sintaxis para la gramática obtenida LL(1).

a) La gramática propuesta no tiene recursividad a izquierdo, por lo que es candidata para LL(1). Para que sea LL(1), la intersección de sus símbolos directores debe ser vacía ( $\emptyset$ ).

#### CÁLCULO CONSUNTOS PRIMERO Y SIGUIENTE

	PRIMERO	SIGUIENTE
S	{-, C, i }	{ ), \$ }
A	{ i }	{ -, ), \$ }
B	{ -, λ }	{ ), \$ }
C	{ (, λ }	{ -, ), \$ }

$$\text{PRIM}(-S) \cap \text{PRIM}((S)) \cap \text{PRIM}(A) = \emptyset$$

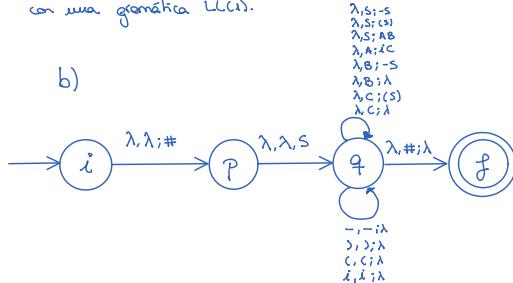
$$\text{PRIM}(-S) \cap \text{SIG}(B) = \emptyset$$

$$\text{PRIM}((S)) \cap \text{SIG}(C) = \emptyset$$

Como las 3 intersecciones son vacías, nos encontramos con una normativa LL(1).

Como las 3 intersecciones son vacías, nos encontramos con una gramática LL(1).

b)



c)

ESTADO	PILA	ENTRADA	Acción	INDETERMINACIÓN	Acción
$i$	$\lambda$	$i - i(i) \$$	$(i, \lambda, \lambda; \#; p)$		
$P$	$\#$	$i - i(i) \$$	$(p, \lambda, \lambda; S; q)$		
$q$	$S\#$	$i - i(i) \$$	$(q, \lambda, S; AB; q)$	$S \rightarrow AB$	
$q$	$AB\#$	$i - i(i) \$$	$(q, \lambda, AB; C; q)$	$A \rightarrow AC$	
$q$	$iCB\#$	$i - i(i) \$$	$(q, i, C; \lambda; q)$	RECONOCER('i');	
$q$	$CB\#$	$i - i(i) \$$	$(q, \lambda, C; \lambda; q)$	$C \rightarrow \lambda$	
$q$	$B\#$	$i - i(i) \$$	$(q, \lambda, B; \lambda; q)$	$B \rightarrow S$	
$q$	$-S\#$	$i - i(i) \$$	$(q, -i, -\lambda; q)$	RECONOCER('i');	
$q$	$S\#$	$i - i(i) \$$	$(q, \lambda, S; -S; q)$	$S \rightarrow S$	
$q$	$-S\#$	$i - i(i) \$$	$(q, -i, -\lambda; q)$	RECONOCER('i');	
$q$	$S\#\#$	$i - i(i) \$$	$(q, \lambda, S; AB; q)$	$S \rightarrow AB$	
$q$	$AB\#\#$	$i - i(i) \$$	$(q, \lambda, AB; C; q)$	$A \rightarrow AC$	
$q$	$iCB\#\#$	$i - i(i) \$$	$(q, i, C; \lambda; q)$	RECONOCER('i');	
$q$	$CB\#\#$	$i - i(i) \$$	$(q, i, C; S; q)$	$C \rightarrow CS$	
$q$	$(CS)\#\#$	$i - i(i) \$$	$(q, i, C; \lambda; q)$	RECONOCER('C');	
$q$	$CS\#\#$	$i - i(i) \$$	$(q, \lambda, S; (S); q)$	$S \rightarrow CS$	
$q$	$(S)\#\#$	$i - i(i) \$$	$(q, \lambda, S; \lambda; q)$	RECONOCER('i');	
$q$	$CS)\#\#$	$i - i(i) \$$	$(q, \lambda, S; AB; q)$	$S \rightarrow AB$	
$q$	$AB)\#\#$	$i - i(i) \$$	$(q, \lambda, AB; C; q)$	$A \rightarrow AC$	
$q$	$(CB)\#\#$	$i - i(i) \$$	$(q, \lambda, C; \lambda; q)$	RECONOCER('C');	
$q$	$(CB)\#\#$	$i - i(i) \$$	$(q, \lambda, C; \lambda; q)$	$C \rightarrow \lambda$	
$q$	$(B)\#\#$	$i - i(i) \$$	$(q, \lambda, B; \lambda; q)$	$B \rightarrow \lambda$	
$q$	$)\#\#$	$i - i(i) \$$	$(q, \lambda, ; \lambda; q)$	RECONOCER('i');	
$q$	$)\#\#$	$i - i(i) \$$	$(q, \lambda, ; \lambda; q)$	RECONOCER('i');	
$q$	$B\#$	$\$$	$(q, \lambda, B; \lambda; q)$	$B \rightarrow \lambda$	
$q$	$\#\#$	$\$$	$(q, \lambda, ; \lambda; q)$	error-sintáctico();	
$f$	$\lambda$	$\lambda$	ACEPTADA		

d)

```

procedure ConstruirTabla()
    ∀ A → α
        ∀ 'a' terminal != λ ∈ PRIM(α)
            Tabla[A][α] = α
        fin ∀
        si λ ∈ PRIM(α)
            ∀ 'b' terminal != λ ∈ SIG(α)
                Tabla[A][α] = λ
            fin ∀
    fin ∀
fin procedure
    
```

o) Tabla | Entrada | Salida

```

procedure AnálisisSintáctico()
    Apilar(H);
    Apilar(S);
    Leer(simbolo);
    Mientras NOT pile_vacia() hacer
        switch cima-pila() of
            case terminal:
                si cima-pila() == simbolo entonces
                    Desapilar(simbolo);
                    Leer(simbolo);
                sino
                    error-sintáctico();
            fin
        fin
    fin
    
```

f procedure

PILA	ENTRADA	ACCIÓN
$\lambda$	$\leftarrow-\lambda((x))\$$	Apilar( $\#$ );
$\#$	$\lambda-\lambda((x))\$$	Apilar( $S$ );
$S\#$	$\lambda-\lambda((x))\$$	$S ::= AB$
$AB\#$	$\lambda-\lambda((x))\$$	$A ::= \lambda C$
$AC\#$	$\lambda-\lambda((x))\$$	Reconocer( $\lambda$ );
$CB\#$	$--\lambda((x))\$$	$C ::= \lambda$
$B\#$	$--\lambda((x))\$$	$B ::= -S$
$-S\#$	$--\lambda((x))\$$	Reconocer( $-$ );
$S\$$	$--\lambda((x))\$$	$S ::= -S$
$-S\$$	$--\lambda((x))\$$	Reconocer( $-$ );
$S\$$	$i((x))\$$	$S ::= AB$
$AB\$$	$i((x))\$$	$A ::= \lambda C$
$CB\$$	$i((x))\$$	Reconocer( $\lambda$ );
$CB\$$	$((x))\$$	$C ::= \lambda$
$(SB\$$	$((x))\$$	Reconocer( $C$ );
$S)\#$	$((x))\$$	$S ::= (S)$
$(S)\#$	$((x))\$$	Reconocer( $($ );
$S)S\#$	$((x))\$$	$S ::= AB$
$AB)\#$	$((x))\$$	$A ::= \lambda C$
$(C)\#$	$((x))\$$	Reconocer( $\lambda$ );
$(C)\#$	$((x))\$$	$C ::= \lambda$
$(B)\#$	$((x))\$$	$B ::= -\lambda$
$(B)B\#$	$((x))\$$	Reconocer( $-$ );
$((B))\$$	$((x))\$$	Reconocer( $-$ );
$((B))\$$	$((x))\$$	$B ::= \lambda$
$\#$	$((x))\$$	ACCEPTADA
$\lambda$	$\lambda$	

... error-sintáctico();

fri

case no terminal:

si Tabla[cima-pila][simbolo] != error entonces  
Desapilar(cima-pila);  
Apilar(Tabla[cima-pila][simbolo]);

sino

error-sintáctico();

fri

fswitch

fmiéntres

si cima-pila == # entonces

Desapilar(#);  
Escribir(Cedera-ACEPTADA);

sino

error-sintáctico();

fri

fprocedure

f)

funcion Programa\_Principal  
SLA = leer-símbolo();  
 $S()$ ;  
si SLA != '\$' entonces  
error();  
fifuncion

funcion Reconocer(símbolo s)  
si (SLA == s) entonces  
leer-símbolo();  
sino  
error();  
fifuncion

funcion S()  
switch SLA of  
case '-' :  
    Reconocer('');  
    S();  
case '(' :  
    Reconocer('(');  
    S();  
    Reconocer('') ;  
case ')' :  
    A();  
    B();  
    default: error();  
fswitch  
ffuncion

funcion A()  
switch SLA of  
case 'i' :  
    Reconocer('i');  
    C();  
    default: error();  
fswitch  
ffuncion

funcion B()  
switch SLA of  
case '-' :  
    Reconocer('');  
    S();  
case '\$', ')' : /\*node\*/  
    fswitch  
ffuncion

funcion C()  
switch SLA of  
case '(' :  
    Reconocer('(');  
    S();  
    Reconocer('') ;  
case '-' : /\*node\*/  
    fswitch  
ffuncion