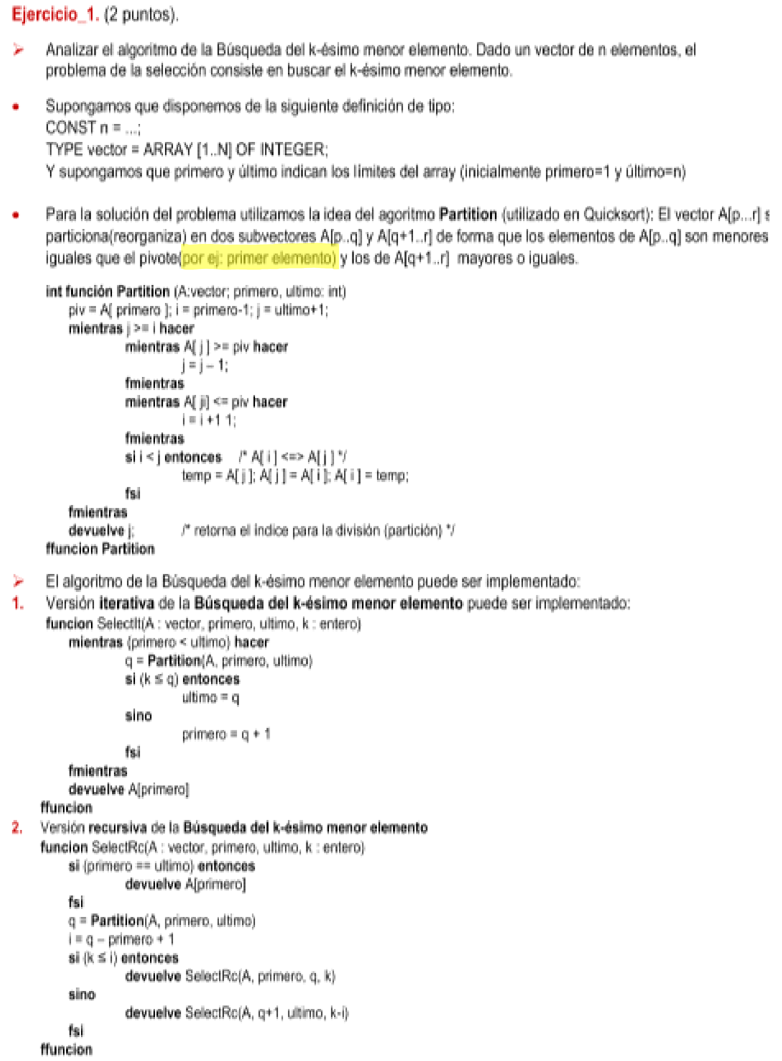
Febrero 2016

miércoles, 29 de diciembre de 2021 18:19



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 (inicialmente primero=1 y último=n)

- Para la solución del problema utilizamos la idea del agoritmo **Partition** (utilizado en Quicksort): El vector A[p...r] s particiona(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[ ji] <= piv hacer
                    i = i +1 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
```

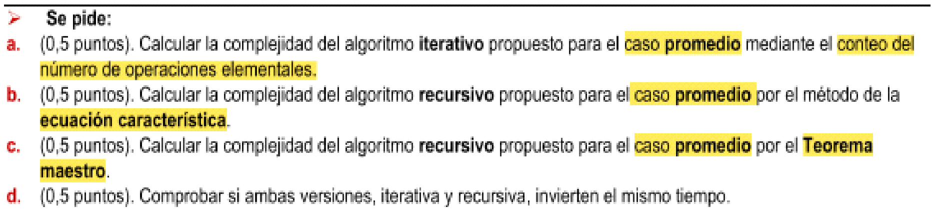
➢ El algoritmo de la Búsqueda del k-ésimo menor elemento puede ser implementado:

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

```
funcion 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**

```
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
```



➢ **Se pide:**

a. (0,5 puntos). Calcular la complejidad del algoritmo **iterativo** propuesto para el caso **promedio** mediante el conteo del 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) $T(n) = 1 + \sum_{i=1} (1 + 1 + 2 + 1 + T_{partition}) + 2$

$$\underline{T_{partition}} = 7 + \sum_{i=1}^{1} \Big(1 + 2 + \sum_{j=1}^{n/2} (5) + \sum_{j=1}^{n/2} (5) +$$

$$8 + 1 + 2 \Big) + 1 =$$

$$= 8 + 14 + \frac{5}{2} n + \frac{5}{2} n =$$

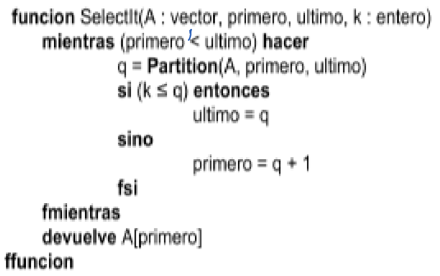
$$22 + n \cdot 5 \in O(n)$$

En cada iteración, el tamaño se divide a la mitad por tanto $n/2^i = 1 \to i = \log(n)$:

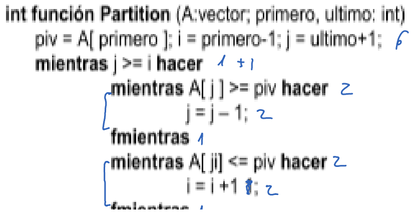
$$T(n) = 1 + \sum_{i=1}^{\log(n)} (1 + 1 + 2 + 1 + 22 + 5n) + 2 =$$

$$T(n) = 3 + \sum_{i=1}^{\log(n)} \left(27 + 5\left(\frac{n}{2^i}\right)\right) =$$

$$= 3 + 27 \log(n) + 5n \left(\cdot - \frac{1}{n} + 2^{*}\right) =$$

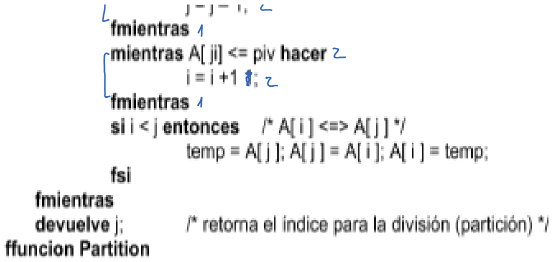


```
funcion SelectIt(A : vector, primero, ultimo, k : entero)
    mientras (primero ≤< ultimo) hacer
            q = Partition(A, primero, ultimo)
            si (k ≤ q) entonces
                    ultimo = q
            sino
                    primero = q + 1
            fsi
    fmientras
    devuelve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



```
int función Partition (A:vector; primero, ultimo: int)
    piv = A[ primero ]; i = primero-1; j = ultimo+1;   6
    mientras j >= i hacer   1 +1
          ┌ mientras A[ j ] >= piv hacer   2
          │         j = j – 1;   2
          └ fmientras   1
          ┌ mientras A[ ji] <= piv hacer   2
          │         i = i +1 ~~1~~;   2
          └ fmientras
```

$$= 3 + 27\log(n) + 5n\left(\cdot -\frac{1}{n}+2^{*}\right) =$$



```
fmientras 1
mientras A[ ji] <= piv hacer 2
        i = i +1 ; 2
fmientras 1
si i < j entonces    /* A[ i ] <=> A[ j ] */
        temp = A[ j ]; A[ j ] = A[ i ]; A[ i ] = temp;
fsi
fmientras
devuelve j;          /* retorna el indice para la división (partición) */
ffuncion Partition
```

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

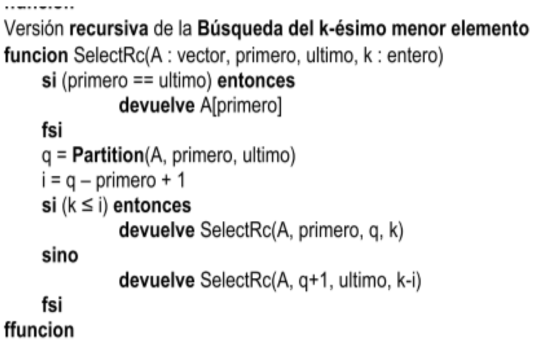
$$\left(\left(\frac{1}{2}\right)^{\log_2(n)+1} = \left(\frac{1}{2}\right)^{\log_2 n}\cdot\frac{1}{2} = \frac{1}{n}\cdot\frac{1}{2} = \frac{1}{2n}\right)$$

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

$$T(n) = 3 + 27\log(n) + 5n\left(-\frac{1}{n}+2\right) =$$

$$3 + 27\log(n) + -5 + 10n = 27\log(n) + 10n - 2 \in O(n)$$

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:

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

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

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

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

$$T(2^K) - T(2^{K-1}) = 5\cdot 2^K + 28 \rightarrow T(2^K) = t_K \rightarrow t_K - t_{K-1} = \underbrace{5\cdot 2^K}_{} \underbrace{+28}_{} \rightarrow$$

$$\rightarrow (x-1)(x-2)(x-1) = 0 \quad \rightarrow r_1 : 1 \text{ (doble)}$$

$$\underline{\underline{r_2 : 2}}$$

$$t_k = 1^k \cdot k^0 \cdot c_{11} + 1^k \cdot k^1 \cdot c_{12} + 2^k \cdot k^0 \cdot c_2 \Rightarrow$$

$$T(n) = c_{11} + \log(n) c_{12} + n c_2 \quad \in O(n)$$

$$T(1) = 3$$
$$T(2) = T(1) + 5 + 28 = 36$$
$$T(4) = T(2) + 10 + 28 = 74$$
$$T(8) = T(4) + 40 + 28 = 142$$

$$\begin{cases} c_{11} + \log_2(2) c_{12} + 2c_2 = 36 \\ c_{11} + \log_2(4) c_{12} + 4c_2 = 74 \\ c_{11} + \log_2(8) c_{12} + 8c_2 = 142 \end{cases} =$$

$$\begin{cases} c_{11} + c_{12} + 2c_2 = 36 \\ c_{11} + 2c_{12} + 4c_2 = 74 \\ c_{11} + 3c_{12} + 8c_2 = 142 \end{cases}$$

$$\begin{pmatrix} 1 & 1 & 2 & 36 \\ 1 & 2 & 4 & 74 \\ 1 & 3 & 8 & 142 \end{pmatrix} \xrightarrow[F_3 - F_1]{F_2 - F_1} \begin{pmatrix} 1 & 1 & 2 & 36 \\ 0 & 1 & 2 & 38 \\ 0 & 2 & 6 & 106 \end{pmatrix} \rightarrow$$

$$\xrightarrow{F_3 - 2F_2} \begin{pmatrix} 1 & 1 & 2 & 36 \\ 0 & 1 & 2 & 38 \\ 0 & 0 & 2 & 30 \end{pmatrix} \xrightarrow{F_3 \leftarrow \frac{1}{2} F_3} \begin{pmatrix} 1 & 1 & 2 & 36 \\ 0 & 1 & 2 & 38 \\ 0 & 0 & 1 & 15 \end{pmatrix}$$

$$\xrightarrow{F_1 - F_2} \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 2 & 38 \\ 0 & 0 & 1 & 15 \end{pmatrix} \xrightarrow{F_2 - 2F_3} \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 15 \end{pmatrix} \begin{cases} c_{11} = -2 \\ c_{12} = 8 \\ c_2 = 15 \end{cases}$$

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

c) El Teorema Maestro es:

$$T(n) \in \begin{cases} n^{\log_b a} & a > b^k \\ n^k \cdot \log^{p+1}(n) & a = b^k \\ n^k \cdot \log^p(n) & a < b^k \end{cases} \qquad \left. \begin{array}{l} a = 1 \\ b = 2 \\ k = 1 \\ p = 0 \end{array} \right\} \; T(n) \in O(n \cdot \log^0(n)) \rightarrow$$

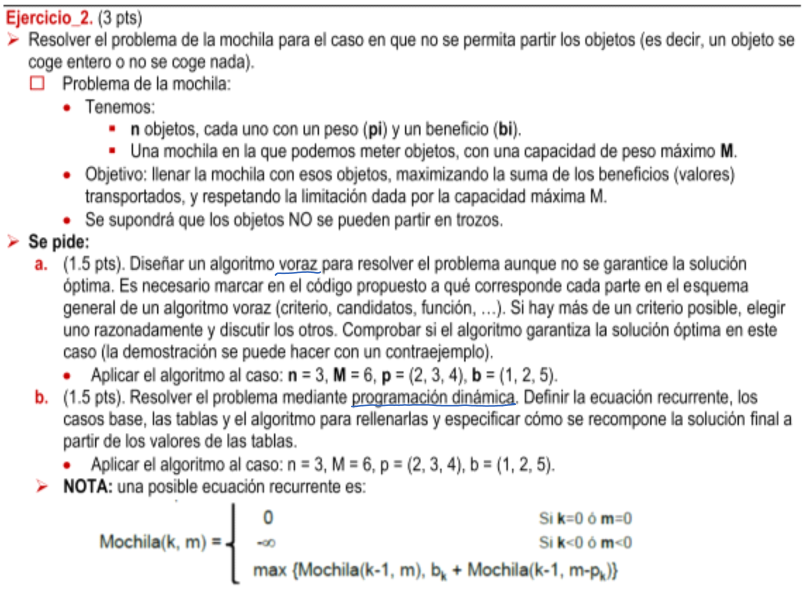
$$T(n) \in O(n);$$

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

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 permita 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 (**pi**) y un beneficio (**bi**).
      - 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:**
  - **a.** (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).
  - **b.** (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 recompone 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:

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

a)

```
funcion MochilaV (p, b: array [1..N] of Integer, M: Integer)
    X = {0, 0, 0};
    ordenarArticulos(); // Criterio: De menor a mayor cociente b/p
    peso ← 0;
    i ← 1 ;
        while( i ≤ N ) hacer
                si ( peso + p[i] ≤ M) entonces
                    peso ← peso + p[i]
                    X[i] = 1;
                fsi
                i ← i+1;
        fwhile
      devolver X;
    ffunción;
```

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:

$n=3$

siguiente ejemplo:

$n=3$
$M=6$
$p=(2,3,3)$
$b=(1,5,6)$
$b/p=(0.5, 1.\hat{6}, 2)$

Ordenamos según el criterio $\longrightarrow$ $p(2,4,2)$ $b(1,5,6)$

$X \rightarrow \{1,0,0\}$
$peso = 2$
$beneficio = 1$

$X \rightarrow \{1,1,0\}$
$peso = 6$
$beneficio = 6$

No sería la solución óptima. ($X=\{0,1,1\}$
$peso = 6$;
$beneficio = 11$;

Apicamos el algoritmo:

$M=6$, $n=3$

$b=(1,2,5)$

$p=(2,3,4)$

$b/p=(0.5, 0.6, 1.25)$

$X \rightarrow \{1,0,0\}$
$benef = 1$
$peso = 2$

$X \rightarrow \{1,1,0\}$
$benef = 3$
$peso = 5$

No genera la solución óptima
($X=\{1,0,1\}$
$peso = 6$
$benef = 6$)

b) La ecuación recurrente será:

$$\text{Mochila}(K, m) = \begin{cases} 0 & K=0,\ m=0 \\ -\infty & K<0,\ m<0 \\ \max(\text{Mochila}(K-1, m),\ \text{Mochila}(K-1, m-p_K)+b_K) \end{cases}$$

CASOS BASE

La tabla será:

T[N][M];

El algoritmo es el siguiente:

```
función MochilaPD(M: integer; b, p: array [1..N] of Integer)
    para i=1 hasta N hacer T[i][0] = 0; fpara;
    para j=0 hasta M hacer T[0][j] = 0; fpara;

    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];
```

```
            sino
                T[i][j] = max(T[i-1][j] , T[i-1][j-p[i]] +b[i] ;
            fsi
        fpara
    fpara
ffuncion
```

Aplicando el algoritmo al ejemplo nos queda:

$n=3$
$M=6$
$p=(2,3,4)$
$b=(1,2,5)$

T	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1
2	0	0	1	2	2	3	3
3	0	0	1	2	5	5	$\underline{\underline{6}}$

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**.

Actualizamos 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 $T[2][3] != T[1][3]$ → Cogemos el objeto $i=2$ y seguimos iterando:

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

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

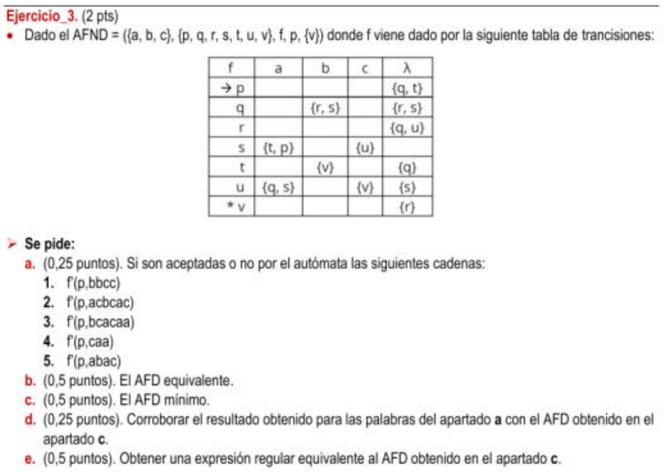
$T[1][0] == T[0][0]$ por lo que no lo cogemos.

El vector solución queda de la siguiente forma:

$X = \{0, 1, 1\}$

El algoritmo que describe como obtener la solución es:

```
funcion 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
ffuncion
```



Ejercicio_3. (2 pts)

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

f	a	b	c	λ
→ 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:**

a. (0,25 puntos). Si son aceptadas o no por el autómata las siguientes cadenas:

1. f'(p,bbcc)
2. f'(p,acbcac)
3. f'(p,bcacaa)
4. f'(p,caa)
5. f'(p,abac)

b. (0,5 puntos). El AFD equivalente.

c. (0,5 puntos). El AFD mínimo.

d. (0,25 puntos). Corroborar el resultado obtenido para las palabras del apartado **a** con el AFD obtenido en el apartado **c**.

e. (0,5 puntos). Obtener una expresión regular equivalente al AFD obtenido en el apartado **c**.

a)

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

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

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

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

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

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

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

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

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

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

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

$f(\{u,v,s,r,q\}, a) = \{t,p,q,s,r,u\}$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$v \in \{u, v, s, r, q\} \rightarrow$ <u>ACEPTADA</u>

b) Para construir el AFD equivalente usaremos una tabla:

	a	b	c
→ Q_0	Q_0	Q_1	Q_1
* Q_1	Q_0	Q_2	Q_1
Q_2	Q_0	Q_2	Q_1

$Q_0 = CL(p) = \{p, q, t, r, s, u\}$

$f'(Q_0, a) = \{t, p, q, s, r, u\} = Q_0$

$f'(Q_0, b) = \{r, s, v, q, u\} = Q_1^*$

$f'(Q_0, c) = \{u, v, s, r, q\} = Q_1$

$f'(Q_1, a) = \{t, p, q, s, r, u\} = Q_0$

$f'(Q_1, b) = \{r, s, q, u\} = Q_2$

$f(Q_1, c) = \{u, v, s, r, q\} = Q_1$

$f'(Q_2, a) = \{t, p, q, s, r, u\} = Q_0$

$f(Q_2, b) = \{r, s, q, u\} = Q_2$

$f(Q_2, c) = \{u, v, s, r, q\} = Q_1$

c) El AFD mínimo se obtiene mediante el algoritmo conjunto-cociente:

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

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

Como no coinciden sus transiciones → Creamos un nuevo conjunto.

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

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

	a	b	c
→ Q_0	Q_0	Q_1	Q_1
* Q_1	Q_0	Q_2	Q_1
Q_2	Q_0	Q_2	Q_1

d)

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

$f'(Q_0, b) = Q_1$

$f'(Q_1, b) = Q_2$

$f'(Q_2, c) = Q_1$

$f'(Q_1, c) = Q_1$

$Q_1 \in F \rightarrow$ <u>ACEPTADA</u>

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

$f'(Q_0, a) = Q_0$

$f'(Q_0, c) = Q_1$

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

$f'(Q_0, a) = Q_0$
$f'(Q_0, c) = Q_1$
$f'(Q_1, b) = Q_2$
$f'(Q_2, c) = Q_1$
$f'(Q_1, a) = Q_0$
$f'(Q_0, c) = Q_1 \in F \rightarrow$ <u>ACEPTADA</u>

3) $f'(Q_0, bcacaa) =$

$f'(Q_0, b) = Q_1$
$f'(Q_1, c) = Q_1$
$f'(Q_1, a) = Q_0$
$f'(Q_0, c) = Q_1$
$f'(Q_1, a) = Q_0$
$f'(Q_0, a) = Q_0 \notin F \rightarrow$ <u>RECHAZADA</u>

4) $f'(Q_0, caa)$

$f'(Q_0, c) = Q_1$
$f'(Q_1, a) = Q_0$
$f(Q_0, a) = Q_0 \notin F \rightarrow$ <u>RECHAZADA</u>

5) $f'(Q_0, abac)$

$f'(Q_0, a) = Q_0$
$f'(Q_0, b) = Q_1$
$f'(Q_1, a) = Q_0$
$f'(Q_0, c) = Q_1 \in F \rightarrow$ <u>ACEPTADA</u>

e)

	a	b	c
→ Q_0	Q_0	Q_1	Q_1
* Q_1	Q_0	Q_2	Q_1
Q_2	Q_0	Q_2	Q_1

$$\left\{\begin{array}{l} X_0 = aX_0 + bX_1 + cX_1 + b + c \\ X_1 = aX_0 + bX_2 + cX_1 + c \\ X_2 = aX_0 + bX_2 + cX_1 + c \end{array}\right. \rightarrow$$

$$\boxed{X = aX + \beta \leftrightarrow X = a^*\beta}$$

$$\rightarrow \left\{\begin{array}{l} X_0 = aX_0 + (bX_1 + cX_1 + b + c) \rightarrow X_0 = a^*(bX_1 + cX_1 + b + c) \\ X_1 = cX_1 + (aX_0 + bX_2 + c) \rightarrow X_1 = c^*(aX_0 + bX_2 + c) \\ X_2 = bX_2 + (aX_0 + cX_1 + c) \rightarrow X_2 = b^*(aX_0 + cX_1 + c) \end{array}\right.$$

$$\left\{\begin{array}{l} X_0 = a^*bX_1 + a^*X_1 + a^*b + a^*c \\ X_1 = c^*aX_0 + c^*bX_2 + c^*c \\ X_2 = b^*aX_0 + b^*cX_1 + b^*c \end{array}\right. \rightarrow$$

$$X_1 = c^*aX_0 + c^*b(b^*aX_0 + b^*cX_1 + b^*c) + c^*c \rightarrow$$

$$\rightarrow X_1 = c^*aX_0 + c^*bb^*aX_0 + c^*bb^*cX_1 + c^*bb^*c + c^*c \rightarrow$$

$$\rightarrow X_1 = c^*bb^*cX_1 + (c^*aX_0 + c^*bb^*aX_0 + c^*bb^*c + c^*c) \rightarrow$$

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

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

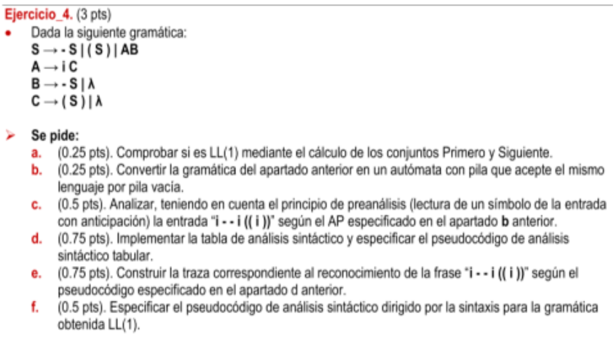
$$\begin{aligned}x_0 = {} & a^*b (c^*bb^*c)^* (c^*ax_0 + c^*bb^*ax_0 + c^*bb^*c + c^*c) + \\ & a^* (c^*bb^*c)^* (c^*ax_0 + c^*bb^*ax_0 + c^*bb^*c + c^*c) + \\ & a^*b + a^*c \rightarrow\end{aligned}$$

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

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

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

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



Ejercicio_4. (3 pts)

- Dada la siguiente gramática:
  S → - S | (S) | AB
  A → i C
  B → - S | λ
  C → (S) | λ

➤ **Se pide:**

- **a.** (0.25 pts). Comprobar si es LL(1) mediante el cálculo de los conjuntos Primero y Siguiente.
- **b.** (0.25 pts). Convertir la gramática del apartado anterior en un autómata con pila que acepte el mismo lenguaje por pila vacía.
- **c.** (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.
- **d.** (0.75 pts). Implementar la tabla de análisis sintáctico y especificar el pseudocódigo de análisis sintáctico tabular.
- **e.** (0.75 pts). Construir la traza correspondiente al reconocimiento de la frase "**i - - i ((i))**" según el pseudocódigo especificado en el apartado d anterior.
- **f.** (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 izquierdas, por lo que es candidata para LL(1). Para que sea LL(1), la intersección de sus símbolos directores debe ser vacía (∅).

CÁLCULO CONJUNTOS PRIMERO y SIGUIENTE

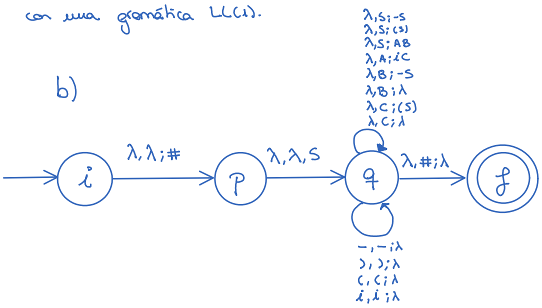
	PRIMERO	SIGUIENTE
S	{-, (, i }	{), $ }
A	{ i }	{-,), $ }
B	{-, λ }	{), $ }
C	{(, λ }	{-,), $ }

$$\mathrm{PRIM}(-S) \cap \mathrm{PRIM}((S)) \cap \mathrm{PRIM}(A) = \emptyset$$
$$\mathrm{PRIM}(-S) \cap \mathrm{SIG}(B) = \emptyset$$
$$\mathrm{PRIM}((S)) \cap \mathrm{SIG}(C) = \emptyset$$

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

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

b)



c)

ESTADO	PILA	ENTRADA	ACCIÓN	INDETERMINACIÓN	ACCIÓN
i	λ	i--i((i))$	(i,λ,λ;#,p)		
p	#	i--i((i))$	(p,λ,λ;S,q)		
q	S#	i--i((i))$	(q,λ,S;AB,q)		S→AB
q	AB#	i--i((i)) $	(q,λ,A;iC,q)		A→iC
q	iCB#	i--i((i)) $	(q,i,i;λ,q)		RECONOCER('i');
q	CB#	--i((i)) $	(q,λ,C;λ,q)	C→(S) C→λ	C→λ
q	B#	--i((i)) $	(q,λ,B;-S,q)		B→-S
q	-S#	--i((i)) $	(q,-,-;λ,q)		RECONOCER('-');
q	S#	-i((i)) $	(q,λ,S;-S,q)		S→-S
q	-S#	-i((i)) $	(q,-,-;λ,q)		RECONOCER(-');
q	S#	i((i)) $	(q,λ,S;AB,q)		S→AB
q	AB#	i((i)) $	(q,λ,A;iC,q)		A→iC
q	iCB#	i((i)) $	(q,i,i;λ,q)		RECONOCER(i);
q	CB#	((i)) $	(q,λ,C;(S),q)		C→(S)
q	(S)B#	((i))$	(q,(,(;λ,q)		RECONOCER('(');
q	S)B#	(i))$	(q,λ,S;(S),q)		S→(S)
q	(S))B#	(i))$	(q,(,(;λ,q)		RECONOCER('(');
q	S))B#	i))$	(q,λ,S;AB,q)		S→AB
q	AB))B#	i))$	(q,λ,A;iC,q)		A→iC
q	iCB))B#	i))$	(q,i,i;λ,q)		RECONOCER(i);
q	CB))B#	))$	(q,λ,C;λ,q)	(q,λ,C;λ,q)	C→λ
q	B))B#	))$	(q,λ,B;λ,q)	(q,λ,B;λ,q)	B→λ
q	))B#	))$	(q,),);λ,q)		RECONOCER(')');
q	)B#	)$	(q,),);λ,q)		RECONOCER(')');
q	B#	$	(q,λ,B;λ,q)	(q,λ,B;λ,q)	B→λ
q	#	$	(q,λ,#;λ,f)		
f	λ	λ	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][a] = λ
            fin ∀
        fsi
    f∀
fprocedure
```

```
procedure AnálisisSintáctico()
        Apilar(#);
        Apilar(S);

        Leer(simbolo);

    Mientras NOT pila_vacia() hacer
        switch cima_pila() OF
            case terminal:
                si cima_pila() == simbolo entonces
                    Desapilar(símbolo);
                    Leer(símbolo);
                sino
                    error_sintactico();
                fsi
```

e)

PILA	ENTRADA	ACCIÓN

```
fprocedure
```

e)

PILA	ENTRADA	ACCIÓN
λ	i--i((i))$	Apilar(#);
#	i--i((i))$	Apilar(S);
S#	i--i((i))$	S::=AB
AB#	i--i((i))$	A::= iC
iCB#	i--i((i))$	RECONOCER(i);
CB#	--i((i))$	C::=λ
B#	--i((i))$	B::=-S
-S#	--i((i))$	RECONOCER(-);
S#	-i((i))$	S::=-S
-S#	-i((i))$	RECONOCER('-');
S#	i((i))$	S::=AB
AB#	i((i))$	A::=iC
iCB#	i((i))$	RECONOCER('i');
CB#	((i))$	C::=(S)
(S)B#	((i))$	RECONOCER('(');
S)B#	(i))$	S::=(S)
(S))B#	(i))$	RECONOCER('(');
S))B#	i))$	S::=AB
AB))B#	i))$	A::=iC
iCB))B#	i))$	RECONOCER('i');
CB))B#	))$	C::=λ
B))B#	))$	B::=λ
))B#	))$	RECONOCER(')');
)B#	)$	RECONOCER(')');
B#	$	B::=λ
#	$	ACEPTADA
λ	λ	

```
                error_sintactico();
            fsi
        case no_terminal:
            si Tabla[cima_pila][simbolo] != error entonces
                Desapilar (cima_pila);
                Apilar(Tabla[cima_pila][simbolo]);
            sino
                error_sintactico ();
            fsi
        fswitch
    fmientras
    si cima_pila == # entonces
        Desapilar(#);
        Escribir (cadena_ACEPTADA);
    sino
        error_sintáctico();
    fsi
fprocedure
```

f)

```
funcion Programa_Principal
    SLA= leer-simbolo();
    S();
    si SLA != '$' entonces
        error();
    fsi
ffuncion
```

```
funcion Reconocer(simbolo s)
    si (SLA == S) entonces
        leer_simbolo();
    sino
        error();
    fsi
ffuncion
```

```
funcion S()
    switch SLA of
        case '-':
                Reconocer('-');
                S();
        case '(':
                Reconocer('(');
                S();
                Reconocer(')');
        case 'i':
                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 '$',')': /*nada*
    fswitch
ffuncion
```

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