# Enero-2018.pdf


CarlosGarSil98


Algorítmica y Modelos de Computación

3º Grado en Ingeniería Informática


Escuela Técnica Superior de Ingeniería
Universidad de Huelva




Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.
Available on the App Store
GET IT ON Google Play

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.
Available on the App Store
GET IT ON Google Play
9:41
WUOLAH
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
122
18 I
Ver mis op
Continúa d
405416_arts_esce ues2016juny.pdf
Top de tu g
7CR
Rocio
pony
Inicio
Asign



Ejercicio_1. (2 puntos)

Dado el esquema del algoritmo de ordenación QuickSort:

```
QuickSort (A, izq, der)  /* Ordena un vector A desde izq hasta der */
    if (izq < der) {
        piv=mediana (izq, der)
        div =partition (A, piv, izq, der)
        /* El vector A[izq..der] se particiona en dos subvectores A[izq..div] y A[div+1..der],
        de forma que los elementos de A[izq..div] son menores o iguales que los de A[div+1..der]
        (según elemento pivote) */
        QuickSort (A, izq, div)
        QuickSort (A, div+1, der)
        }
```

Donde, con "*mediana*" se obtiene la mediana de los elementos del array A entre las posiciones izq y der (el elemento que ocuparía la posición central si estuvieran ordenados), y "partition" es el procedimiento de particionar pero usando *piv* como pivote, con lo que el problema se divide en dos subproblemas de igual tamaño. Si el tiempo de ejecución del procedimiento "*mediana*" es $t_{med}(n)=20n$, y el de "partition" es $t_{par}(n)=n$:

a. (0,75 puntos). Calcular la complejidad del algoritmo propuesto por el método de la **ecuación característica**.

b. (0,25 puntos). Calcular la complejidad del algoritmo propuesto por el Teorema maestro.

c. (0,5 puntos). Calcular la complejidad del algoritmo propuesto por expansión de recurrencia.

d. (0,5 puntos). Si el método de la `Burbuja` tiene un tiempo de ejecución de $n^2$, justificar para qué valores de la entrada es preferible esta versión del `QuickSort` al método de la `Burbuja`.

NOTAS:

- Suma de los valores de la progresión geométrica $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$
- El Teorema maestro aplicado a **T(n) = aT(n/b) + Θ (n^k log^p n)** es:

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

**Apartado a:**

$$T(n) \begin{cases} 1 & \text{si } n=1 \\ 2T(n/2) + T_{med} + T_{part} + 8 & \text{si } n>1 \end{cases}$$

$T_{med}(n) = 20n$
$T_{part}(n) = n$

$$T(n) \begin{cases} 1 & \text{si } n=1 \\ 2T(n/2) + 21n + 8 & \text{si } n>1 \end{cases}$$

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.
WUOLAH

$T(n) - 2T(n/2) = 21n + 8$ $\left[\begin{array}{c}\text{cambio de base}\\ n = 2^k\end{array}\right]$ $T(2^k) - 2T(2^{k-1}) = 21 \cdot 2^k + 8$

No Homogénea

$T(2^k) - 2T(2^{k-1}) \longrightarrow (x-2)$
$b^k \cdot p(k)^d = 21 \cdot 2^k \longrightarrow (x-2)^{0+1}$
$b^k \cdot p(k)^d = 8 = 8 \cdot 1^k \longrightarrow (x-1)^{0+1}$
$p(x) = (x-2)^2(x-1)$ ; Raices: $r1 = 2$ doble, $r2 = 1$

$T(2^k) = C_0 \cdot 2^k \cdot k^0 + C_1 \cdot 2^k \cdot k^1 + C_2 \cdot 1^k \cdot k^0 = C_0 \cdot 2^k + C_1 \cdot 2^k \cdot k + C_2 \cdot 1^k$

$\left[\begin{array}{c}\text{cambio de base}\\ 2^k = n\end{array}\right]$ $k = \log(n)$ $T(n) = C_0 n + C_1 n \log(n) + C_2$

$T(1) = 1$ caso base
$T(2) = 2T(2/2) + 21 \cdot 2 + 8 = 51$
$T(4) = 2T(4/2) + 21 \cdot 4 + 8 = 194$
$T(8) = 2T(8/2) + 21 \cdot 8 + 8 = 564$

$$\begin{cases} C_0 \cdot 2 + C_1 \cdot 2\log(2) + C_2 = 51 \\ C_0 \cdot 4 + C_1 \cdot 4\log(4) + C_2 = 194 \\ C_0 \cdot 8 + C_1 \cdot 8\log(8) + C_2 = 564 \end{cases}$$

$C_0 = 17/2$, $C_1 = 21$, $C_2 = -8$

$T(n) = \frac{17}{2} n + 21 n \log(n) - 8 \in O(n \cdot \log(n))$

## Apartado b:

A partir del Teorema maestro: $aT(n/b) + O(n^k \cdot \log^p(n))$
En este caso: $a = 2$, $b = 2$, $k = 1$, $p = 0$

$a > b^k \longrightarrow 2 \quad 2^1$ ; No se cumple
$a = b^k \longrightarrow 2 \quad 2^1$ ; sí se cumple

$T(n) \in O(n \cdot \log(n))$

## Apartado c:

$T(n) = 2T(n/2) + 21n + 8$ ; $T(n) = 2(2T(n/4) + 21\frac{n}{2} + 8) + 21n + 8$

$T(n) = 4T(n/4) + 42n + 24$; $T(n) = 4(2T(n/8) + 21\frac{n}{4} + 8) + 42n + 24$

$T(n) = 8T(n/8) + 63n + 56 \longrightarrow T(n) = 2^i \cdot T(n/2^i) + 21 \cdot i\, n + 8 \cdot \sum_{j=1}^{\log(n)} (2^j)$

$\sum_{i=0}^{n} (2^i) = 2^{n+1} - 1 \longrightarrow \sum_{i=0}^{\log(n)} (2^i) = 2^{\log(n)+1} - 1$ ; $\sum_{i=1}^{\log(n)} (2^i) = 2^{\log(n)} - 1$

Detalle importante

$2^{\log(n)} = 1$; $T(n) = n + 21 \cdot n \cdot \log(n) + 8(n-1)$ ; $T(n) = 9n + 21 n \log(n) - 8 \in O(n \cdot \log(n))$

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Descarga la app de Wuolah desde tu store favorita

KEEP
CALM
AND
ESTUDIA
UN POQUITO

WUOLAH + #QuédateEnCasa

## Apartado d:

$T_{burbuja}(n) = n^2$ ; $T_{quicksort}(n) = 9n + 21 n \log(n) - 8$

Lo que tenemos que averiguar es:

$n^2 \geq 9n + 21 n \log(n) - 8 \longrightarrow n^2 - 9 \cdot n - 21 \cdot n \cdot \log(n) + 8 = 0$

$n = 64 \longrightarrow -4536$ ; A favor de Burbuja
$n = 128 \longrightarrow -3576$ ; A favor de Burbuja
$n = 256 \longrightarrow 20232$ ; A favor de Quicksort

valor entre 128 y 256

$n = 160 \longrightarrow -433'678$ ; A favor de Burbuja
$n = 170 \longrightarrow 936'474$ ; A favor de Quicksort

$n = 162 \longrightarrow -176'170$ ; A favor de Burbuja
$n = 163 \longrightarrow -44'697$ ; A favor de Burbuja
$n = 164 \longrightarrow 88'591$ ; A favor de Quicksort

Como no hay números enteros entre 163 y 164, paramos y podemos afirmar:
para $n \geq 164$ el algoritmo Quicksort es más eficiente

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Descarga la app de Wuolah desde tu store favorita

Ejercicio_2. (3 puntos)

- 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 (**$p_i$**) y un valor o 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 de capacidad máxima **M**.
    - Se supondrá que los objetos **NO** se pueden partir en trozos.
- **Se pide**:

a. (1.5 puntos). Diseñar un algoritmo **voraz** para resolver el problema aunque no se garantice la solución óptima. Es necesario marcar en el código propuesto a que 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 puntos). 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:

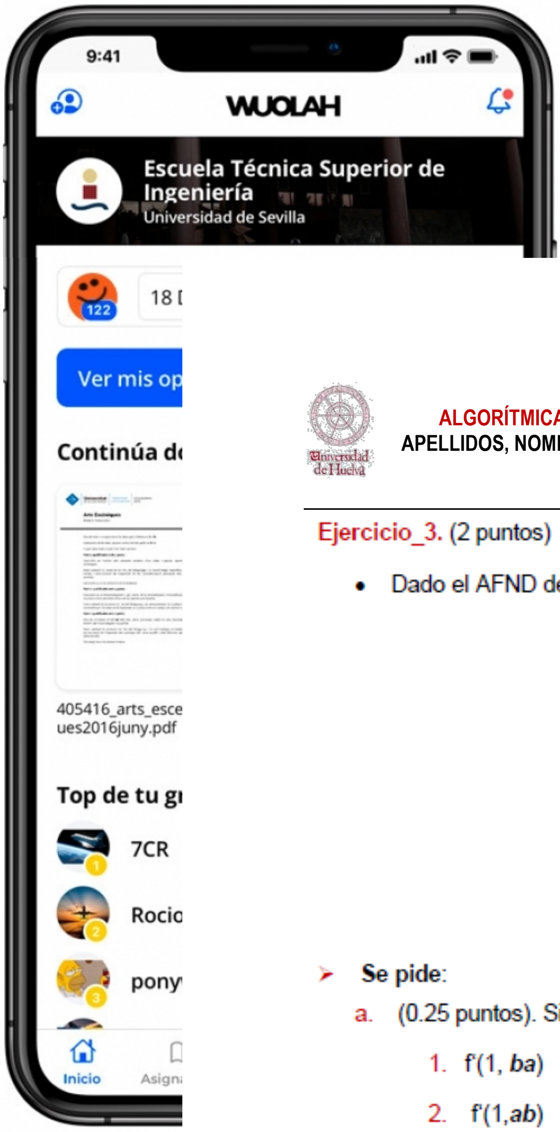
$$\text{Mochila}(k, m) = \begin{cases} 0 & \text{Si } k=0 \text{ ó } m=0 \\ -\infty & \text{Si } k<0 \text{ ó } m<0 \\ \max\{\text{Mochila}(k-1, m),\ b_k + \text{Mochila}(k-1, m-p_k)\} & \end{cases}$$



Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Descarga la app de Wuolah desde tu store favorita



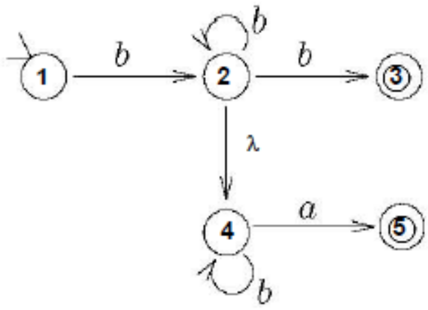
Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.
Available on the App Store
GET IT ON Google Play



APELLIDOS, NOMBRE García Silva, Carlos NOTA______

Ejercicio_3. (2 puntos)

- Dado el AFND definido en el grafo:




➢ **Se pide**:

a. (0.25 puntos). Si son aceptadas o no por el autómata las siguientes cadenas:

1. f'(1, *ba*)
2. f'(1,*ab*)
3. f'(1,*bb*)
4. f'(1,*b*)
5. f'(1,*bba*)

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.**

**Apartado a**

f'' (1, ba)
λ-clausura(1) = 1
f'({1}, b) = {2, 4}
f'({2, 4}, a) = 5
5 es estado final; ACEPTADA

f'' (1, ab)
λ-clausura(1) = 1
f'({1}, a) = ∅ conjunto vacío
NO ACEPTADA

f'' (1, bb)
λ-clausura(1) = 1
f'({1}, b) = {2, 4}
f'({2, 4}, b) = {2, 3, 4}
En el conjunto se encuentra el estado final 3; ACEPTADA

f'' (1, b)
λ-clausura(1) = 1
f'({1}, b) = {2, 4}
No hay estado final en el conjunto; NO ACEPTADA

WUOLAH
Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

$f''(1, bba)$
$\lambda$- clausura $(1) = 1$
$f'(\{1\}, b) = \{2, 4\}$
$f'(\{2,4\}, b) = \{2,3,4\}$
$f'(\{2,3,4\}, a) = 5$
5 es estado final; ACEPTADA

## Apartado b:

$\rightarrow Q_0 = \lambda$- Clausura $(1) = \{1\}$
$f'(Q_0, a) = \emptyset$ conjunto vacío
$f'(Q_0, b) = \{2, 4\}$ $Q_1$ estado normal

$Q_1 = \{2,4\}$
$f'(Q_1, a) = 5$ $Q_2$ estado final
$f'(Q_1, b) = \{2,3,4\}$ $Q_3$ estado final

$^*Q_2 = \{5\}$
$f'(Q_2, a) = \emptyset$
$f'(Q_2, b) = \emptyset$

$^*Q_3 = \{2,3,4\}$
$f'(Q_3, a) = 5 \longrightarrow Q_2$
$f'(Q_3, b) = 2,3,4 \longrightarrow Q_3$

| | a | b |
|---|---|---|
| $\rightarrow Q_0$ | | $Q_1$ |
| $Q_1$ | $Q_2$ | $Q_3$ |
| $* Q_2$ | | |
| $* Q_3$ | $Q_2$ | $Q_3$ |

## Apartado c:

Agrupamos entre estados finales y no finales
$Q/E_0 = ( C_0 = \{Q_0, Q_1\}, C_1 = \{Q_2, Q_3\}$

$f'(Q_0, a) = \emptyset$ $\quad$ $f'(Q_1, a) = C_1$ $\Big\}$ No coincide
$f'(Q_0, b) = C_0$ $\quad$ $f'(Q_1, b) = C_1$ $\quad$ hay que dividir

$f'(Q_2, a) = \emptyset$ $\quad$ $f'(Q_3, a) = C_1$ $\Big\}$ No coincide
$f'(Q_2, b) = \emptyset$ $\quad$ $f'(Q_3, b) = C_1$ $\quad$ hay que dividir

Como el resultado es un estado por cada conjunto, ya nos encontrábamos con el AFD mínimo en el anterior apartado.

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH
Descarga la app de Wuolah desde tu store favorita

### Apartado d:

$f''(Q_0, ba)$
$f'(Q_0, b) = Q_1$
$f'(Q_1, a) = Q_2$
$Q_2$ es estado final; ACEPTADA

$f''(Q_0, ab)$
$f'(Q_0, a) = \emptyset$
No es estado final;
NO ACEPTADA

$f''(Q_0, bb)$
$f'(Q_0, b) = Q_1$
$f'(Q_1, b) = Q_3$
$Q_3$ es estado final; ACEPTADA

$f''(Q_0, b)$
$f'(Q_0, b) = Q_1$
$Q_1$ no es estado final;
NO ACEPTADA

$f''(Q_0, bba)$
$f'(Q_0, b) = Q_1$
$f'(Q_1, b) = Q_3$
$f'(Q_3, a) = Q_2$
$Q_2$ es estado final; ACEPTADA

### Apartado e:

Ecuación característica
$$\begin{cases} X_0 = bX_1 \\ X_1 = aX_2 + bX_3 + a + b \\ X_2 = \lambda \\ X_3 = aX_2 + bX_3 + a + b \end{cases}$$

Se realizará mediante sustitución:
$X_2 = \lambda$

$X_3 = aX_2 + bX_3 + a + b; X_3 = a\lambda + bX_3 + a + b$
$X_3 = bX_3 + a + b; X_3 = b^*(a + b)$

$X_1 = aX_2 + bX_3 + a + b, X_1 = a\lambda + bX_3 + a + b;$
$X_1 = a\lambda + b(b^*(a+b)) + a + b; X_1 = bb^*a + bb^*b + a + b$

$X_0 = bX_1; X_0 = b(bb^*a + bb^*b + a + b);$
$X_0 = bbb^*a + bbb^*b + ba + bb$

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Descarga la app de Wuolah desde tu store favorita

**Ejercicio_4.** (3 puntos)

Considérese la siguiente gramática:

S → ( L )
 | a
L → L % S
 | S

a. (0,25 puntos). Comprobar si es LL(1) mediante el cálculo de los conjuntos Primero y Siguiente.

b. (0.25 puntos). Con la gramática equivalente LL(1), especificar un autómata con pila que acepte el mismo lenguaje por pila vacía.

c. (0.5 puntos). Analizar por el autómata del apartado b. anterior, teniendo en cuenta el principio de preanálisis (lectura de un símbolo de la entrada con anticipación) la entrada "(a%(a%a))".

d. (0,75 puntos) Con la gramática equivalente LL(1), construir la tabla de análisis LL(1) y especificar el pseudocódigo de análisis sintáctico tabular.

e. (0,75 puntos) Construir la traza correspondiente al reconocimiento de la frase: "(a%(a%a))" según el pseudocódigo especificado en el apartado d. anterior.

f. (0,5 puntos) Especificar el pseudocódigo de análisis sintáctico dirigido por la sintaxis para la gramática obtenida LL(1).

## Apartado a:

vamos a quitar la recursividad por la izquierda:

| | |
|---|---|
| L → L % S | L → S L' |
| \| S | L' → % S L' |
| | \| λ |

Gramática equivalente
1. S → ( L )
2. | a
3. L → S L'
4. L' → % S L'
5. | λ

| | Primeros | Siguientes | Predicción |
|---|---|---|---|
| S | ( | % λ $ | ( |
| | a | | a |
| L | ( a | ) | ( a |
| L' | % | ) | % |
| | λ | | ) |

intersección vacía (S)

intersección vacía (L')

Como todas las intersecciones son vacías, podemos decir que nos encontramos con la gramática equivalente LL(1).

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

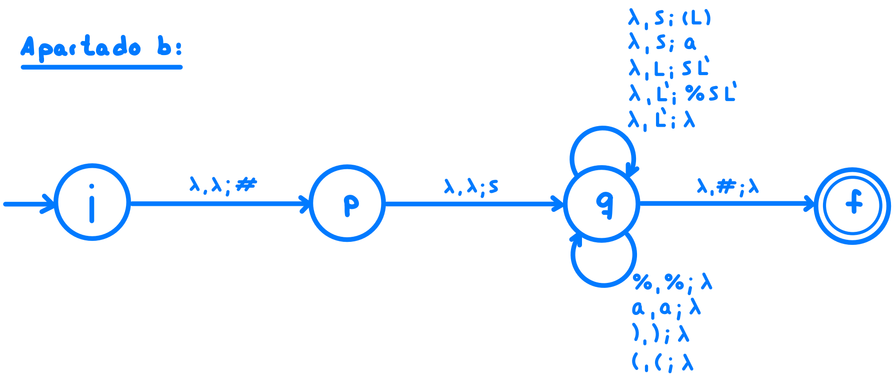


WUOLAH
Descarga la app de Wuolah desde tu store favorita

9:41
WUOLAH
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
122
18 [
Ver mis op
Continúa d
405416_arts_esce
ues2016juny.pdf
Top de tu g
7CR
Rocio
pony
Inicio
Asign

# Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.
Available on the App Store
GET IT ON Google Play

**Apartado b:**




**Apartado c:**

| Estado | Pila | Entrada | Acción | Indetermina | Acción |
|---|---|---|---|---|---|
| i | λ | (a%(a%a))$ | i λ λ;p # | | |
| p | # | (a%(a%a))$ | p λ λ;q S | | |
| q | S # | (a%(a%a))$ | q λ S;q (L) | | S::=(L) |
| q | ( L ) # | (a%(a%a))$ | q ( (;q λ | | Reconoce (() |
| q | L ) # | (a%(a%a))$ | q λ L;q SL' | | L::= SL' |
| q | S L' ) # | a%(a%a))$ | q λ S;q a | | S::=a |
| q | a L' ) # | a%(a%a))$ | q a a;q λ | | Reconoce (a) |
| q | L' ) # | %(a%a))$ | q λ L';q %SL' | | L'::= %SL' |
| q | % S L' ) # | %(a%a))$ | q % %;q λ | | Reconoce (%) |
| q | S L' ) # | (a%a))$ | q λ S;q (L) | | S::= (L) |
| q | ( L ) L' ) # | (a%a))$ | q ( (;q λ | | Reconoce (() |
| q | L ) L' ) # | a%a))$ | q λ L;q SL' | | L:: SL' |
| q | S L' ) L' ) # | a%a))$ | q λ S;q a | | S::=a |
| q | a L' ) L' ) # | a%a))$ | q a a;q λ | | Reconoce (a) |
| q | L' ) L' ) # | %a))$ | q λ L';q %SL' | | L' ::= %SL' |
| q | % S L' ) L' ) # | %a))$ | q % %;q λ | | Reconoce (%) |
| q | S L' ) L' ) # | a))$ | q λ S;q a | | S::=a |
| q | a L' ) L' ) # | a))$ | q a a;q λ | | Reconoce (a) |
| q | L' ) L' ) # | ))$ | q λ L';%SL' | q λ L';q λ | L'::= λ |
| q | ) L' ) # | ))$ | q ) );q ) | | Reconoce ()) |
| q | L' ) # | )$ | q λ L';%SL' | q λ L';q λ | L'::= λ |
| q | ) # | )$ | q ) );q λ | | Reconoce ()) |
| q | # | $ | q λ #;f λ | | |
| f | λ | $ | ACEPTAR | | ACEPTAR |

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Apartado d:

La tabla se obtiene mediante el siguiente algoritmo:

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

```
procedimiento   Analisis_tabular()
    Apilar(#);
    Apilar(S);        S = axioma
    Leer(simbolo);    preanalisis = simbolo
    mientras NOT pila_vacia hacer
        switch cima_pila
            case terminal:
                si cima_pila == simbolo entonces
                    Desapilar(simbolo);
                    Leer(simbolo);
                sino
                    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_sintactico();
    fsi
fprocedimiento
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Descarga la app de Wuolah desde tu store favorita

## Apartado e:

| Pila | Entrada | Acción |
|---|---|---|
| λ | (a%(a%a))$ | Apilar(#) |
| # | (a%(a%a))$ | Apilar(S) |
| S # | (a%(a%a))$ | S::=(L) |
| ( L ) # | (a%(a%a))$ | Leer(() |
| L ) # | a%(a%a))$ | L::=SL' |
| S L' ) # | a%(a%a))$ | S::=a |
| a L' ) # | a%(a%a))$ | Leer(a) |
| L' ) # | %(a%a))$ | L'::=%SL' |
| % S L' ) # | (a%a))$ | Leer(%) |
| S L' ) # | (a%a))$ | S::=(L) |
| ( L ) L' ) # | (a%a))$ | Leer(() |
| L ) L' ) # | a%a))$ | L::=SL' |
| S L' ) L' ) # | a%a))$ | S::=a |
| a L' ) L' ) # | a%a))$ | Leer(a) |
| L' ) L' ) # | %a))$ | L'::=%SL' |
| % S L' ) L' ) # | %a))$ | Leer(%) |
| S L' ) L' ) # | a))$ | S::=a |
| a L' ) L' ) # | a))$ | Leer(a) |
| L' ) L' ) # | ))$ | Desapilar(L') |
| ) L' ) # | ))$ | Leer()) |
| L' ) # | )$ | Desapilar(L') |
| ) # | )$ | Leer()) |
| # | $ | Desapilar(#) |
| λ | $ | Aceptar |

## Apartado f:

```
programa_Principal ()
    SLA = leer_simbolo();
    S();
    si SLA != $ entonces
        Error ();
    fsi
fprograma

procedimiento Reconocer(simbolo T)
    si SLA == T entonces
        leer_simbollo();
    sino
        error_sintactico();
    fsi
fprocedimiento
```

```
funcion S()
    switch SLA
        case (:
            Reconoce(();
            L();
            Reconoce());
        case a:
            Reconoce(a);

        default:
            error_sintactico();
    fswitch
ffuncion

funcion L()
    S();
    L'();
ffuncion
```

```
funcion L'()
    switch SLA
        case %:
            Reconoce(%);
            S();
            L'();
        case ):
        default:
            error_sintactico();
    fswitch
ffuncion
```

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Descarga la app de Wuolah desde tu store favorita