Febrero-2020.pdf

CarlosGarSil98

Algorítmica y Modelos de Computación

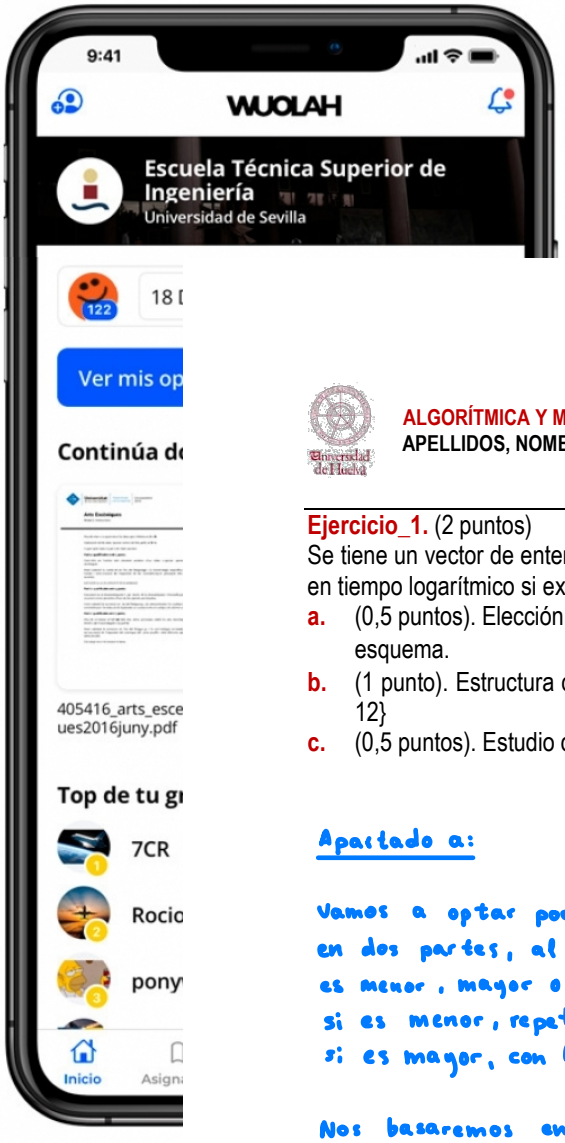
3º Grado en Ingeniería Informática

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

Universidad de Huelva. Escuela Técnica de Ingeniería. Departamento de Tecnologías de la Información.
ALGORÍTMICA Y MODELOS DE COMPUTACIÓN. 3º Grado Ingeniería Informática. **El Carmen 2 de septiembre del 2020.**
APELLIDOS, NOMBRE García Silva, Carlos **NOTA**

Ejercicio_1. (2 puntos)
Se tiene un vector de enteros no repetidos y ordenados de menor a mayor. Diseñar un algoritmo que compruebe en tiempo logarítmico si existe algún elemento del vector que coincida con su índice. Se pide:

a. (0,5 puntos). Elección del esquema (voraz, vuelta atrás, divide y vencerás). Identificación del problema con el esquema.

b. (1 punto). Estructura de datos. Algoritmo completo (con pseudocódigo). Aplicar a v = {-3, -2, 0, 2, 3, 5, 7, 9, 12}

c. (0,5 puntos). Estudio de coste.

Apartado a:

Vamos a optar por un esquema divide y vencerás. Dividiremos la lista en dos partes, al estar ordenada, podremos comprobar si la posición mitad es menor, mayor o igual. Si es igual habremos encontrado la solución, si es menor, repetiremos el proceso con la parte izquierda o por el contrario, si es mayor, con la parte derecha.

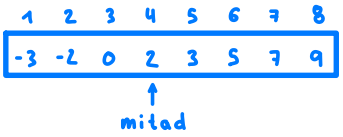
Nos basaremos en la búsqueda binaria recursiva, que en su caso peor tiene coste O(log(n))

Apartado b:

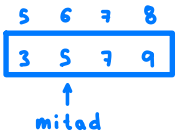
```
funcion IndiceRC (A: vector, p, u: entero)
  si p > u entonces
    devuelve 0
  fsi
  mitad ← (p+u)/2
  si mitad > A[mitad]
    devuelve IndiceRC (A, mitad+1, u)
  sino
    si mitad < A[mitad]
      devuelve IndiceRC (A, p, mitad-1)
    sino
      devuelve mitad
    fsi
  fsi
ffuncion
```

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

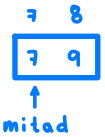
IndiceRC(A, 1, 8)
si 1 > 8 False
mitad = 9/2 = 4
si 4 > 2 True
devuelve IndiceRC(A, 4+1, 8)

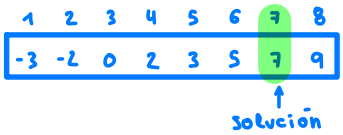


IndiceRC(A, 5, 8)
si 5 > 8 False
mitad = 13/2 = 6
si 6 > 5 True
devuelve IndiceRC(A, 6+1, 8)



IndiceRC(A, 7, 8)
si 7 > 8 False
mitad = 15/2 = 7
si 7 > 7 False
si 7 < 7 False
devuelve 7





Apartado C:

Para el caso base, indices cruzados, se comprueba la condición y se devuelve 0
Para el caso general, concretamente el peor, hacer el recorrido hasta la llamada donde se cruzan los indices, ya que el elemento siempre será menor que el que se encuentra en la mitad.

$$T(n) \begin{cases} 2 & \text{si } n = 0 \\ T(n/2) + 12 & \text{si } n > 0 \end{cases}$$

Usando el teorema maestro:

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

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

$a > b^k$; $1 > 2^0$ ⟶ No se cumple
$a = b^k$; $1 = 2^0$ ⟶ Sí se cumple

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

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

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 mochila: Una mochila en la que podemos meter objetos, con una capacidad de peso máximo **M**; tenemos **n** objetos, cada uno con un peso (**p**i) y un valor o beneficio (**b**i). El **Objetivo** es 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 punto). 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 punto). Resolver el problema mediante **programación dinámica**. Definir la ecuación recurrente, los casos base, las tablas. Aplicar el algoritmo al caso: **n**=3, **M**=6, **p**=(2,3,4), **b**=(1,2,5)

c. (1 punto). Resolver el problema por backtracking usando el esquema iterativo. Indicar cómo es la representación de la solución, la forma del árbol y programar las funciones genéricas del esquema correspondiente. Aplicar el algoritmo al caso: **n**=3, **M**=6, **p**=(2,3,4), **b**=(1,2,5)

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_3. (2 puntos)

Una caja fuerte dispone de una única cerradura. Para abrirla es necesario hacer girar en ella tres llaves diferentes (denominadas 1, 2 y 3), en un orden predeterminado, que se describe a continuación:

- Llave 1, seguida de llave 2, seguida de llave 3, o bien
- Llave 2, seguida de llave 1, seguida de llave 3.

Si no se respeta este orden, la caja se bloquea, y es imposible su apertura; por ejemplo, si se hace girar la llave 1, se retira la misma, se introduce de nuevo y se hace girar. Una vez abierta la caja fuerte, la introducción de las llaves en su cerradura, en cualquier orden, no afecta al mecanismo de cierre (permanece abierta).
Considérese que las denominaciones de las llaves son símbolos del alfabeto, sobre el que define el lenguaje L cuyas palabras son las secuencias permitidas de apertura de la caja fuerte. Por ejemplo, 12323 es una palabra del referido lenguaje.

Se pide:

a. (1 punto). Diseño del autómata AFND que acepta el lenguaje L.
b. (0,5 puntos). Gramática que genera las palabras de L.
c. (0,5 puntos). Autómata AFD mínimo equivalente del apartado **a**.

Apartado a:

Como observamos, hay 3 fases, desbloquear con la primera llave, la segunda y la tercera, por tanto hay 3 estados.

Hay un estado de bloqueo que lo trataremos como estado muerto

i: Estado inicial

pa: Desbloqueo de la primera parte con llave 1
pb: Desbloqueo de la primera parte con llave 2

sa: Desbloqueo de la segunda parte con llave 2
sb: Desbloqueo de la segunda parte con llave 1

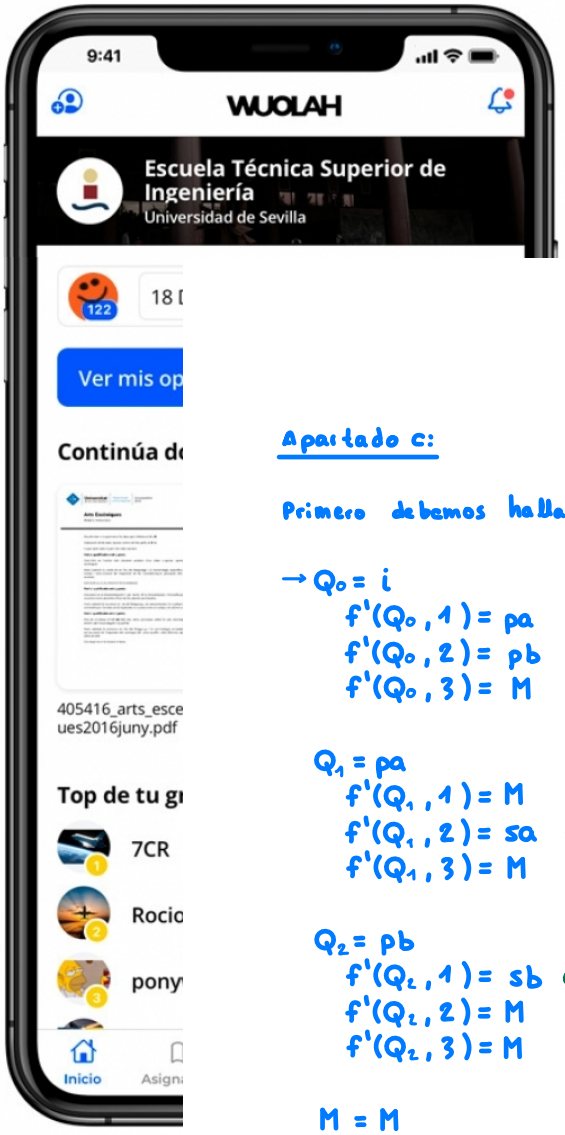
d: desbloqueo de la cerradura completa

M estado de bloqueo de la cerradura (estado de absorción)

	1	2	3	λ
→i	pa	pb	M	
pa	M	sa	M	
pb	sb	M	M	
sa	M	M	d	
sb	M	M	d	
* d	d	d	d	d
M	M	M	M	M

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

Apartado c:

Primero debemos hallar el autómata finito determinista equivalente:

→ $Q_0 = i$
- $f'(Q_0, 1) = pa$ Q_1 estado normal
- $f'(Q_0, 2) = pb$ Q_2 estado normal
- $f'(Q_0, 3) = M$ M estado de absorción

$Q_1 = pa$
- $f'(Q_1, 1) = M$
- $f'(Q_1, 2) = sa$ Q_3 estado normal
- $f'(Q_1, 3) = M$

$Q_2 = pb$
- $f'(Q_2, 1) = sb$ Q_4 estado normal
- $f'(Q_2, 2) = M$
- $f'(Q_2, 3) = M$

$M = M$
- $f'(M, 1) = M$
- $f'(M, 2) = M$
- $f'(M, 3) = M$

$Q_3 = sa$
- $f'(Q_3, 1) = M$
- $f'(Q_3, 2) = M$
- $f'(Q_3, 3) = d$ Q_5 estado final

$Q_4 = sb$
- $f'(Q_4, 1) = M$
- $f'(Q_4, 2) = M$
- $f'(Q_4, 3) = d$ Q_5

*$Q_5 = d$
- $f'(Q_5, 1) = d$ Q_5
- $f'(Q_5, 2) = d$ Q_5
- $f'(Q_5, 3) = d$ Q_5

	1	2	3
→Q_0	Q_1	Q_2	M
Q_1	M	Q_3	M
Q_2	Q_4	M	M
Q_3	M	M	Q_5
Q_4	M	M	Q_5
*Q_5	Q_5	Q_5	Q_5
M	M	M	M

Para generar el autómata mínimo, agrupamos entre estados no finales y finales:

$Q/E_0 = (C_0 = (Q_0, Q_1, Q_2, Q_3, Q_4), C_1 = (Q_5), C_2 = (M))$

$f'(Q_0, 1) = C_0$	$f'(Q_0, 2) = C_0$	$f'(Q_0, 3) = C_2$
$f'(Q_1, 1) = C_2$	$f'(Q_1, 2) = C_0$	$f'(Q_1, 3) = C_2$
$f'(Q_2, 1) = C_0$	$f'(Q_2, 2) = C_2$	$f'(Q_2, 3) = C_2$
$f'(Q_3, 1) = C_2$	$f'(Q_3, 2) = C_2$	$f'(Q_3, 3) = C_1$
$f'(Q_4, 1) = C_2$	$f'(Q_4, 2) = C_2$	$f'(Q_4, 3) = C_1$

No coinciden hay que dividir, mantenemos los que coinciden

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

WUOLAH

$Q/E_1 = (C_0 = (Q_0, Q_1, Q_2), C_1 = (Q_3), C_2 = (M), C_3 = (Q_3, Q_4))$

$f'(Q_0,1) = C_0 \quad f'(Q_0,2) = C_0 \quad f'(Q_0,3) = C_2$
$f'(Q_1,1) = C_2 \quad f'(Q_1,2) = C_3 \quad f'(Q_1,3) = C_2$
$f'(Q_2,1) = C_3 \quad f'(Q_2,2) = C_2 \quad f'(Q_2,3) = C_2$
} No coinciden, hay que dividir

$f'(Q_3,1) = C_2 \quad f'(Q_3,2) = C_2 \quad f'(Q_3,3) = C_1$
$f'(Q_4,1) = C_2 \quad f'(Q_4,2) = C_2 \quad f'(Q_4,3) = C_1$
} coinciden, se mantienen juntos

$Q/E_2 = (C_0 = (Q_0, Q_1), C_1 = (Q_3), C_2 = (M), C_3 = (Q_3, Q_4), C_4 = (Q_2))$

$f'(Q_0,1) = C_0 \quad f'(Q_0,2) = C_4 \quad f'(Q_0,3) = C_2$
$f'(Q_1,1) = C_2 \quad f'(Q_1,2) = C_3 \quad f'(Q_1,3) = C_2$
} No coinciden, hay que dividir

$Q/E_3 = (C_0 = (Q_0), C_1 = (Q_3), C_2 = (M), C_3 = (Q_3, Q_4), C_4 = (Q_2), C_5 = (Q_1))$

Apartado b:

Primero definiremos el autómata de forma:
$M = (Q, \Sigma, f, q_0, F)$
$M = (\{C_0, C_1, M, C_3, C_4, C_5\}, \{1,2,3\}, f, C_0, C_1)$

La gramática regular equivalente se define como:
$G = (Q, \Sigma, P, q_0)$
$G = (\{C_0, C_1, M, C_3, C_4, C_5\}, \{1,2,3\}, P, C_0)$

$$P = \begin{cases} C_0 ::= 1C_5 + 2C_4 \\ C_1 ::= 1C_1 + 2C_1 + 3C_1 + 1 + 2 + 3 \\ C_3 ::= 3C + 3 \\ C_4 ::= 1C_3 \\ C_5 ::= 2C_3 \end{cases}$$

	1	2	3
→ C_0	C_5	C_4	M
* C_1	C_1	C_1	C_1
C_3	M	M	C_1
C_4	C_3	M	M
C_5	M	C_3	M
M	M	M	M

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 → S = A | A
A → A + B | A – B | B
B → (S) | a | b

a. (0,5 puntos). Eliminar la recursividad por la izquierda y comprobar si es LL(1).
b. (0,5 puntos). Con la gramática equivalente LL(1), especificar un autómata con pila que acepte el mismo lenguaje por pila vacía. Analizar por el autómata anterior, teniendo en cuenta el principio de preanálisis (lectura de un símbolo de la entrada con anticipación) la entrada "**a=b**".
c. (1 punto). Con la gramática equivalente LL(1), construir la tabla de análisis LL(1) y especificar el pseudocódigo de análisis sintáctico tabular. Construir la traza correspondiente al reconocimiento de la frase: "**a-b**" según el pseudocódigo especificado.
d. (1 punto). Especificar el pseudocódigo de análisis sintáctico dirigido por la sintaxis para la gramática obtenida LL(1) y realizar el análisis para la entrada "**a=b**".

Apartado a:

Quitamos recursividad izquierda:

$S \to S=A \mid A$ | $S \to AS'$, $S' \to =AS' \mid \lambda$

$A \to A+B \mid A-B \mid B$ | $A \to BA'$, $A' \to +BA' \mid -BA' \mid \lambda$

Gramática equivalente:

1. $S \to AS'$
2. $S' \to =AS'$
3. $\mid \lambda$
4. $A \to BA'$
5. $A' \to +BA'$
6. $\mid -BA'$
7. $\mid \lambda$
8. $B \to (S)$
9. $\mid a$
10. $\mid b$

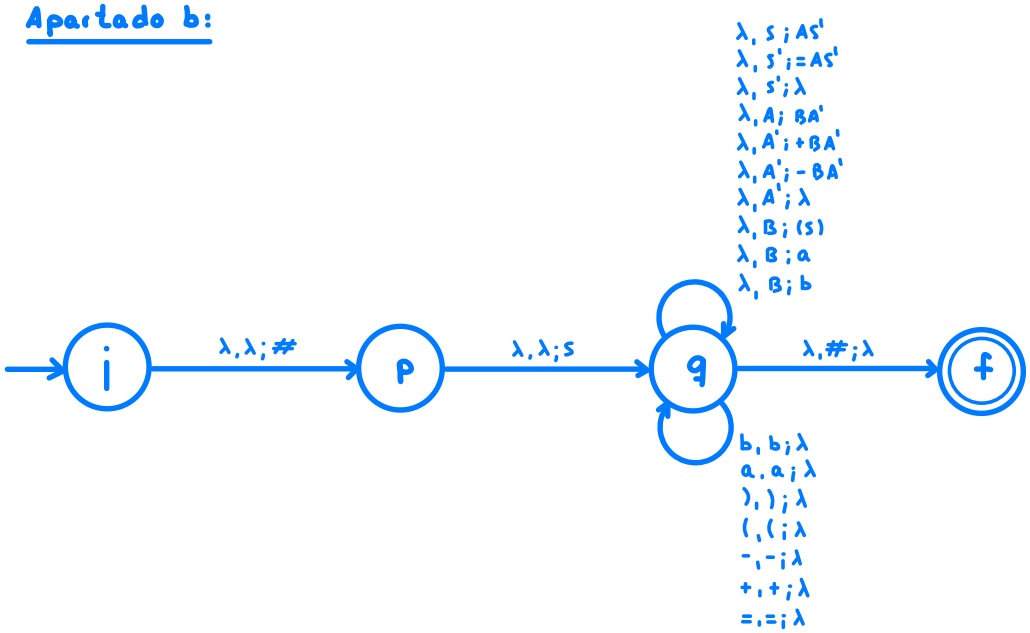
	Primeros	Siguientes	Predicción	
S	(a b	) $	(a b	
S'	=	) $	=	intersección vacía
	λ		) $	
A	(a b	= λ	(a b	
A'	+	= λ	+	intersección vacía
	-		-	
	λ		= λ	
B	(	+ - λ	(	intersección vacía
	a		a	
	b		b	

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

Descarga la app de Wuolah desde tu store favorita

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

Apartado b:

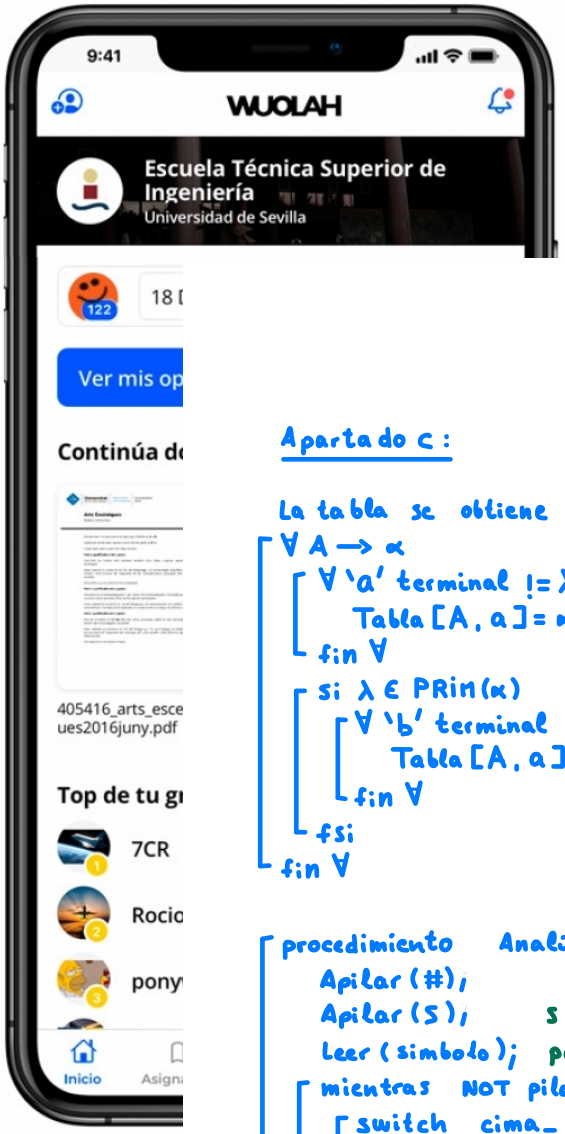


Estado	Pila	Entrada	Acción	Indetermina	Acción
i	λ	a = b $	i λ λ ; P #		
p	#	a = b $	p λ λ ; q S		
q	S #	a = b $	q λ S ; q AS'		S ::= AS'
q	A S' #	a = b $	q λ A ; q BA'		A ::= BA'
q	B A' S' #	a = b $	q λ B ; q a		B ::= a
q	a A' S' #	a = b $	q a a ; q λ		Reconoce (a)
q	A' S' #	= b $	q λ A' ; q + BA'	q λ A' ; q λ	A' ::= λ
q	S' #	= b $	q λ S' ; q = AS'		S' ::= = AS'
q	= A S' #	= b $	q = = ; q λ		Reconoce (=)
q	A S' #	b $	q λ A ; q BA'		A ::= BA'
q	B A' S' #	b $	q λ B ; q b		B ::= b
q	b A' S' #	b $	q b b ; q λ		Reconoce (b)
q	A' S' #	$	q λ A' ; q + BA'	q λ A' ; q λ	A' ::= λ
q	S' #	$	q λ S' ; q = AR'	q λ S' ; q λ	S' ::= λ
q	#	$	q λ # ; f λ		
f	λ	λ	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

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

Apartado c:

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
```

Pila	Entrada	Acción
λ	a - b $	Apilar(#)
#	a - b $	Apilar(S)
S #	a - b $	S::= AS'
A S' #	a - b $	A::= BA'
B A' S' #	a - b $	B::= a
a A' S' #	a - b $	Leer(a)
A' S' #	- b $	A'::= -BA'
- B A' S' #	- b $	Leer(-)
B A' S' #	b $	B::= b
b A' S' #	b $	Leer(b)
A' S' #	$	Desapilar(λ')
S' #	$	Desapilar(S')
#	$	Desapilar(#)
λ	$	Aceptar

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

Apartado d:

```
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()
    A();
    s'();
ffuncion
```

```
funcion s'()
    switch SLA
        case =:
            Reconocc(=);
            A();
            s'();
        case ), $:
        default:
            error_sintactico();
    fswitch
ffuncion
```

```
funcion A()
    B();
    A'();
ffuncion
```

```
funcion A'()
    switch SLA
        case +:
            Reconoca(+);
            B();
            A'();
        case -:
            Reconoca(-);
            B();
            A'();
        case =:
        default:
            error_sintactico();
    fswitch
ffuncion
```

```
funcion B()
    switch SLA
        case (:
            Reconocc(();
            S();
            Reconocc());
        case a:
            Reconocc( );
        case b:
            Reconocc(b);
        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.

WUOLAH

Descarga la app de Wuolah desde tu store favorita