Febrero-2017.pdf

CarlosGarSil98

Algorítmica y Modelos de Computación

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería
Universidad de Huelva

APELLIDOS, NOMBRE García Silva, Carlos **NOTA**

Ejercicio_1. (2 puntos).

Dado el esquema del algoritmo de ordenación QuickSort:

```
procedimiento QuickSort(A, izq, der) // Ordena un vector A desde izq hasta der
        si izq < der
                piv = mediana(A, izq, der)
                div = partition(A, piv, izq, der)
                QuickSort(A, izq, div)
                QuickSort(A, div+1, der)
        fsi
fprocedimiento
```

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 tmed(n) = 20n**, y el de "**partition**" es **tpar(n) = n**:

a. (0.75 pts). Calcular la complejidad del algoritmo propuesto por el método de la ecuación Característica.
b. (0.75 pts). Calcular la complejidad del algoritmo propuesto por expansión de recurrencia.
c. (0.5 pts). Si el método de la Burbuja tiene un tiempo de ejecución de n2, justificar para qué valores de la entrada es preferible esta versión del QuickSort al método de la Burbuja.

NOTA: Suma de los valores de la progresión geométrica es $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

Apartado a:

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

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

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

No Homogénea

$T(n) - 2T(n/2) = 21n + 8$ [cambio de base $n = 2^k$] $T(2^k) - 2T(2^{k-1}) = 21 \cdot 2^k + 8$

$T(2^k) - 2T(2^{k-1}) \longrightarrow (x-2)$
$b^k \cdot p(k)^d = 21 \cdot 2^k \longrightarrow (x-2)^{0+1}$
$b^k \cdot p(k)^d = 8 = 8 \cdot 1^k \longrightarrow (x-1)^{0+1}$
$p(x) = (x-2)^2(x-1)$; Raices: $r_1 = 2$ doble, $r_2 = 1$

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

[cambio de base $2^k = n$] $k = \log(n)$ $T(n) = C_0 n + C_1 n \log(n) + C_2$

$T(1) = 1$ 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:

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

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

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

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

Detalle importante

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

Apartado c:

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

Lo que tenemos que averiguar es:
$n^2 \geq 9n + 21n\log(n) - 8 \longrightarrow n^2 - 9 \cdot n - 21 \cdot n \cdot \log(n) + 8 = \Theta$

$n = 64 \longrightarrow -4536$; 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

masaltOS.com

Sube al siguiente nivel

¡7CM MÁS ALTO
SIN QUE NADIE SEPA CÓMO!

Ejercicio_2. (3 pts)

La agencia matrimonial Celestina & Co. Quiere informatizar parte de la asignación de parejas entre sus clientes. Cuando un cliente llega a la agencia se describe a sí mismo y cómo le gustaría que fuera su pareja. Con la información de los clientes, la agencia construye dos matrices, M y H, que contienen las preferencias de los unos por los otros, tales que la fila M[i, -] es una ordenación de mayor a menor de las mujeres cliente según las preferencias del i-ésimo hombre, y la fila H[i, -] es una ordenación de mayor a menor de los hombres según las preferencias de la i-ésima mujer. Por ejemplo, M[i, 1] almacenaría a la mujer preferida por i y M[i, 2] a su segunda preferida. Dado el alto índice de divorcios, la empresa se ha planteado como objetivo que los emparejamientos sean lo más estables posibles, evitando la siguiente situación:

1. Que dada una pareja (h',m') se dé el caso de m' prefiera aun h sobre h' y además h' prefiera a un m sobre m'.
2. Que dada una pareja(h'',m'') se dé el caso de h'' prefiera a un m sobre m'' y además m prefiera a un h sobre h''.

➢ La agencia quiere que dadas las matrices de preferencia un programa establezca parejas evitando la situación expuesta anteriormente.

Ejemplo: Sean María, Ana y Pepa el conjunto de mujeres, y Carlos, Nacho y Juan el conjunto de hombres, y sean las matrices de preferencia las que se muestran a continuación:

María	Carlos	Nacho	Juan
Ana	Juan	Nacho	Carlos
Pepa	Juan	Carlos	Nacho

Matriz de preferencias de las mujeres

Carlos	Ana	María	Pepa
Nacho	María	Ana	Pepa
Juan	María	Pepa	Ana

Matriz de preferencias de los hombres

Un emparejamiento estable sería: E = { (María, Carlos), (Ana, Nacho), (Pepa, Juan) } y uno inestable sería I = { (María, Juan), (Ana, Nacho), (Pepa, Carlos) }

a. (1 puntos). Resolver el problema por backtracking, el esquema general y explicación de su aplicación al problema. Aplicar al ejemplo.

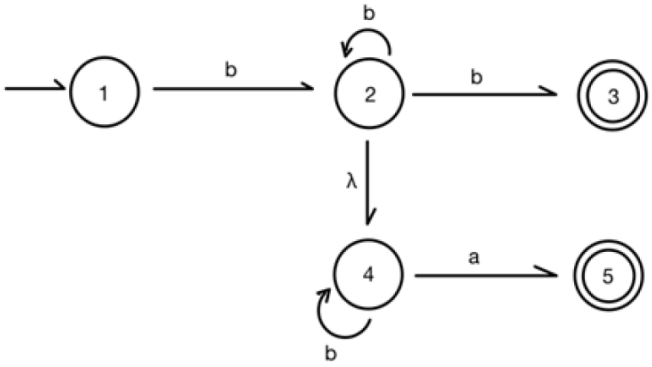
b. (2 puntos). Resolver el problema por el algoritmo de Gale & Shapley. Explicar de qué estrategia algorítmica se trata. El esquema general y explicación de su aplicación al problema. Aplicar al ejemplo.

NOTA: El algoritmo de Gale & Shapley establece que si el número de hombres es el mismo que el de mujeres, siempre existe una solución con matrimonios estables. La descripción del algoritmo es la siguiente:

- En un comienzo todos están sin parejas.
- Suponer que la mujer es la que propone matrimonio (nos fijamos en la lista de las mujeres).
- Mientras haya una mujer libre, hace lo siguiente:
  - Le propone matrimonio al primer hombre de su lista y ocurre que:
    - Si el hombre está libre, se casan.
    - Si el hombre ya está emparejado, le pregunta si la prefiere a ella antes que a su pareja actual (mirando la lista ordenada de preferencias de él):
      - Si la prefiere a ella, el hombre se divorcia.
      - Si no, entonces la mujer sigue y le propondrá matrimonio al siguiente de su lista.

Al final, el resultado es un conjunto de parejas estables, en el sentido que hemos definido antes. Esta solución es óptima para el conjunto de las mujeres y la peor para el conjunto de los hombres. Si se empieza con el conjunto de hombres, se obtiene una solución óptima para ellos.

Ejercicio_3. (2 pts)
Dado el AFND definido en el grafo, hallar:



a. (0.5 pts). Comprobar si son aceptadas o no por el autómata las siguientes cadenas: **ba**, **ab**, **bb**,**b**, **bba**
b. (0.5 pts). El AFD mínimo equivalente.
c. (0.5 pts). Corroborar el resultado obtenido para las palabras del apartado a con el AFD obtenido en el apartado **b**.
d. (0.5 pts). La expresión regular del lenguaje reconocido por el autómata del apartado anterior.

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

f''(1, bba)
λ-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:

→Q_0 = λ-Clausura(1) = {1}
f'(Q_0, a) = ∅ 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) = ∅
f'(Q_2, b) = ∅

*Q_3 = {2,3,4}
f'(Q_3, a) = 5 ⟶ Q_2
f'(Q_3, b) = 2,3,4 ⟶ Q_3

	a	b
→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) = ∅ f'(Q_1, a) = C_1 } No coincide
f'(Q_0, b) = C_0 f'(Q_1, b) = C_1 } hay que dividir

f'(Q_2, a) = ∅ f'(Q_3, a) = C_1 } No coincide
f'(Q_2, b) = ∅ f'(Q_3, b) = C_1 } hay que dividir

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

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) = ∅
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

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

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:

$X_2 = \lambda$

$X_3 = aX_2 + bX_3 + a + b$; $X_3 = a\lambda + bX_3 + a + b$

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

$X_1 = aX_2 + bX_3 + a + b$, $X_1 = a\lambda + bX_3 + a + b$;

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

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

$X_0 = bbb^*a + bbb^*b + ba + bb$

Ejercicio_4. (3 puntos).

- Dada la siguiente gramática:

  S → a S | (S) | AB
  A → n C
  B → a 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. (0.75 pts). Implementar la tabla de análisis sintáctico y especificar el pseudocódigo de análisis sintáctico tabular.
c. (1 pt). Construir la traza correspondiente al reconocimiento de la frase "baab((b))" según el pseudocódigo especificado en el apartado b anterior.
d. (0.75 pts). Especificar el pseudocódigo de análisis sintáctico dirigido por la sintaxis para la gramática obtenida LL(1).

Apartado a:

	Primeros	Siguientes	Predicción	
S	a	) $	a	intersección vacía
	(		(	
	n		n	
A	n	a λ	n	
B	a	) $	a	intersección vacía
	λ		) $	
C	(	a λ	(	intersección vacía
	λ		a λ	

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

Apartado b:

La tabla se obtiene mediante el siguiente algoritmo:

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

Apartado c:

Pila	Entrada	Acción
λ	b a a b ((b))	Apilar(#)
#	b a a b ((b))	Apilar(S)
S #	b a a b ((b))	error_sintactico()

Apartado d:

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

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

```
funcion S()
    switch SLA
        case a:
            Reconoce(a);
            S();
        case (:
            Reconoce(();
            S();
            Reconoce());
        case n:
            A();
            B();
        default:
            error_sintactico();
    fswitch
ffuncion
```

```
funcion A()
    switch SLA
        case n:
            Reconoce(n);
            C();
        default:
            error_sintactico();
    fswitch
ffuncion
```

```
funcion B()
    switch SLA
        case a:
            Reconoce(a);
            S();
        case ),$:
        default:
            error_sintactico();
    fswitch
ffuncion
```

```
funcion C()
    switch SLA
        case (:
            Reconoce(();
            S();
            Reconoce());
        case a:
        default:
            error_sintactico();
    fswitch
ffuncion
```