Febrero 2017

miércoles, 29 de diciembre de 2021 19:16

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 $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

a) Extraemos el sistema recurrente del algoritmo

$$T(n) = \begin{cases} c_1 & n = 1 \\ 21n + 2T(n/2) + c_2 & n > 1 \end{cases}$$

• Por ecuación característica:

$T(n) = 2T(\frac{n}{2}) + 21n + c_2 \rightarrow T(n) - 2T(\frac{n}{2}) = 21n + c_2 \rightarrow$

$\rightarrow T(n) - 2T(n/2) = 21n + c_2 \cdot n^0$ $[n = 2^k \leftrightarrow k = \log_2 n] \rightarrow$

$T(2^k) - 2T(2^{k-1}) = 21 \cdot 2^k + c_2 \rightarrow [T(2^k) = t_k]$

$\rightarrow t_k - 2t_{k-1} = 21 \cdot 2^k + c_2 \cdot 1^k \rightarrow$

$(x-2)(x-2)(x-1) = 0 \rightarrow (x-2)^2(x-1) = 0$ $\begin{cases} r_1: 2 \text{ (doble)} \\ r_2: 1 \end{cases}$

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

$= c_{11} \cdot 2^k + c_{12} \cdot k \cdot 2^k + c_2 = [k = \log_2 n]$

$T(n) = c_{11} \cdot n + c_{12} \cdot n \cdot \log_2 n + c_2 \rightarrow \boxed{T(n) \in O(n \log_2(n))}$

b) $T(n) = 2T(\frac{n}{2}) + 21n + C_2$

$= 2\left(2T(\frac{n}{4}) + 21\frac{n}{2} + C_2\right) + 21n + C_2$

$= 2^2 T(\frac{n}{2^2}) + 21n + 2C_2 + 21n + C_2$

$= 2^2 T(\frac{n}{2^2}) + 21n + 21n + C_2 + 2C_2$

$= 2^2\left(2T(\frac{n}{2^3}) + 21\frac{n}{2^2} + C_2\right) + (21n)\cdot 2 + C_2(1+2)$

$= 2^3 T(\frac{n}{2^3}) + 21n + 2^2 C_2 + (21n)\cdot 2 + C_2(1+2)$

$= 2^3 T(\frac{n}{2^3}) + 21n\cdot 3 + C_2(1^0 + 2^1 + 2^2) \rightarrow$

En general...

$= 2^i T(\frac{n}{2^i}) + 21n\cdot i + C_2 \sum_{j=0}^{i-1} 2^j \rightarrow$

$= 2^i T(\frac{n}{2^i}) + 21n\cdot i + C_2\left(2^i - 1\right) \rightarrow$

$= 2^i T(\frac{n}{2^i}) + 21ni + 2^i C_2 - C_2$

Para el caso base $\left(\frac{n}{2^i} = 1\right) \rightarrow (n = 2^i \leftrightarrow i = \log_2 n)$

$= n + 21n\cdot\log_2 n + nC_2 - C_2 =$

$= 21n\cdot\log_2 n + (C_2+1)n - C_2 \rightarrow$

$\boxed{T(n) \in O(n\log_2 n)}$.

c)

$n^2 \geq 21n\log_2(n)$

$n = 64 \rightarrow -3968 \rightarrow$ Burbuja más eficiente

$n = 128 \rightarrow -2432 \rightarrow$ Burbuja " "

$n = 256 \rightarrow 22528 \rightarrow$ Quicksort " "

$n = 160 \rightarrow 998 \rightarrow$ Quicksort

$n = 140 \rightarrow -1360 \rightarrow$ Burbuja

$n = 150 < 0 \rightarrow$ Burbuja

$n = 155 > 0 \rightarrow$ Quicksort

$n = 154 > 0$

$n = 153 > 0$

$n = 152 < 0$

Para $n \geq 153$ es preferible usar el algoritmo Quicksort, ya que es más eficiente

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.

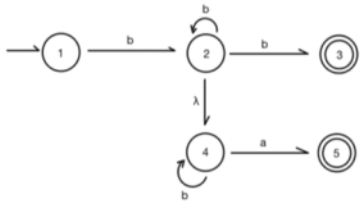
NOTA: El algoritmo de Gale & Shapley establece que si el número de hombres es el mismo que el de mujeres, siempre existe una solución con matrimonios estables. La descripción del algoritmo es la siguiente:

- 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.

(Reta por que no caiga este ejercicio porque no tengo ni idea).

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.

a)

1) $f'(1, ba) \rightarrow f'(CL(1), ba) \rightarrow f'(124, b) \rightarrow$

$f'(124, b) = 12, 44$

$f(12,44, a) = 154 \cap F \neq \emptyset \rightarrow$ ACEPTADA

2) $f'(1, ab) \rightarrow f'(CL(1), ab) \rightarrow f'(124, a,b) \rightarrow$

$f'(\emptyset, a) = \emptyset$

$f'(\emptyset, b) = \emptyset \cap F = \emptyset \rightarrow$ RECHAZADA

3) $f'(1, bb) = f'(CL(1), bb) \rightarrow f'(124, b) \rightarrow$

$f'(124, b) = 12, 44$

$f'(\{1,4\}, b) = \{2, 4\}$

$f'(\{2,4\}, b) = \{2, 3, 4\} \cap F \neq \emptyset \rightarrow$ ACEPTADA

4) $f'(1, b) \rightarrow f'(CL(1), b) = f'(\{1,4\}, b) \rightarrow$

$f'(\{1,4\}, b) = \{2, 4\} \cap F = \emptyset \rightarrow$ RECHAZADA

5) $f'(1, bba) \rightarrow f'(CL(1), bba) = f'(\{1,4\}, bba) \rightarrow$

$f'(\{1,4\}, b) = \{2, 4\}$

$f'(\{2,4\}, b) = \{2, 3, 4\}$

$f'(\{2,3,4\}, a) = \{5, 4\} \cap F \neq \emptyset \rightarrow$ ACEPTADA

b) El AFD mínimo equivalente se construye de la siguiente forma:

1) Transformar AFND a AFD:

$Q_0 = CL(1) \rightarrow Q_0 = \{1, 4\}$

$f'(Q_0, a) = \emptyset = M$

$f'(Q_0, b) = \{2, 4\} = Q_1$

$f'(Q_1, a) = \{5, 4\} = Q_2$

$f'(Q_1, b) = \{2, 3, 4\} = Q_3$

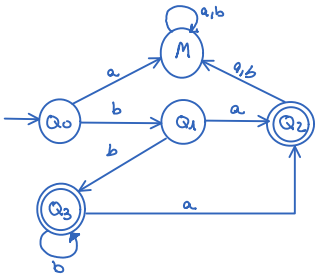
$f'(Q_2, a) = \emptyset = M$

$f'(Q_2, b) = \emptyset = M$

$f'(Q_3, a) = \{5, 4\} = Q_2$

$f'(Q_3, b) = \{2, 3, 4\} = Q_3$

	a	b
→ Q_0	M	Q_1
Q_1	Q_2	Q_3
* Q_2	M	M
* Q_3	Q_2	Q_3
M	M	M



2) Minimizar el autómata mediante el algoritmo de conjunto-cociente

	a	b
→ Q_0	M	Q_1
Q_1	Q_2	Q_3
* Q_2	M	M
* Q_3	Q_2	Q_3
M	M	M

$E/Q_0 = (C_0 = \{Q_0, Q_1\}, C_1 = \{Q_2, Q_3\}, C_2 = \{M\})$

Como son distinguibles los separamos: $f'(Q_0, a) = C_2$; $f'(Q_0, b) = C_0$

$f'(Q_1, a) = C_1$; $f'(Q_1, b) = C_1$

$E/Q_1 = (C_0 = \{Q_0\}, C_1 = \{Q_2, Q_3\}, C_2 = \{M\}, C_3 = \{Q_1\})$

Como son distinguibles los separamos: $f'(Q_2, a) = C_2$; $f'(Q_2, b) = C_2$

$f'(Q_3, a) = C_1$; $f'(Q_3, b) = C_1$

El $Q_2 = \{Q_0 = \{Q_0\}, Q_1 = \{Q_2\}, Q_2 = \{M\}, Q_3 = \{Q_1\}$
$Q_4 = \{Q_3\}\}$

Como tenemos un conjunto por cada estado, significa que ya tenemos el AFD mínimo.

c)

	a	b
→ Q_0	M	Q_1
Q_1	Q_2	Q_3
* Q_2	M	M
* Q_3	Q_2	Q_3
M	M	M

$f'(Q_0, ba) =$

$f'(Q_0, b) = Q_1$

$f'(Q_1, a) = Q_2 \in F \rightarrow$ ACEPTADA

$f'(Q_0, ab) =$

$f'(Q_0, a) = M$

$f'(M, b) = M \notin F \rightarrow$ RECHAZADA

$f'(Q_0, bb) =$

$f'(Q_0, b) = Q_1$
$f'(Q_0, b) = Q_3 \in F \rightarrow$ ACEPTADA

$f'(Q_0, b)$

$f'(Q_0, b) = Q_1 \notin F \rightarrow$ RECHAZADA

$f'(Q_0, bba)$

$f'(Q_0, b) = Q_1$
$f'(Q_1, b) = Q_3$
$f'(Q_3, a) = Q_2 \in F \rightarrow$ ACEPTADA

d)

	a	b
→ Q_0	M	Q_1
Q_1	Q_2	Q_3
* Q_2	M	M
* Q_3	Q_2	Q_3
M	M	M

$$\begin{cases} x_0 = bx_1 \\ x_1 = a x_2 + bx_3 + a + b \\ x_2 = \lambda \\ x_3 = a x_2 + b x_3 + a + b \end{cases}$$

$x_3 = bx_3 + (ax_2 + a + b) \rightarrow$

$x_3 = bx_3 + (a + b) \rightarrow$

$x_3 = b^*(a+b) \rightarrow$

$x_1 = ax_2 + bx_3 + a + b \rightarrow$

$x_1 = a\lambda + b(b^*(a+b)) + a + b \rightarrow$

$x_1 = bb^*(a+b) + a + b$

$x_0 = b(bb^*(a+b) + a + b) \rightarrow$

$x_0 = bbb^*(a+b) + ba + bb \rightarrow$

Expresión Regular Equivalente. → $\boxed{x_0 = bbb^*a + bbb^*b + ba + bb}$

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).

a) Para que una gramática sea LL(1) no debe ser recursiva por la izq.
La gramática propuesta lo cumple.

La condición necesaria y suficiente para que una gramática sea LL(1) es que la intersección de sus símbolos directores sea vacía:

Para ello calcularemos los conjuntos PRIMERO y SIGUIENTE:

	PRIMERO	SIGUIENTE
S	{a, (, n}	{), $}
A	{n}	{a,), $}
B	{a, λ}	{), $}
C	{(, λ}	{a,), $}

1.) $PRIM(aS) \cap PRIM((S)) \cap PRIM(A)$

$\{a\} \cap \{(\} \cap \{n\} = \emptyset$ ✓

2) $PRIM(aS) \cap SIG(B) =$

$\{a\} \cap \{), \$\} = \emptyset$ ✓

3) $PRIM((S)) \cap SIG(C) =$

$\{(\} \cap \{a,), \$\} = \emptyset$ ✓

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

b)
El algoritmo para generar la tabla de análisis sintáctico es el siguiente:

```
∀ 'A' → α
    ∀ 'a' terminal != λ ∈ PRIM(α) hacer
        Table[A][a] = α
    fA
    si λ∈PRIM(α)
        ∀ 'b' terminal != λ ∈ SIG(α)
            Table[A][a] = λ;
        fA
    fsi
fA
```

```
funcion analisis_sintactico()

    Apilar(#);
    Apilar($);

    leer_simbolo();

    mientras (!pila_vacia()) hacer
        switch cima_pila of

            case terminal:

                si (terminal == cima_pila) hacer

                    leer_simbolo();
                    Desapilar(simbolo);
                sino
                    error_sintactico();

            case no_terminal:

                si (Table[cima][simbolo] != error)

                    Desapilar(cima_pila);
                    Apilar(Table[cima][simbolo]);

                sino
                    error_sintactico();

                fsi

        fswitch
    fmientras

    si cima_pila == # →

        Escribir('ACEPTADA');

    sino
        error();
```

c)

PILA	ENTRADA	ACCIÓN
λ	baab((b))$	Apilar(#);
#	baab((b))$	Apilar(S);
S#	baab((b))$	error_sintactico();

d)

```
program Programa_Principal()
    SLA = leer_simbolo();
    S();
    si SLA != '$' entonces
        error_sintactico();
    fsi
fprograma
```

```
procedure Reconocer(simbolo S)
    si SLA == S entonces
        leer_simbolo();
    sino
        error_sintactico();
    fsi
fprocedure
```

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

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

```
funcion B()
    switch SLA
        case 'a':
            Reconocer('a');
            S();
        break;
        case ')', '$': /* nada */
        break;
        default: error_sintactico();
    fswitch
ffuncion
```

```
funcion C()
    switch SLA
        case '(':
            Reconocer('(');
            S();
```

```
            S();
            Reconocer (')');
        break;
        case 'a',')', '$': /* nada */
        break;
        default: error_sintactico();
    }switch
}funcion
```