Pauta de Corrección

Primer Certamen
Algoritmos y Complejidad

14 de noviembre de 2015

1. La expresión regular dada puede describirse mediante nuestras operaciones:

$$\mathscr{S} = \textsc{Seq}(\{a\} \times \{b\} + \{b\}) \times (\{a\} + \mathscr{E})$$

Representando los símbolos por $\mathscr{Z}$ y marcamos las a con $\mathscr{U}$ (una clase con un único elemento de tamaño cero), obtenemos:

$$\mathscr{S} = \textsc{Seq}(\mathscr{Z} \times \mathscr{Z} \times \mathscr{U} + \mathscr{Z}) \times (\mathscr{Z} \times \mathscr{U} + \mathscr{E})$$

El método simbólico traduce esto en:

$$\begin{aligned} S(z,u) &= \frac{1}{1-(z^2u+z)} \cdot (zu+1) \\ &= \frac{1+zu}{1-z-zu^2} \end{aligned}$$

Vemos que $S(z,1)$ es la función generatriz de la secuencia $\langle F_{n+2} \rangle_{n \geq 0}$, confirmando lo indicado en la pregunta.

Para obtener el número promedio de a en las palabras de largo n:

$$\overline{a}_n = \frac{[z^n]S_u(z,1)}{[z^n]S(z,1)}$$

Necesitamos:

$$\begin{aligned} S_u(z,u) &= \frac{z(1-z-z^2u)-(1+zu)(-z^2)}{(1-z-z^2u)^2} \\ &= \frac{z}{(1-z-z^2u)^2} \\ S_u(z,1) &= \frac{z}{(1-z-z^2)^2} \end{aligned}$$

Del enunciado sabemos:

$$S(z,1) = \frac{1+z}{1-z-z^2}$$

Con estas podemos evaluar $\overline{a}_n$.

Puntajes

Total	30
Descripción simbólica	5
Función generatriz multivariada	10
Plantear ecuación para $\overline{a}_n$	10
Derivada $S_u(z,1)$	5

2. Si siempre se consideran solo elementos vecinos, el método puede eliminar a lo más una inversión con cada intercambio, y el número promedio de operaciones estará acotado por abajo por el número promedio de inversiones en permutaciones. Y sabemos que esto es $n(n-1)/4$, y por tanto $\Omega(n^2)$.

Puntajes

Total	20
Número de intercambios es a lo menos el número de inversiones	10
Número promedio de inversiones es cuadrático	5
Cota inferior cuadrática	5

3. Suponiendo que "rellenamos" las matrices a tamaño par cada vez, la recurrencia para el tiempo de ejecución al multiplicar matrices de $N \times N$ es:

   $$T(N) = 7T(\lceil N/2 \rceil) + 4\lceil N/2 \rceil^2 \qquad T(1) \text{ dado}$$

   Si nos restringimos a potencias de 2 para N:

   $$T(N) = 7T(N/2) + N^2$$

   Se aplica dividir y conquistar, con $a = 7$, $b = 2$, $e = 2$; es $7 > 2^2$:

   $$T(N) = O(N^{\log_2 7})$$

   Notamos que $\log_2 7 < 3$, asintóticamente es mejor que el algoritmo tradicional.

Puntajes

Total		20
Plantear recurrencia "exacta"	8	
Restringir a $N = 2^k$	4	
Aplicar fórmula de dividir y conquistar	5	
Comparar con $O(N^3)$	3	

4. En concreto, un programa sería el dado en el listado 1.

```
int fibonacci(int n)
{
        if(n == 0)
                return 0;
        else if(n == 1)
                return 1;
        else
                return fibonacci(n - 1)
                        + fibonacci(n - 2);
}
```

Listado 1: Cálculo de números de Fibonacci

Sea c_n el número de llamadas que hace al calcular F_n (incluyendo la llamada original `fibonbacci(n)`). Vemos que para $n = 0$ y $n = 1$ se hace una llamada, y para $n \geq 2$ se hacen $c_{n-1} + c_{n-2} + 1$. Tenemos la recurrencia:

$$c_{n+2} = c_{n+1} + c_n + 1 \qquad c_0 = c_1 = 1$$

Aplicando funciones generatrices, definimos:

$$C(z) = \sum_{n \geq 0} c_n z^n$$

y la tradicional danza para resolver recurrencias da:

$$\frac{C(z) - 1 - z}{z^2} = \frac{C(z) - 1}{z} + C(z) + \frac{1}{1 - z}$$

Resulta:

$$\begin{aligned} C(z) &= \frac{1 - z + z^2}{1 - 2z + z^3} \\ &= \frac{2}{1 - z - z^2} - \frac{1}{1 - z} \end{aligned}$$

Los coeficientes son inmediatos:

$$c_n = 2F_{n+1} - 1$$

Puntajes

Total	25
Plantear la recurrencia	15
Valores iniciales	10

5. El algoritmo puede reorganizarse ligeramente para dar el programa del listado 2. Representamos cada dígito como un caracter, que en C admite al menos los valores entre 0 y 127, por lo que no hay problemas de rebalse.

```
void mult(char r[], char a[], char b[], int n)
{
        int i, j;
        int carry;

        /* Clear destination */
        for(i = 0; i <= 2 * n + 1; i++)
                r[i] = 0;

        /* Multiply */
        carry = 0;
        for(i = 0; i <= n; i++) {
                for(j = 0; j <= n; j++) {
                        /* One multiply, two sums */
                        r[i + j] += a[i] * b[j]
                                        + carry;
                        /* This is just adjusting the above */
                        carry = r[i + j] / 10;
                        r[i + j] %= 10;
                }
        }
        r[2 * n + 1] = carry;
}
```

Listado 2: Multiplicación dígito a dígito

Vemos que hay dos ciclos anidados, en cada iteración se efectúa una multiplicación entre dos dígitos y dos sumas. Claramente hay $O(n^2)$ operaciones.

Puntajes

Total	25
Planteo del algorithmo	20
Concluir que es $O(n^2)$	5

HvB/LaTeX2ε