Pauta de Corrección

Certamen Recuperativo
Algoritmos y Complejidad

23 de diciembre de 2017

1. Son iteraciones de punto fijo, $x_{n+1} = g(x_n)$, en general convergerán al punto fijo ξ si $|g'(\xi)| < 1$. Analizamos cada uno por turno.

   a)

   $$\begin{aligned} g_a(x) &= x(x+1) - 3 \\ g'_a(x) &= 2x + 1 \\ g'_a(\sqrt{3}) &= 2\sqrt{3} + 1 \end{aligned}$$

   Esto es mayor a 1, diverge.

   b)

   $$\begin{aligned} g_b(x) &= \frac{3}{x} \\ g'_b(x) &= -\frac{3}{x^2} \\ g'_b(\sqrt{3}) &= -1 \end{aligned}$$

   No converge.
   En todo caso, es claro que para $x_0 \neq \sqrt{3}$, esta iteración alterna entre x_0 y $3/x_0$.

c)

$$
\begin{aligned}
g_c(x) &= \frac{x^3+3}{x(x+1)} \\
g_c'(x) &= \frac{(3x^2)\cdot(x^2+x)-(x^3+3)\cdot(2x+1)}{x^2(x+1)^2} \\
&= \frac{(3x^4+3x^3)-(2x^4+x^3+6x+3)}{x^2(x+1)^2} \\
&= \frac{x^4+2x^3-6x-3}{x^2(x+1)^2} \\
g_c'(\sqrt{3}) &= \frac{3^2+2\cdot 3\cdot\sqrt{3}-6\sqrt{3}-3}{3(\sqrt{3}+1)^2} \\
&= \frac{6}{3(3+2\sqrt{3}+1)} \\
&= \frac{2}{4+2\sqrt{3}} \\
&= \frac{1}{2+\sqrt{3}}
\end{aligned}
$$

Claramente es menor a 1, converge linealmente.

Puntajes

Total	30
Comentarios previos	10
a) Cálculo de derivada, conclusión	6
b) Cálculo de derivada, conclusión	6
c) Cálculo de derivada, conclusión	8

2. Si visualizamos el árbol de llamadas, vemos que el número de veces que come maní es $p(n) = n$, y que en número de cervezas ingeridas es $b(n) = n - 1$.

   Recurrencias relevantes son, para algún $1 \leq m < n$:

   $$p(n) = p(m) + p(n-m) \qquad p(1) = 1$$

   Claramente $p(n) = n$ es solución. De la misma forma:

   $$b(n) = 1 + b(m) + b(n-m) \qquad b(1) = 0$$

   Es solución $b(n) = n - 1$. Esto corrobora las funciones sospechadas antes.

   Este mismo resultado se obtiene suponiendo funciones lineales para p y b como indica la pista, usando las recurrencias para obtener los coeficientes.

Puntajes

Total		25
Recurrencias para $p(n)$ y $b(n)$	19	
Verificar $p(n) = n$	3	
Verificar $b(n) = n - 1$	3	

3. En general, son aplicables en situaciones en las que buscamos un objeto con características dadas, que vamos construyendo paso a paso. Veamos cada técnica por turno.

   Algoritmo voraz: Construye el objeto buscado, en cada paso elige la mejor opción según un criterio local (sin considerar consecuencias futuras).

   Al nunca reconsiderar sus decisiones, y solo analizar una alternativa en cada paso, suelen ser algoritmos simples y muy eficientes. Pero garantizan hallar la solución solo si hay un criterio local adecuado, lo que es poco frecuente.

   Programación dinámica: Construye el objeto buscado, en cada paso considera todas las alternativas relevantes. Esto genera subproblemas a ser resueltos recursivamente; si muchos de ellos se repiten, conviene recordar subproblemas ya resueltos y sus soluciones para evitar rehacer el trabajo (memoización) o directamente organizar el trabajo de manera de resolver sistemáticamente todos los subproblemas en orden, de forma que cuando se requiera la solución de uno de ellos esta ya esté (programación dinámica).

   Backtracking: Construye el objeto buscado, en cada paso considera por turno todas las alternativas relevantes. Si no se repiten subproblemas en la búsqueda recursiva, no tiene sentido registrar subproblemas y soluciones.

Puntajes

Total		20
Ámbito general común	5	
Algoritmo voraz	5	
Programación dinámica/memoización	5	
Backtracking	5	

4. Cada punto por turno.

   a) Nos interesa OPT(n), la máxima ganancia total esperada si se consideran las posiciones $1,\ldots,n$. Es claro que OPT(0) = 0, si no hay posiciones a considerar la ganancia total esperada es nula.

   b) Consideremos la posición k. Tenemos dos posibilidades:

   Instalamos un restaurante en x_k: Es claro que en tal caso no pueden haber restaurantes previos en el rango $x_k - d$ a x_k. Los restaurantes deberán estar ubicados en forma óptima hasta $x_k - d$, o sea, en este caso el óptimo es OPT(x_j) + g_k, donde j es tal que $x_j \leq x_k - d$ y $x_{j+1} > x_k - d$.

   No se instala el restaurante en x_k: En este caso, el óptimo es OPT(x_{k-1}).

   Elegimos la mejor entre estas dos alternativas.

   c) Definimos un arreglo OPT[$n+1$], que llenamos sistemáticamente desde OPT[0] usando los puntos anteriores. El resultado buscado es OPT[n]. Nuestro algoritmo 1 va actualizando el índice j de la última posición a considerar antes de la actual.

Algoritmo 1: Empanadas To Go

```
function EmpanadasToGo(x, g, d)
    OPT[0] ← 0
    j ← 0
    for k ← 1 to n do
        while j < n ∧ x_{j+1} + d < x_k do
            j ← j + 1
        end
        OPT[k] ← máx{OPT[k − 1], OPT[j] + g_k}
    end
    return OPT[n]
end
```

Puntajes

Total		30
a) Resultado a obtener, caso base		5
b) Recurrencia para OPT		10
Caso instalar en x_k	5	
Caso no instalar en x_k	5	
c) Algoritmo		15
Orden de cálculo	5	
Pseudocódigo	10	

5. Corresponde hacer un análisis amortizado. Como hay una única operación, lo más sencillo es usar el método agregado. Sea n el número máximo de widgets fabricados. Es claro que George Akeley hará un total de n viajes, el costo por este concepto es nv. El número total de cambios de dígito es (el último dígito cambia n veces, el dígito de las decenas $\lfloor n/10 \rfloor$ veces, y así sucesivamente):

$$\begin{aligned} n + \lfloor n/10 \rfloor + \lfloor n/10^2 \rfloor + \cdots &= \sum_{k \geq 0} \lfloor n/10^k \rfloor \\ &< \sum_{k \geq 0} n/10^k \\ &= n \sum_{k \geq 0} 10^{-k} \\ &= n \cdot \frac{1}{1 - 10^{-1}} \\ &= \frac{10n}{9} \end{aligned}$$

El costo total de n operaciones está acotado por:

$$nv + \frac{10nc}{9} = n\left(v + \frac{10c}{9}\right)$$

con lo que el costo amortizado por operación está acotado por $v + 10c/9$.

Puntajes

Total	30
Usar método agregado	5
Costo de viajes	5
Costo de cambios de dígito	10
Cotas manejables	5
Costo amortizado	5

HvB/LaTeX2ε