CC40A: Diseño y Análisis de Algoritmos
Pauta Control 1 - Semestre Otoño 2002

Profesor: Ricardo Baeza Yates
Prof. Auxiliar: Rodrigo Paredes

Tiempo: 1:30 hras, con apuntes y en hojas separadas

P1 (2 ptos.)

Una cierta especie de conejos tiene un patrón de reproducción muy definido. La primera camada, cuando son jóvenes, consiste en dos hembras y dos machos, mientras que la segunda camada, cuando están más viejos, consiste solamente de una pareja. Suponiendo que los conejos se aparean libremente una vez al mes, calcule cuántos conejos hay después de n meses si al comienzo sólo hay una pareja de conejos.

Indique si este número es menor, igual o mayor a F_n, el número de Fibonacci enésimo.

Respuesta

Considerando que los conejos viven felices para siempre la secuencia de conejos es:

n	nacidos	f_n
1	2	2
2	4	6
3	$2+8$	16
4	$4+20$	40
5	$10+48$	98

Luego la recurrencia que se genera es:

$$f_n = 2 \cdot f_{n-1} + f_{n-2} + 2$$

Luego:

$$f_n \geq 2 \cdot f_{n-1} \Rightarrow f_n \geq 2^n > Fibonacci_n$$

Hasta aca a un punto.

La solución exacta de la recurrencia se puede calcular de varias formas, y la solución es:
$f_n = 1.207 \cdot 2.4142^n - 0.207 \cdot (-0.4142)^n - 1$

Y con la solucion exacta de la recurrencia tenemos el otro punto.

Ahora, si pensamos que los conejos tienen sus dos camadas y luego mueren:

Sean J_i los conejos jóvenes y V_i los viejos.

Entonces los viejos son $V_i = J_{i-1}$

Esto implica que $J_i = 2 \cdot J_{i-1} + J_{i-2}$, $J_1 = 1$, $J_2 = 2$ y hacemos el mismo razonamiento anterior.

La solución de esta recurrencia es:
$f_n = 0.707 \cdot 2.4142^n - 0.707 \cdot (-0.4142)^n$

P2 (2 ptos.)

Cada día en Chile el servicio metereológico nacional toma temperaturas en distintos puntos del país, los cuales son recibidos como tuplas de la forma $(id_i, h1_i, h2_i, T_i)$ que indican que en la ciudad id_i entre las horas $h1_i$ y $h2_i$ la temperatura fue T_i.

Diseñe un algoritmo eficiente en pseudo-código que dadas n tuplas, retorna un conjunto de tuplas de igual forma que indica que la temperatura en esa ciudad y esas horas fue la temperatura máxima en todo Chile. En el ejemplo de la Figura 1 la respuesta está marcada con líneas más oscuras (note que pueden haber horas sin ninguna temperatura).

Para simplificar el algoritmo puede suponer que los identificadores de ciudad y las horas son todos números enteros y que todas las temperaturas son distintas.

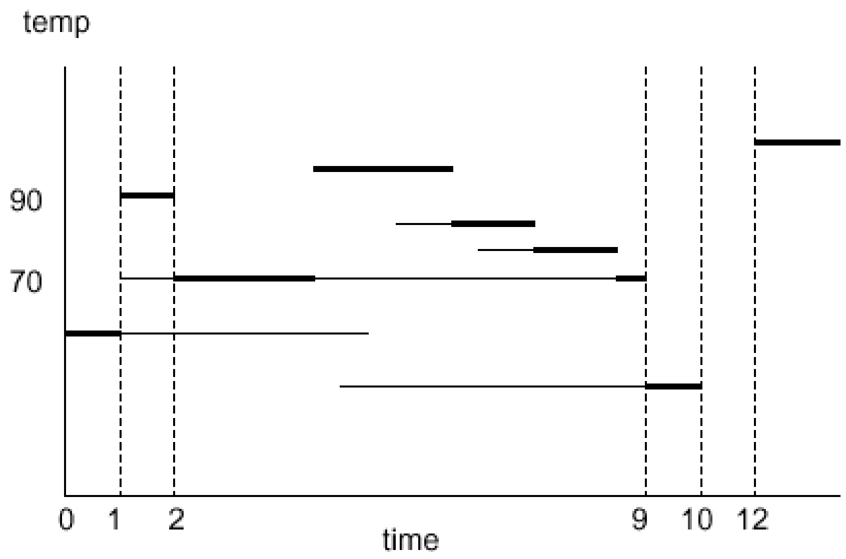

Figura 1: Gráfico de ejemplo P2.

Respuesta

Considerando que los perfiles los construyo con listas enlazadas e inserto preservando el orden.

`perfil` $\leftarrow$ `merge(`$tuplas$`)`

`merge` $(tuplas)$
  `if` $|tuplas| < 2$
    `return` $tuplas$
  `else`
    $m \leftarrow |tuplas|/2$
    $tuplas_1 \leftarrow \{tupla_i, i \in [1, m]\}$
    $tuplas_2 \leftarrow \{tupla_i, i \in [m+1, |tuplas|]\}$

```
    perfil₁ ← merge(tuplas₁)
    perfil₂ ← merge(tuplas₂)
    perfil ← perfil₁
    while (perfil₂ ≠ ∅)
      tup₂ ← perfil₂.getFirstTupla()
      perfil₂.extractFirstTupla()
      tupP ← tupla de perfil que contiene a tup₂.ini
      if tupP.ini ≠ tup₂.ini
        tupP1 ← (tupP.ciudad, tupP.ini, tupP.T, tup₂.ini)
        tupP2 ← (tupP.ciudad, tup₂.ini, tupP.T, tupP.fin)
        reemplaza en perfil tupP por tupP1, tupP2
      else tupP2 ← tupP
      if (tup₂.T > tupP2.T)
        tupP3 ← (tupP.ciudad, tup₂.fin, tupP.T, tup₂.fin)
        reemplaza en perfil tupP2 por tup₂ y tupP3
  return perfil
```

La gracia de esto es que $tupp_P$ se puede calcular incrementalmente, es decir, cuando tenga que revisar la siguiente tupla de $perfil_2$ retomo la búsqueda en $perfil$ desde la tupla anterior.

Otra forma de ver esto es pensar en dos hechos:

- Los intervalos son discretos.
- Son sólo 24 intervalos.

Esto permite hacer lo siguiente:

```
Ciudades ← [nul, nul, ... , null] // 24 nuls.
Temperaturas ← [-273.15, -273.15, ... , -273.15] // 24 -273.15.
Para todas las tuplas t del conjunto
  Para cada intervalo i cubierto por la tupla t
    if t.T > Temperaturas[i]
      Temperaturas[i] ← t.T
      Ciudades[i] ← t.Ciudad
construir el perfil con Temperaturas y Ciudades y retornarlo
```

Una solución correcta de $O(n\log(n))$ da 2 puntos
Una solución correcta $O(n^2)$ da 1 punto
Otras soluciones dan 0.5 puntos

P3 (2 ptos.)

Para celebrar la nota obtenida en este control, el curso decide ir a un restaurante tipo buffet que permite comer por un precio fijo todo lo que quepa en un gran plato. El buffet es una mesa larga con distintas opciones las cuales no se conocen de antemano (y si perdemos tiempo en ver cuales son muchas de ellos se agotarán, en particular las mejores) con algunas restriciones: un volumen de porción y un número máximo de porciones que uno se puede servir.

Considere un plato con volumen máximo M y n opciones de comida, cada una con porciones de volumen v_i de las cuales se pueden tomar hasta p_i porciones.

a) Diseñe un algoritmo en-línea personal en pseudo-código tomando en cuenta que cuando llega a la opción i, esta le gusta g_i por unidad de volumen (donde g_i puede ser negativo si no le gusta o 0 si le es indiferente), de modo de maximizar cuánto le gusta su plato final de comida. Puede suponer que Ud. no es mañoso y que existe al menos un g_i positivo.

b) Indique cuál sería el algoritmo óptimo si conocemos las opciones por adelantado y exprese el valor final del plato con una fórmula.

c) Indique cómo Ud. definiría la competitividad de su algoritmo en este caso. ¿Puede decir algo del rango de valores que puede tomar?

Respuesta

a) Mi estrategia es: llenar el plato con cosas ricas, y que aumente lo rico del plato en promedio.

   Sea $< rico >$ el promedio ponderado del sabor del plato:

   1 $< rico > \leftarrow 0$

   2 mientras paso por la barra
   if ($g_i >\ < rico >$) agrego lo que pueda al plato (limitado por las p_i porciones, por lo que quede disponible en la barra para sacar de comida y por la capacidad M)

   Cualquier algoritmo razonable que no devuelva comida ni se devuelva en la cola está bien. (0.7 puntos)

b) Ordenamos los g_i de mayor a menor. Sean $g_1, \ldots, g_k$ los mayores que 0 (hay al menos uno). Luego el óptimo sera:

$$\max_{1 \leq j \leq k} \sum_{i=1}^{j-1} \left(g_i \cdot v_i + g_j \cdot \left(M - \sum_{i=1}^{j} v_i \right) \right)$$

   (0.7 puntos)

c) En este problema la competitividad hay que definirla al revés:

$$c = \frac{C_{opt}}{C_{on-line}}$$

   ya que estamos maximizando el "gusto" de la comida y no minimizando el tiempo (0.6 ptos)