UNIVERSIDAD NACIONAL
DE EDUCACIÓN A DISTANCIA

Programación III

Solución

Prueba Presencial

Prueba Presencial

Segunda semana

Febrero de 2008

Duración: **2 horas**
Material permitido: **NINGUNO**

Cuestión 1 (2 puntos). En la práctica obligatoria del presente curso 2007/2008 se ha tenido que diseñar y desarrollar un algoritmo para resolver el problema del nonograma. Dado el cuadrado de 4x4 de la figura, trazar el algoritmo que lo resuelve tal como lo hace el desarrollado para la práctica y explicar cada paso. Hacer al menos 3 niveles y al menos 2 backtrakings.

Restricciones del nonograma:

	1 1	1 1	2	1 2
2 1	X	X		X
1			X	
3		X	X	X
1 1	X			X

Solución:
En función del algoritmo desarrollado por el alumno en la práctica.

Cuestión 2 (2 puntos). Sea T[1..n] con k elementos (k<n) un montículo de mínimos. Se pide programar una función "flotar" recursiva que dado un nuevo elemento T[k+1] restaure la propiedad de montículo en T. Una función iterativa que lo resuelva puntuará cero puntos.

Solución:

```
PROCEDURE Flotar(i: INTEGER; VAR T: TipoMonticulo);

VAR
  i_padre: INTEGER;

BEGIN
  i_padre := i DIV 2 ;


  IF (i > 1) AND T[i]<T[i_padre]) THEN
     Intercambia(T[i],T[i_padre]);
     Flotar(i_padre,T);
  END;

END Flotar;
```

Cuestión 3 (2 puntos). Dado *n* potencia de 2, escribe un algoritmo recursivo que calcule en tiempo logarítmico el valor de a^n suponiendo que solo se pueden realizar multiplicaciones y que éstas tienen coste unitario. Demuestre el coste mediante la ecuación de recurrencia. No justifique el esquema usado, aplíquelo.

Solución:

fun *exp (a:**entero**,n:**natural**)* ***dev entero***

  si *n=1* ***entonces dev*** *a*
  sino si *n=0* ***entonces dev*** *1*
  sino

t ← exp(a,n DIV 2)
*dev t*t*
fsi
ffun

Ecuación de recurrencia:

$$t(n) \begin{cases} cn^k & \text{si} \quad 1 \leq n < b \\ l\, t(n \div b) + cn^k & \text{si} \quad n \geq b \end{cases}$$

La solución depende de los valores de *l,b,k*

$$T(n) \in \begin{cases} \Theta(n^k) & \text{si } l < b^k \\ \Theta(n^k \log n) & \text{si } l = b^k \\ \Theta(n^{\log_b l}) & \text{si } l > b^k \end{cases}$$

En este problema: l=1, b=2, k=0, luego el caso es $T(n) \in \Theta(n^k \log n)$, por lo que el coste es $T(n) \in \Theta(\log n)$

Problema (4 puntos). Una flota de 4 camiones (T1..T4) debe transportar cargamento variado a otras tantas ciudades (C1..C4). El coste de adjudicar el transporte varía en función de la distancia y de la peligrosidad del trayecto y se resume en la tabla adjunta. Exponer un algoritmo que calcule de manera óptima a quién encargarle qué destino de manera que en total el coste sea mínimo.

	T1	**T2**	**T3**	**T4**
C1	24	45	12	34
C2	56	56	12	76
C3	90	67	32	54
C4	32	23	12	23

La resolución de este problema debe incluir, por este orden:

1. Elección del esquema más apropiado, el esquema general y explicación de su aplicación al problema (0,5 puntos).
2. Descripción de las estructuras de datos necesarias (0.5 puntos).
3. Algoritmo completo a partir del refinamiento del esquema general (2,5 puntos).
4. Estudio del coste del algoritmo desarrollado (0.5 puntos).

Solución:

El problema de las asignaciones Seccion 9.7 *Brassad&Bradley*
Similar al problema 4.6 del libro *Esquemas Algorítmicos: Enfoque metodológico y problemas resueltos*. Gonzalo, J. Rodríguez, M.