Departamento de Estadística, Matemáticas e Informática
Grado en Ingeniería Informática en Tecnologías de la Información

Metodología de la Programación y Algoritmia

Convocatoria de Junio 2014
SOLUCIÓN

Apellidos ____

Nombre ____ **DNI** ____________________

1.- Diseña un algoritmo lo más eficiente posible que además de obtener el coste de los caminos más cortos desde el vértice 1, imprima los caminos. Para el grafo del ejemplo el algoritmo imprimirá:

```
1 -> 2
1 -> 5 -> 3
1 -> 5 -> 3 -> 4
1 -> 5
```

Explica cómo funciona tu algoritmo, qué tipo de estrategia sigue y por qué e impleméntalo en pseudocódigo. Puedes utilizar como base el algoritmo de Dijkstra. La función imprimir(…) imprime el valor de la variable o dato que se introduce como argumento.

SOLUCIÓN:

Una posible solución, planteada en los ejercicios del tema de algoritmos voraces, es modificar el algoritmo de Dijkstra añadiendo un vector V donde almacenamos para cada vértice cuál es el vértice anterior en el camino.

Para imprimir el camino desde el vértice 1 a cualquier otro vértice utilizamos una función recursiva llamada ImprimirCamino.

```
función Dijkstra(C:entero+U{0}[n], P:real+[n,n], D:&real[n])
   i,j,x:natural
   V[n]:entero+

   para i←1 hasta n hacer
      Di ← P1,i
      Vi ← 1
   fpara

   para i←1 hasta n-1 hacer
      x ← elegirvértice_costemin(C,D)
      Cx ← 0
      para j←1 hasta n hacer
         si pertenece(C,j)
            si Dj > Dx + Px,j
               Dj ← Dx + Px,j
               Vj ← x
            fsi
         fsi
      fpara
   fpara

   para i←2 hasta n hacer
      si Di = INF
         imprimir("No hay camino al vértice ")
         imprimir(i)
      si no
         ImprimirCamino(i,V)
      fsi
      imprimir(SaltoLinea)
   fpara
ffunción
```

Metodología de la Programación y Algoritmia

Convocatoria de Junio 2014

SOLUCIÓN

```
función ImprimirCamino(destino:entero+, V:entero[n])
    si destino = 1
       imprimir("1")
    si no
        ImprimirCamino(V_destino,V)
        imprimir(" -> ")
        imprimir(destino)
    fsi
ffunción
```

2.- Distribución partidos de las jornadas de fútbol.

2.a) Explica **detalladamente** el funcionamiento de un algoritmo que obtenga una distribución de partidos que cumpla las restricciones del problema para cualquier número de centros participantes e indica de qué tipo de estrategia se trata y por qué.

2.b) Se han inscrito los siguientes 9 Institutos de Educación Secundaria: Carrús, Cayetano Sempere, L'Assumpció, Misteri d'Élx, Nit de l'Albá, Pere Ibarra, Sixto Blanco,Tirant Lo Blanc y Victoria Kent.
Aplica el algoritmo a este caso, indicando los valores que se van asignando de distribuciones de partidos hasta llegar a la distribución final de partidos.

<u>SOLUCIÓN:</u>

2.a)
Se puede utilizar el algoritmo del Torneo de pádel, visto en el tema de Divide y Vencerás.
Este algoritmo está descrito en el Tema 3 del libro recomendado en la bibliografía

Análisis y diseño de algoritmos
Guerequeta R., Vallecillo A.
Servicio de publicaciones de la Universidad de Málaga (2000)

2.b)
Las llamadas recursivas que se realizan a la función son las siguientes:

1º) Jornadas(9) → 2º) Jornadas(10) → 3º) Jornadas(5) → 4º) Jornadas(6) → 5º) Jornadas(3) →
6º) Jornadas(4) → 7º) Jornadas(2)

Cada equipo lo vamos a renombrar por un número del 1 al 9.

1: Carrús
2: Cayetano Sempere
3: L'Assumpció
4: Misteri d'Élx
5: Nit de l'Albá
6: Pere Ibarra
7: Sixto Blanco
8: Tirant Lo Blanc
9: Victoria Kent

La distribución para 5 equipos (llamada 3º) está hecha en los ejercicios de clase y su resultado es:

	Días				
Equipos	**1**	**2**	**3**	**4**	**5**
1	2	3	4	5	0
2	1	5	3	0	4
3	0	1	2	4	5
4	5	0	1	3	2
5	4	2	0	1	3

A partir de estos valores vamos a completar los resultados para las llamadas 1º y 2º.

Metodología de la Programación y Algoritmia
Convocatoria de Junio 2014
SOLUCIÓN

2º) Jornadas(10)

	Días								
Equipos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	2	3	4	5	**6**	7	8	9	10
2	1	5	3	**7**	4	8	9	10	6
3	**8**	1	2	4	5	9	10	6	7
4	5	**9**	1	3	2	10	6	7	8
5	4	2	**10**	1	3	6	7	8	9
6	7	8	9	10	**1**	5	4	3	2
7	6	10	8	**2**	9	1	5	4	3
8	**3**	6	7	9	10	2	1	5	4
9	10	**4**	6	8	7	3	2	1	5
10	9	7	**5**	6	8	4	3	2	1

1º) Jornadas(9)

	Días								
Equipos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	2	3	4	5	6	7	8	9	**0**
2	1	5	3	7	4	8	9	**0**	6
3	8	1	2	4	5	9	**0**	6	7
4	5	9	1	3	2	**0**	6	7	8
5	4	2	**0**	1	3	6	7	8	9
6	7	8	9	**0**	1	5	4	3	2
7	6	**0**	8	2	9	1	5	4	3
8	3	6	7	9	**0**	2	1	5	4
9	**0**	4	6	8	7	3	2	1	5

Esta sería la distribución final para las jornadas. Por ejemplo, el primer día los partidos que se juegan son:

Partido 1:	1: Carrús	2: Cayetano Sempere
Partido 2:	3: L'Assumpció	8: Tirant Lo Blanc
Partido 3:	4: Misteri d'Élx	5: Nit de l'Albá
Partido 4:	6: Pere Ibarra	7: Sixto Blanco

Descansa: 9: Victoria Kent

3.- Dado el siguiente algoritmo

```
función junio14(a:entero+ U {0}, b:entero):entero
   si a < 3
      devolver a + b
   si no
       devolver a + junio14(a / 3, b – 2)
   fsi
función
```

3.a) Indica de qué tipo de recursividad se trata y por qué.
3.b) Calcula la complejidad asintótica y justifica tu respuesta.
3.c) Convierte el algoritmo a iterativo.

<u>SOLUCIÓN:</u>

3.a) Indica de qué tipo de recursividad se trata y por qué.

Se trata de recursividad lineal no final porque en el caso general solamente hay una llamada a la función y al completar la última llamada recursiva no se obtiene lo solución del problema directamente pues quedan operaciones pendientes de realizar. La solución se va calculando sumando al valor del parámetro 'a' los resultados de cada una de las llamadas recursivas.

Metodología de la Programación y Algoritmia
Convocatoria de Junio 2014
SOLUCIÓN

3.b) Calcula la complejidad asintótica y justifica tu respuesta.

La variable que indica el tamaño del problema de este algoritmo es el parámetro a. Sea n=a.

T(n) = 3 si n ≤ 2
T(n) = 6 + T(n/3) si n > 2

Este problema es del tipo de disminución del tamaño del problema por división.

$$T(n) = \begin{cases} c_1 & \text{si } 0 \le n \le n_1 \\ a \cdot T(n/b) + c \cdot n^k & \text{si } n > n_1 \end{cases} \qquad T(n) \in \begin{cases} O(n^k) & \text{si } a < b^k \\ O(n^k \cdot \log n) & \text{si } a = b^k \\ O(n^{\log_b a}) & \text{si } a > b^k \end{cases}$$

Identificamos los elementos: $c_1 = 3$, $n_1 = 2$, a = 1, b = 3, c = 6, k = 0.
Como $a = b^k = 1$, la complejidad asintótica es O(logn).

El ejercicio también puede resolverse utilizando la técnica del desplegado.

3.c) Convierte el algoritmo a iterativo.

Aplicamos el esquema de conversión de recursividad lineal no final a iterativo con pila porque es necesario utilizar algún tipo de almacenamiento auxiliar al no existir inversa para la función de transformación del parámetro a de la función.

```
función junio14_iterativo(a:entero+U{0}, b:entero):entero
   a0: entero+U{0}
   b0,s: entero
   p:pila

   a0 ← a
   b0 ← b
   mientras !(a0 < 3) hacer
      apilar(p , a0)
      a0 ← a0 / 3
      b0 ← b0 - 2
   fmientras
   s ← a0 + b0
   mientras !(p=ø) hacer
      a0 ← cima(p)
      desapilar(p)
      s ← a0 + s
   fmientras
   devolver s
ffunción
```