Metodología de la Programación y Algoritmia

Convocatoria de Junio 2019
SOLUCIÓN

1.- Dado el siguiente algoritmo

```
función recursivo19(x:entero, y:entero):entero
   si x ≤ y
      devolver y - x
   si no
      devolver recursivo19(x / 2 , y + 3) – x * y
   fsi
ffunción
```

1.a) Realiza la traza de llamadas recursivas que se generan para x=80 e y=1 y cuál es el resultado final.
1.b) Indica qué tipo de recursividad es y por qué y obtén su versión iterativa aplicando el esquema general según el tipo de recursividad de que se trate. Para la resolución de este apartado solamente se admite el uso de una pila o de algún tipo de almacenamiento si es estrictamente necesario.

SOLUCIÓN

1.a) Traza para x=80, y=1

1º) recursivo19(80,1) = –300 – 80*1 = –380
2º) recursivo19(40,4) = –140 – 40*4 = –140 – 160 = –300
3º) recursivo19(20,7) = 0 – 20*7 = 0 – 140 = –140
4º) recursivo19(10,10) = 0

Resultado final: -380

1.b) Tipo de recursividad es, por qué y versión iterativa aplicando el esquema general según el tipo de recursividad de que se trate. Para la resolución de este apartado solamente se admite el uso de una pila o de algún tipo de almacenamiento si es estrictamente necesario.

Se trata de recursividad lineal no final porque se realiza una llamada recursiva para cada función y el resultado final se obtiene combinado los resultados de las llamadas recursivas que se van realizando. Es necesario utilizar una pila para ir guardando los valores de x0 para los valores de y0 se puede aplicar función inversa.

```
función iterativo19(x:entero, y:entero):entero
   x0,y0,s: entero
   p : pila

   x0 ← x
   y0 ← y
   mientras x0 > y0 hacer
      apilar(p,x0)
      x0 ← x0 / 2
      y0 ← y0 + 3
   fmientras
   s ← y0 – x0
   mientras !(p = ø) hacer
      x0 ← cima(p)
      desapilar(p)
      y0 ← y0 – 3
      s ← s – x0*y0
   fmientras
   devolver s
ffunción
```

Metodología de la Programación y Algoritmia

Convocatoria de Junio 2019
SOLUCIÓN

2.- Una empresa de reparto tiene registrado cuál el tiempo aproximado que lleva realizar un reparto por carretera desde el ayuntamiento de una ciudad al ayuntamiento de otra y quiere obtener cuál es el menor tiempo de reparto entre todo par de ciudades, teniendo en cuenta que para ir de una ciudad a otra se puede pasar por ciudades intermedias.

SOLUCIÓN

En este caso se puede utilizar el Algoritmo de Floyd. Este algoritmo se ha estudiado en el tema de Programación Dinámica, concretamente en los ejercicios prácticos del tema. El alumno puede consultar el funcionamiento del algoritmo en el ejercicio correspondiente y obtener más información consultando la bibliografía. A continuación se realiza la traza del pseudocódigo del algoritmo que se encuentra en los ejercicios del tema.

k=1

	Ciudad 1	**Ciudad 2**	**Ciudad 3**	**Ciudad 4**	**Ciudad 5**
Ciudad 1	0	20	8	15	30
Ciudad 2	21	0	17	9	8
Ciudad 3	5	11	0	10	6
Ciudad 4	17	6	7	0	10
Ciudad 5	35	10	7	12	0

k=2

	Ciudad 1	**Ciudad 2**	**Ciudad 3**	**Ciudad 4**	**Ciudad 5**
Ciudad 1	0	20	8	15	**28**
Ciudad 2	21	0	17	9	8
Ciudad 3	5	11	0	10	6
Ciudad 4	17	6	7	0	10
Ciudad 5	**31**	10	7	12	0

k=3

	Ciudad 1	**Ciudad 2**	**Ciudad 3**	**Ciudad 4**	**Ciudad 5**
Ciudad 1	0	**19**	8	15	**14**
Ciudad 2	21	0	17	9	8
Ciudad 3	5	11	0	10	6
Ciudad 4	**12**	6	7	0	10
Ciudad 5	**12**	10	7	12	0

k=4

	Ciudad 1	**Ciudad 2**	**Ciudad 3**	**Ciudad 4**	**Ciudad 5**
Ciudad 1	0	19	8	15	14
Ciudad 2	21	0	**16**	9	8
Ciudad 3	5	11	0	10	6
Ciudad 4	12	6	7	0	10
Ciudad 5	12	10	7	12	0

k=5

	Ciudad 1	**Ciudad 2**	**Ciudad 3**	**Ciudad 4**	**Ciudad 5**
Ciudad 1	0	19	8	15	14
Ciudad 2	**20**	0	**15**	9	8
Ciudad 3	5	11	0	10	6
Ciudad 4	12	6	7	0	10
Ciudad 5	12	10	7	12	0

La última matriz contiene los costes de los caminos más cortos entre cada par de ciudades. Así por ejemplo para ir de la ciudad 1 a la 2 el menor coste es 19.

Metodología de la Programación y Algoritmia
Convocatoria de Junio 2019
SOLUCIÓN

3.- Calcula la complejidad asintótica del siguiente algoritmo y **justifica** tu respuesta. La complejidad asintótica de obtener(A:&real[n]):real es O(n) y la de combinar(X:&real[n],Y:&real[n],z:real):real es O(nlogn).

SOLUCIÓN

La variable que indica el tamaño del problema es x.

Para cada línea indicamos cuál es su complejidad asintótica por niveles.

```
      función junio19(A:real[x],B:real[x]):real  Nivel 1   Nivel 2   Nivel 3
(1)      i,j,n:entero
(2)      v1,v2:real
(3)      v1 ← 0                                  O(1)
(4)      n ← 50                                  O(1)
(5)      i ← 1                                   O(1)
(6)      mientras i ≤ x hacer                    O(x)
(7)         j ← 1                                         O(1)
(8)         mientras j ≤ n hacer                          O(1)
(9)            v1 ← v1 + obtener(A)*obtener(B)                      O(x)
(10)           j ← j * 2                                            O(1)
(11)        fmientras
(12)        i ← i + 1                                     O(1)
(13)     fmientras
(14)     v2 ← combinar(A,B,v1)                   O(xlogx)
(15)     devolver v1/v2                          O(1)
      ffunción
```

En el bucle de la línea (6) el número de iteraciones varía en función de x y en el de la línea (6) el número de iteraciones depende de una constante n que vale 50, con lo cual el nºiteraciones de este bucle es del O(1). L La condición en ambos bucles es del O(1), con lo cual la complejidad de estas líneas son O(x) y O(1), respectivamente. En la línea 9, la función obtener se realiza dos veces de forma secuencial, con lo cual como el resto de operaciones son del orden constante la complejidad de esta línea corresponde al O(x), que es la complejidad de la función obtener. En la línea (14) se aplica la complejidad de la función combinar.

Para resolver el bucle de las líneas (8) – (11), aplicamos la regla de la secuencia de instrucciones a las instrucciones que están dentro del bucle: max{O(x), O(1)} = O(1) y resolvemos la complejidad del bucle aplicando la regla del producto O(1)*O(x) = O(x).

```
      función junio19(A:real[x],B:real[x]):real  Nivel 1   Nivel 2
(1)      i,j,n:entero
(2)      v1,v2:real
(3)      v1 ← 0                                  O(1)
(4)      n ← 50                                  O(1)
(5)      i ← 1                                   O(1)
(6)      mientras i ≤ x hacer                    O(x)
(7)         j ← 1                                         O(1)
(8)         mientras j ≤ n hacer                          O(x)
(9)            v1 ← v1 + obtener(A)*obtener(B)
(10)           j ← j * 2
(11)        fmientras
(12)        i ← i + 1                                     O(1)
(13)     fmientras
(14)     v2 ← combinar(A,B,v1)                   O(xlogx)
(15)     devolver v1/v2                          O(1)
      ffunción
```

Metodología de la Programación y Algoritmia

Convocatoria de Junio 2019
SOLUCIÓN

A continuación, resolvemos el bucle de las líneas (6) – (13) procediendo del mismo modo. Las instrucciones que están dentro del bucle son del orden de max{O(1), O(x), O(1)} = O(x) y aplicamos la regla del producto O(x)*O(x) = O(x^2).

```
      función junio19(A:real[x],B:real[x]):real  Nivel 1
(1)      i,j,n:entero
(2)      v1,v2:real
(3)      v1 ← 0                                  O(1)
(4)      n ← 50                                  O(1)
(5)      i ← 1                                   O(1)
(6)      mientras i ≤ x hacer                    O(x^2)
(7)         j ← 1
(8)         mientras j ≤ n hacer
(9)            v1 ← v1 + obtener(A)*obtener(B)
(10)           j ← j * 2
(11)        fmientras
(12)        i ← i + 1
(13)     fmientras
(14)     v2 ← combinar(A,B,v1)                   O(xlogx)
(15)     devolver v1/v2                          O(1)
      ffunción
```

Finalmente aplicamos la regla de instrucciones secuenciales a las instrucciones del nivel 1 y se obtiene que la complejidad del algoritmo junio19 es max{O(1), O(1), O(1), O(x^2), O(x logx), O(1)} = O(x^2).