

**Metodología de la Programación y Algoritmia**  
Convocatoria de Septiembre 2015  
SOLUCIÓN

**Apellidos** \_\_\_\_\_

**Nombre** \_\_\_\_\_ **DNI** \_\_\_\_\_

- 1.- Menú ideal (igual número de calorías) o en caso de no encontrarlo menú más cercano por debajo al número de calorías.  
1.a) Descripción del funcionamiento de un algoritmo que utilice programación dinámica.  
1.b) Traza para  $M=16$ ,  $n=4$ ,  $P=\{5, 6, 3, 6\}$  y  $V=\{5, 6, 3, 6\}$ .

**SOLUCIÓN:**

1.a) Este ejercicio se puede resolver utilizando el algoritmo que utiliza la estrategia de programación dinámica que se realizó en los ejercicios del Tema 4, considerando que los pesos y valores son las calorías de cada plato. Este algoritmo también se puede encontrar en la bibliografía web recomendada.

- Guerequeta R., Vallecillo A. (2000), Análisis y diseño de algoritmos. Servicio de publicaciones de la Universidad de Málaga (2000). (Capítulo 7)

1.b) Considerando  $M=16$  calorías,  $n=4$  platos y  $P=V=\{5, 6, 3, 6\}$  las calorías de cada plato, los valores que va generando el algoritmo y que se almacenan en la matriz T, que se va rellenando por filas es la siguiente:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 (5)	0	0	0	0	0	5	5	5	5	5	5	5	5	5	5	5	5
2 (6)	0	0	0	0	0	5	6	6	6	6	6	11	11	11	11	11	11
3 (3)	0	0	0	3	3	5	6	6	8	9	9	11	11	11	14	14	14
4 (6)	0	0	0	3	3	5	6	6	8	9	9	11	12	12	14	15	15

El menú final con el mayor número de calorías sin sobrepasar la cantidad máxima de calorías  $M=16$  está calculado en la última celda de la matriz y su valor es de 15 calorías

2.- Diseña un algoritmo recursivo en pseudocódigo llamado suma\_recursivo que calcule la suma de los dígitos de un número entero positivo. Esta función recibe como argumento el número entero positivo y devuelve la suma. Por ejemplo, si el número es 2408 el resultado es 14. Indica qué tipo de recursividad tiene el algoritmo y por qué. Realiza una traza para el número 2408.

**SOLUCIÓN:**

Este ejercicio está planteado en los ejercicios del tema de Recursividad.

## Metodología de la Programación y Algoritmia

Convocatoria de Septiembre 2015  
SOLUCIÓN

3.- Calcula la complejidad asintótica del algoritmo calcular. Justifica la respuesta.  
La complejidad asintótica de obtener(n:entero):entero es  $O(\log n)$ .

```
función calcular(x:entero):entero
  v,i,j,n:entero
  v ← 2
  n ← 10
  para i←1 hasta n
    v ← v + obtener(x)
    j ← 1
    mientras j < x hacer
      v ← v + j
      j ← j * 2
    fmientras
  fpara
  devolver v
ffunción
```

### SOLUCIÓN:

Como se pide la complejidad asintótica se puede analizar el algoritmo por niveles.  
La variable que indica el tamaño del problema es x.

	función calcular(x:entero):entero	<u>Nivel 1</u>	<u>Nivel 2</u>	<u>Nivel 3</u>
	v,i,j,n:entero			
(1)	v ← 2	$O(1)$		
(2)	n ← 10	$O(1)$		
(3)	para i←1 hasta n	$O(1)$		
(4)	v ← v + obtener(x)		$O(\log x)$	
(5)	j ← 1		$O(1)$	
(6)	mientras j < x hacer		$O(\log x)$	
(7)	v ← v + j			$O(1)$
(8)	j ← j * 2			$O(1)$
(9)	fmientras			
(10)	fpara			
(11)	devolver v	$O(1)$		
	ffunción			

Las líneas (7) y (8) siguen la regla de las secuencias de instrucciones, con lo cual cogemos el máximo de ellas que es  $\max\{O(1), O(1)\} = O(1)$ .

	función calcular(x:entero):entero	<u>Nivel 1</u>	<u>Nivel 2</u>	<u>Nivel 3</u>
	v,i,j,n:entero			
(1)	v ← 2	$O(1)$		
(2)	n ← 10	$O(1)$		
(3)	para i←1 hasta n	$O(1)$		
(4)	v ← v + obtener(x)		$O(\log x)$	
(5)	j ← 1		$O(1)$	
(6)	mientras j < x hacer		$O(\log x)$	
(7)	v ← v + j			$O(1)$
(8)	j ← j * 2			
(9)	fmientras			
(10)	fpara			
(11)	devolver v	$O(1)$		
	ffunción			

## Metodología de la Programación y Algoritmia

Convocatoria de Septiembre 2015  
SOLUCIÓN

El bucle “mientras” de la línea (6) finaliza cuando  $j < x$ . Como la variable  $j$  se actualiza multiplicando su valor por una constante, el número de iteraciones está en función del logaritmo y es  $O(\log x)$ . Aplicamos la regla del producto, con lo cual las líneas (6) – (9) tienen una complejidad  $O(\log x) * O(1) = O(\log x)$

	<u>Nivel 1</u>	<u>Nivel 2</u>
función calcular(x:entero):entero		
v,i,j,n:entero		
(1)  v ← 2	$O(1)$	
(2)  n ← 10	$O(1)$	
(3)  para i←1 hasta n	$O(1)$	
(4)    v ← v + obtener(x)		$O(\log x)$
(5)    j ← 1		$O(1)$
(6)    mientras j < x hacer		$O(\log x)$
(7)      v ← v + j		
(8)      j ← j * 2		
(9)    fmientras		
(10)  fpara		
(11)  devolver v	$O(1)$	
ffunción		

Las líneas (4) – (9) siguen la regla de las secuencias de instrucciones, con lo cual cogemos el máximo de todas ellas que es  $\max\{O(\log x), O(1), O(\log x)\} = O(\log x)$ .

	<u>Nivel 1</u>	<u>Nivel 2</u>
función calcular(x:entero):entero		
v,i,j,n:entero		
(1)  v ← 2	$O(1)$	
(2)  n ← 10	$O(1)$	
(3)  para i←1 hasta n	$O(1)$	
(4)    v ← v + obtener(x)		$O(\log x)$
(5)    j ← 1		
(6)    mientras j < x hacer		
(7)      v ← v + j		
(8)      j ← j * 2		
(9)    fmientras		
(10)  fpara		
(11)  devolver v	$O(1)$	
ffunción		

El bucle “para” de la línea (3) se realiza un número fijo de veces  $n$  que no depende del tamaño del problema  $v$ . Las líneas (5) - (10) siguen la regla de los bucles, con lo cual su complejidad es  $O(1) * O(\log x) = O(\log x)$ .

	<u>Nivel 1</u>
función calcular(x:entero):entero	
v,i,j,n:entero	
(1)  v ← 2	$O(1)$
(2)  n ← 10	$O(1)$
(3)  para i←1 hasta n	$O(\log x)$
(4)    v ← v + obtener(x)	
(5)    j ← 1	
(6)    mientras j < x hacer	
(7)      v ← v + j	
(8)      j ← j * 2	
(9)    fmientras	
(10)  fpara	
(11)  devolver v	$O(1)$
ffunción	

Finalmente, volvemos a aplicar la regla de las instrucciones secuenciales al Nivel 1, con lo cual la complejidad asintótica de calcular es  $O(\log x)$ .