

Evaluación Continua

Algoritmos Recursivos

🕒 Fecha de Creación	@5 de noviembre de 2023 14:36
📄 Asignatura	FAL
🕒 Fecha de Modificación	@5 de noviembre de 2023 14:39

Pregunta 1

Correcta
Se puntúa 1,00
sobre 1,00
🚩 Marcar
pregunta

La función de tiempo $T(n)$ de un algoritmo A satisface la siguiente recurrencia:

$$T(n) = 2, n \leq 1$$

$$T(n) = 4n + 12 + T(n/2), n > 1$$

¿Qué orden de complejidad se obtiene resolviendo esta recurrencia mediante los teoremas maestros (patrones genéricos de resolución discutidos en clase)?

Seleccione una o más de una:

- ☐ a. $O(n^2 \log n)$
- ☐ b. $O(n^2)$
- ☒ c. $O(n)$ ✓ **Cierto.** Se ajusta al patrón 'división', con $a = 1$, $k = 1$ (ya que $w(n) = 4n + 12 \in O(n)$) y $b = 2$. Por tanto, como $a = 1 < b^k = 2$, $T(n) \in O(n^k) = O(n)$.
- ☐ d. $O(\sqrt{n})$
- ☐ e. $O(n^3)$

- a. **Falso.** Se obtiene el orden de complejidad $O(n)$
- b. **Falso.** Se obtiene el orden de complejidad $O(n)$
- c. **Cierto.** Se ajusta al patrón 'división', con $a = 1$, $k = 1$ (ya que $w(n) = 4n + 12 \in O(n)$) y $b = 2$. Por tanto, como $a = 1 < b^k = 2$, $T(n) \in O(n^k) = O(n)$.
- d. **Falso.** Se obtiene el orden de complejidad $O(n)$.
- e. **Falso.** Se obtiene el orden de complejidad $O(n)$

La respuesta correcta es: $O(n)$

Pregunta 2

Incorrecta
Se puntúa 0,00
sobre 1,00
🚩 Marcar
pregunta

Considera el siguiente algoritmo recursivo:

$$P \equiv \{a \geq 0\}$$

```
int /* resul*/ incognita(int a, int b) {  
    if (a == 0) return b * b;  
    else return incognita(a - 1, b + 1);  
}
```

$$Q \equiv \{???\}$$

¿Cuál es la post-condición de este algoritmo?:

Seleccione una o más de una:

- ☐ a. $resul = (a + b) * a$
- ☐ b. $resul = b^2$
- ☐ c. $resul = a * b$
- ☐ d. Ninguna de las que se proponen.
- ☒ e. $resul = (a + b) * b$ ✗ **Falso.** Por ejemplo, si $a = 1$ y $b = 1$, $(a + b) * b = 2$. Pero el algoritmo devuelve 4.

- a. **Falso.** Por ejemplo, si $a = 1$ y $b = 1$, $(a + b) * a = 2$. Pero el algoritmo devuelve 4.
- b. **Falso.** Por ejemplo, si $a = 1$ y $b = 1$, $b^2 = 1$. Pero el algoritmo devuelve 4.
- c. **Falso.** Por ejemplo, si $a = 0$ y $b = 1$, $a * b = 0$, pero el algoritmo devuelve 1.
- d. **Cierto.** El algoritmo calcula $(a + b)^2$. Efectivamente, si $a = 0$, $(a + b)^2 = b^2 = b \times b$, valor devuelto por el algoritmo. Si $a > 0$, supongamos que $incognita(a - 1, b + 1) = ((a - 1) + (b + 1))^2$. Pero $((a - 1) + (b + 1))^2 = (a + b)^2$, valor devuelto por el algoritmo.
- e. **Falso.** Por ejemplo, si $a = 1$ y $b = 1$, $(a + b) * b = 2$. Pero el algoritmo devuelve 4.

La respuesta correcta es: Ninguna de las que se proponen.

Pregunta 3

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

La función de tiempo $T(n)$ de un algoritmo A satisface la siguiente recurrencia:

$$T(n) = 1, n \leq 1$$

$$T(n) = 4n^3 + 12 + 8T(n/2), n > 1$$

¿Qué orden de complejidad se obtiene resolviendo esta recurrencia mediante los *teoremas maestros* (patrones genéricos de resolución discutidos en clase)?

Seleccione una o más de una:

- ☐ a. $O(\log n)$
- ☐ b. $O(n^3)$
- ☐ c. $O(\sqrt{n})$
- ☒ d. $O(n^3 \log n)$ **Cierto.** Se ajusta al patrón 'división', con $a = 8$, $k = 3$ (ya que $w(n) = 4n^3 + 12 \in O(n^3)$) y $b = 2$. Por tanto, como $a = b^k = 8$, $T(n) \in O(n^k \log n) = O(n^3 \log n)$.
- ☐ e. $O(n)$

- a. **Falso.** Se obtiene el orden de complejidad $O(n^3 \log n)$
- b. **Falso.** Se obtiene el orden de complejidad $O(n^3 \log n)$
- c. **Falso.** Se obtiene el orden de complejidad $O(n^3 \log n)$.
- d. **Cierto.** Se ajusta al patrón 'división', con $a = 8$, $k = 3$ (ya que $w(n) = 4n^3 + 12 \in O(n^3)$) y $b = 2$. Por tanto, como $a = b^k = 8$, $T(n) \in O(n^k \log n) = O(n^3 \log n)$.
- e. **Falso.** Se obtiene el orden de complejidad $O(n^3 \log n)$

La respuesta correcta es: $O(n^3 \log n)$

Pregunta 4

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

La función de tiempo $T(n)$ de un algoritmo A satisface la siguiente recurrencia:

$$T(n) = 2, n < 2$$

$$T(n) = 4n + 12 + 4T(n-2), n \geq 2$$

¿Qué orden de complejidad se obtiene resolviendo esta recurrencia mediante los *teoremas maestros* (patrones genéricos de resolución discutidos en clase)?

Seleccione una o más de una:

- ☐ a. $O(2^{n/2})$
- ☐ b. $O(n^3)$
- ☐ c. $O(2^{n/4})$
- ☐ d. $O(4^n)$
- ☒ e. $O(4^{n/2})$ **Cierto.** Se ajusta al patrón 'sustracción', con $a = 4$ y $b = 2$. Por tanto, $T(n) \in O(4^{n/2})$

- a. **Falso.** Se obtiene el orden de complejidad $O(4^{n/2})$
- b. **Falso.** Se obtiene el orden de complejidad $O(4^{n/2})$
- c. **Falso.** Se obtiene el orden de complejidad $O(4^{n/2})$
- d. **Falso.** Se obtiene el orden de complejidad $O(4^{n/2})$.
- e. **Cierto.** Se ajusta al patrón 'sustracción', con $a = 4$ y $b = 2$. Por tanto, $T(n) \in O(4^{n/2})$

La respuesta correcta es: $O(4^{n/2})$

Pregunta 5

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Considérese la siguiente función recursiva:

```
int algo(int a) {
    if (a < 2) return a;
    else return algo(a - 1) + algo(a - 2);
}
```

¿Cuáles de las siguientes afirmaciones son ciertas?

Seleccione una o más de una:

- ☒ a. Todas las llamadas recursivas son no finales. **Cierto.** Tras la primera llamada, queda pendiente realizar la segunda llamada, y la suma. Tras la segunda llamada, queda pendiente realizar la suma.
- ☐ b. Todas las llamadas recursivas son finales.
- ☐ c. Tiene recursión simple.
- ☒ d. Tiene recursión múltiple. **Cierto.** El caso recursivo involucra dos llamadas recursivas.
- ☐ e. Una de las llamadas es no final, y la otra es final.

- a. **Cierto.** Tras la primera llamada, queda pendiente realizar la segunda llamada, y la suma. Tras la segunda llamada, queda pendiente realizar la suma.
- b. **Falso.** Ninguna de las llamadas son finales.
- c. **Falso.** El caso recursivo involucra dos llamadas recursivas.
- d. **Cierto.** El caso recursivo involucra dos llamadas recursivas.
- e. **Falso.** Ambas llamadas son no finales.

Las respuestas correctas son: Todas las llamadas recursivas son no finales., Tiene recursión múltiple.

Pregunta 6



Correcta

Se puntúa 1,00 sobre 1,00

 Desmarcar

¿Cuáles de las siguientes afirmaciones son verdaderas?:

Seleccione una o más de una:

- ☐ a. Para los algoritmos recursivos no pueden determinarse órdenes de complejidad exactos de la forma $\Theta(f(n))$, ya que los patrones de resolución de recurrencias únicamente proporcionan órdenes de la forma $O(f(n))$.
- ☒ b. Cualquier bucle puede sustituirse por una recursión.  **Cierto.** De hecho, todo bucle puede expresarse trivialmente mediante una recursión final.
- ☐ c. Utilizando algoritmos recursivos es posible resolver problemas que no pueden resolverse iterativamente.
- ☒ d. Todo algoritmo recursivo puede implementarse sin utilizar recursión.  **Cierto.** De hecho, los compiladores generan automáticamente implementaciones no recursivas de cualquier programa recursivo.
- ☐ e. La resolución iterativa de un problema siempre es más eficiente que la resolución recursiva de dicho problema.

- a. **Falso.** Las recurrencias pueden resolver mediante otros métodos (v.g., directamente con el método de desplegado), y estos otros métodos pueden permitir, a veces, determinar órdenes de complejidad exactos.
- b. **Cierto.** De hecho, todo bucle puede expresarse trivialmente mediante una recursión final.
- c. **Falso.** De hecho, cuando se traduce un programa recursivo a ensamblador, ya no hay recursión (el proceso resultante es un proceso iterativo).
- d. **Cierto.** De hecho, los compiladores generan automáticamente implementaciones no recursivas de cualquier programa recursivo.
- e. **Falso.** Por ejemplo, el método de la burbuja resuelve iterativamente el problema de ordenación, mientras que el *quicksort* lo resuelve recursivamente. El segundo es, en la mayoría de los casos, mucho más eficiente que el primero.

Las respuestas correctas son: Cualquier bucle puede sustituirse por una recursión., Todo algoritmo recursivo puede implementarse sin utilizar recursión.

Pregunta 7

Correcta

Se puntúa 1,00 sobre 1,00

 Marcar pregunta


Considera el siguiente algoritmo recursivo para determinar si un elemento aparece o no en un vector de enteros **ordenado crecientemente**:

```
bool esta_aux(int a[], int v, int i, int j) {
    if ((j - i) + 1 <= 20) {
        bool este=false;
        for (int u=i; u<=j && ! este; u++) {
            if(a[u]==v) este=true;
        }
        return este;
    }
    else {
        int m = (i + j) / 2;
        if (a[m] == v) return true;
        else if(a[m] < v) return esta_aux(a,v,m+1,j);
        else return esta_aux(a,v,i,m-1);
    }
}

bool esta(int a[], int n, int v) {
    return esta_aux(a, v, 0, n - 1);
}
```

¿Cuáles de los siguientes órdenes caracteriza de manera más exacta el coste en el peor caso de este algoritmo?:

Seleccione una o más de una:

- ☐ a. $O(n \times \log n)$
- ☐ b. $O(n + \log n)$
- ☒ c. $O(\log n)$  **Cierto.** El algoritmo es una búsqueda binaria que, para vectores de 20 o menos elementos, realiza una búsqueda secuencial. No obstante, el coste de dichas búsquedas en tramos de 20 elementos o menos no dependerá del tamaño del vector / tramo original, por lo que pueden considerarse operaciones constantes.
- ☐ d. Ninguno de los órdenes propuestos son válidos.
- ☐ e. $O(n)$

- a. **Falso.** El coste del algoritmo es logarítmico, ya que el componente aparentemente lineal opera siempre sobre tramos que no superan los 20 elementos (por tanto, la búsqueda secuencial en dichos tramos suma únicamente un coste constante al coste total del algoritmo).
- b. **Falso.** De hecho, $O(n + \log n) = O(n)$, y es mucho más exacto $O(\log n)$.
- c. **Cierto.** El algoritmo es una búsqueda binaria que, para vectores de 20 o menos elementos, realiza una búsqueda secuencial. No obstante, el coste de dichas búsquedas en tramos de 20 elementos o menos no dependerá del tamaño del vector / tramo original, por lo que pueden considerarse operaciones constantes.
- d. **Falso.** La opción $O(\log n)$ es adecuada.
- e. **Falso.** El coste del algoritmo es logarítmico, ya que el componente aparentemente lineal opera siempre sobre tramos que no superan los 20 elementos (por tanto, la búsqueda secuencial en dichos tramos suma únicamente un coste constante al coste total del algoritmo)

La respuesta correcta es: $O(\log n)$

Pregunta 8

Correcta

Se puntúa 1,00 sobre 1,00

🚩 Marcar pregunta

Considera el siguiente algoritmo recursivo:

$P = \{b \geq 0\}$

```
int /*resul*/ incognita(int a, int b) {
    if (b == 0) return a;
    else return incognita(a, b-1)-1;
}
```

$Q = \{???\}$

¿Cuál es la post-condición de este algoritmo?:

Seleccione una o más de una:

- ☐ a. Ninguna de las que se proponen.
- ☐ b. $resul = \lfloor \log_b(a) \rfloor$, con $\lfloor x \rfloor$ la parte entera de x .
- ☐ c. $resul = \lfloor \log_a(b) \rfloor$, con $\lfloor x \rfloor$ la parte entera de x .
- ☐ d. $resul = a/b$
- ☒ e. $resul = a - b$ **Cierto.** Si $b = 0$, $a - b = a$, valor devuelto por el algoritmo. Supongamos $a > 0$, y supongamos $incognita(a, b - 1) = a - (b - 1)$. Entonces, el algoritmo devuelve $a - (b - 1) - 1 = a - b$.

- a. **Falso.** El algoritmo computa $a - b$.
- b. **Falso.** Por ejemplo, si $a = 8$ y $b = 2$, $\lfloor \log_b(a) \rfloor = 3$. Pero el algoritmo devuelve 6.
- c. **Falso.** Por ejemplo, si $a = 0$, $\lfloor \log_a(b) \rfloor$ no está definido.
- d. **Falso.** Si $a = 0$ y $b = 1$, $a/b = 0$. Pero el algoritmo devuelve -1 .
- e. **Cierto.** Si $b = 0$, $a - b = a$, valor devuelto por el algoritmo. Supongamos $a > 0$, y supongamos $incognita(a, b - 1) = a - (b - 1)$. Entonces, el algoritmo devuelve $a - (b - 1) - 1 = a - b$.

La respuesta correcta es: $resul = a - b$

Pregunta 9

Incorrecta

Se puntúa 0,00 sobre 1,00

🚩 Desmarcar

Considera el siguiente algoritmo recursivo para calcular a^n , con $n > 0$:

```
int pot(int a, int n) {
    if (n == 0) return 1;
    else if (n % 2 == 0) {
        int r_medios = pot(a, n / 2);
        return r_medios * r_medios;
    }
    else {
        return a * pot(a, n - 1);
    }
}
```

¿Cuál de las siguientes recurrencias caracterizan la función de tiempo de este algoritmo?:

Seleccione una o más de una:

- ☐ a. $T(0) = c_0, T(n) = c_1 + \frac{T(n-1) + T(n/2)}{2}, (n \geq 1)$, con c_0 y c_1 constantes.
- ☐ b. $T(0) = c_0, T(n) = c_1 + T(n/2), (n \geq 1)$, con c_0 y c_1 constantes.
- ☐ c. $T(0) = c_0, T(n) = c_1 + T(n-2), (n \geq 1)$, con c_0 y c_1 constantes.
- ☐ d. $T(0) = c_0, T(n) = c_1 + T(n-1), (n \geq 1)$, con c_0 y c_1 constantes.
- ☒ e. Ninguna de las propuestas. **Falso.** La función queda correctamente caracterizada por la recurrencia $T(0) = c_0, T(n) = c_1 + T(n/2), (n \geq 1)$.

- a. **Falso.** $\frac{T(n-1) + T(n/2)}{2}$ es la media del coste de las llamadas recursivas en los dos casos recursivos, pero no caracteriza el coste en cada caso.
- b. **Cierto.** Si n es par, las operaciones que rodean a la llamada recursiva tienen un coste constante, y se realiza una única llamada recursiva, reduciendo el problema a la mitad del tamaño. Si n es impar, tras una llamada recursiva se llega al caso en el que n es par. En este caso, el coste antes de la llamada recursiva que reduce el tamaño del problema a la mitad es, igualmente, constante. c_1 puede representar el máximo del componente constante del coste en ambos casos.
- c. **Falso.** El tamaño del problema no se reduce siempre en dos unidades, sino que, bien se reduce directamente a la mitad, bien se reduce en 1, e inmediatamente después, se reduce a la mitad.
- d. **Falso.** El tamaño del problema no se reduce siempre en una unidad, sino que, bien se reduce directamente a la mitad, bien se reduce en 1, e inmediatamente después, se reduce a la mitad.
- e. **Falso.** La función queda correctamente caracterizada por la recurrencia $T(0) = c_0, T(n) = c_1 + T(n/2), (n \geq 1)$.

La respuesta correcta es: $T(0) = c_0, T(n) = c_1 + T(n/2), (n \geq 1)$, con c_0 y c_1 constantes.

Pregunta 10

Correcta

Se puntúa 1,00 sobre 1,00

🚩 Marcar pregunta

La función de tiempo $T(n)$ de un algoritmo A satisface la siguiente recurrencia:

$T(n) = 15, n \leq 1$

$T(n) = 5 + 3T(n/27), n > 1$

¿Qué orden de complejidad se obtiene resolviendo esta recurrencia mediante los *teoremas maestros* (patrones genéricos de resolución discutidos en clase)?

Seleccione una o más de una:

- ☐ a. $O(n^2)$
- ☐ b. $O(n^2 \log n)$
- ☐ c. $O(n)$
- ☒ d. $O(n^{1/3})$ **Cierto.** Se ajusta al patrón 'división', con $a = 3, k = 0$ y $b = 3$. Por tanto, como $a = 3 > b^0 = 1, T(n) \in O(n^{\log_3 a}) = O(n^{\log_3 7(3)}) = O(n^{1/3})$.
- ☐ e. $O(n \log n)$

- a. **Falso.** Se obtiene el orden de complejidad $O(n^{1/3})$
- b. **Falso.** Se obtiene el orden de complejidad $O(n^{1/3})$
- c. **Falso.** Se obtiene el orden de complejidad $O(n^{1/3})$
- d. **Cierto.** Se ajusta al patrón 'división', con $a = 3, k = 0$ y $b = 3$. Por tanto, como $a = 3 > b^0 = 1, T(n) \in O(n^{\log_3 a}) = O(n^{\log_3 7(3)}) = O(n^{1/3})$.
- e. **Falso.** Se obtiene el orden de complejidad $O(n^{1/3})$

La respuesta correcta es: $O(n^{1/3})$

