Cuestiones Algoritmos Recursivos

Fecha de Creación	@1 de noviembre de 2023 13:23
Asignatura	FAL
Fecha de Modificación	@12 de noviembre de 2023 15:16

Pregunta 1
Correcta
Se puntúa 1,00 sobre 1,00
Desmarcar

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

$T(n) = 15, n < 4$

$T(n) = 5n^2 + 6n + 1 + 2T(n/2), n \geq 4$

¿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:

- [x] a. $O(n^2)$ ✔ **Cierto**. Se ajusta al patrón 'división', con $a = 2, k = 2$ (ya que $w(n) = 5n^2 + 6n + 1 \in O(n^2)$) y $b = 2$. Por tanto, como $a = 2 < b^k = 4, T(n) \in O(n^k) = O(n^2)$.
- [ ] b. $O(n^3)$
- [ ] c. $O(\sqrt{n})$
- [ ] d. $O(n)$
- [ ] e. $O(n^2\ logn)$

a. **Cierto**. Se ajusta al patrón 'división', con $a = 2, k = 2$ (ya que $w(n) = 5n^2 + 6n + 1 \in O(n^2)$) y $b = 2$. Por tanto, como $a = 2 < b^k = 4, T(n) \in O(n^k) = O(n^2)$.
b. **Falso**. Se obtiene el orden de complejidad $O(n^2)$
c. **Falso**. Se obtiene el orden de complejidad $O(n^2)$.
d. **Falso**. Se obtiene el orden de complejidad $O(n^2)$
e. **Falso**. Se obtiene el orden de complejidad $O(n^2)$

La respuesta correcta es: $O(n^2)$

Pregunta 2
Incorrecta
Se puntúa 0,00 sobre 1,00
Marcar pregunta

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

$T(n) = 15, n < 4$

$T(n) = 5 + 2T(n/4), n \geq 4$

¿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. $O(n)$
- [ ] b. $O(n\ log\ n)$
- [x] c. $O(n^2)$ ✖ **Falso**. Se obtiene el orden de complejidad $O(\sqrt{n})$
- [ ] d. $O(\sqrt{n})$
- [ ] e. $O(n^2\ log\ n)$

a. **Falso**. Se obtiene el orden de complejidad $O(\sqrt{n})$
b. **Falso**. Se obtiene el orden de complejidad $O(\sqrt{n})$
c. **Falso**. Se obtiene el orden de complejidad $O(\sqrt{n})$
d. **Cierto**. Se ajusta al patrón 'división', con $a = 2, k = 0$ y $b = 4$. Por tanto, como $a = 2 > b^0 = 1, T(n) \in O(n^{log_b a}) = O(n^{log_4(2)}) = O(n^{1/2}) = O(\sqrt{n})$.
e. **Falso**. Se obtiene el orden de complejidad $O(\sqrt{n})$

La respuesta correcta es: $O(\sqrt{n})$

Pregunta 3
Incorrecta
Se puntúa 0,00 sobre 1,00
Marcar pregunta

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

$T(n) = 15, n < 4$

$T(n) = 5n^2 + 6n + 1 + 2T(n-4), n \geq 4$

¿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. $O(4^{n/2})$
- b. $O(n^3)$ ✖ **Falso**. Se obtiene el orden de complejidad $O(2^{n/4})$
- c. $O(2^{n/2})$
- d. $O(2^{n/4})$
- e. $O(4^n)$

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

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

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:

$T(n) = 15, n < 4$

$T(n) = 5n^2 + 6n + 1 + 4T(n/2), n \geq 4$

¿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. $O(\sqrt{n})$
- b. $O(n^2\ log\ n)$ ✔ **Cierto**. Se ajusta al patrón 'división', con $a = 4$, $k = 2$ (ya que $w(n) = 5n^2 + 6n + 1 \in O(n^2)$) y $b = 2$. Por tanto, como $a = b^k = 4$, $T(n) \in O(n^k\ log\ n) = O(n^2\ log\ n)$.
- c. $O(n)$
- d. $O(n^2)$
- e. $O(n\ log\ n)$

a. **Falso**. Se obtiene el orden de complejidad $O(n^2\ log\ n)$.
b. **Cierto**. Se ajusta al patrón 'división', con $a = 4$, $k = 2$ (ya que $w(n) = 5n^2 + 6n + 1 \in O(n^2)$) y $b = 2$. Por tanto, como $a = b^k = 4$, $T(n) \in O(n^k\ log\ n) = O(n^2\ log\ n)$.
c. **Falso**. Se obtiene el orden de complejidad $O(n^2\ log\ n)$
d. **Falso**. Se obtiene el orden de complejidad $O(n^2\ log\ n)$
e. **Falso**. Se obtiene el orden de complejidad $O(n^2\ log\ n)$

La respuesta correcta es: $O(n^2\ log\ n)$

Pregunta 5
Correcta
Se puntúa 1,00 sobre 1,00
Marcar pregunta

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

$T(n) = 15, n < 4$

$T(n) = 5n^2 + 6n + 1 + T(n-4), n \geq 4$

¿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. $O(n^4)$
- b. $O(n^5)$
- c. $O(n^2)$
- d. $O(n^3)$ ✔ **Cierto**. Se ajusta al patrón 'sustracción', con $a = 1$ y $k = 3$ (ya que $w(n) = 5n^2 + 6n + 1 \in O(n^2)$). Por tanto, $T(n) \in O(n^3)$
- e. $O(n)$

a. **Falso**. Se obtiene el orden de complejidad $O(n^3)$
b. **Falso**. Se obtiene el orden de complejidad $O(n^3)$.
c. **Falso**. Se obtiene el orden de complejidad $O(n^3)$
d. **Cierto**. Se ajusta al patrón 'sustracción', con $a = 1$ y $k = 3$ (ya que $w(n) = 5n^2 + 6n + 1 \in O(n^2)$). Por tanto, $T(n) \in O(n^3)$
e. **Falso**. Se obtiene el orden de complejidad $O(n^3)$

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

Pregunta 6

Correcta

Se puntúa 1,00 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) + 2 * a + 2*b - 1;
}
```

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

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

Seleccione una o más de una:

- a. $resul = ((a+1)+b)^2$
- b. $resul = a \times \frac{a+b}{2}$
- c. $resul = (a+b)^2$ ✔ **Cierto**. Si $a = 0$, $(a+b)^2 = b^2$, que es el valor devuelto por el algoritmo en este caso. Si $a > 0$, supongamos que $incognita(a-1, b) = ((a-1)+b)^2$. Entonces, como $((a-1)+b)^2 = (a-1)^2 + b^2 + 2(a-1)b = a^2 + 1 - 2a + b^2 + 2ab - 2b$, el algoritmo devuelve $a^2 + 1 - 2a + b^2 + 2ab - 2b + 2a + 2b - 1 = a^2 + b^2 + 2ab = (a+b)^2$.
- d. $resul = (a-b)^2$
- e. Ninguna de las que se proponen.

a. **Falso**. Por ejemplo, si $a = 0$ y $b = 2$, $((a+1)+b)^2 = 9$. Pero, en este caso, el algoritmo devuelve 4.
b. **Falso**. Por ejemplo, si $a = 0$ y $b = 1$, $a \times \frac{a+b}{2} = 0$. Sin embargo, en este caso el algoritmo devuelve 1.
c. **Cierto**. Si $a = 0$, $(a+b)^2 = b^2$, que es el valor devuelto por el algoritmo en este caso. Si $a > 0$, supongamos que $incognita(a-1, b) = ((a-1)+b)^2$. Entonces, como $((a-1)+b)^2 = (a-1)^2 + b^2 + 2(a-1)b = a^2 + 1 - 2a + b^2 + 2ab - 2b$, el algoritmo devuelve $a^2 + 1 - 2a + b^2 + 2ab - 2b + 2a + 2b - 1 = a^2 + b^2 + 2ab = (a+b)^2$.
d. **Falso**. Por ejemplo, si $a = 1$ y $b = 2$, $(a-b)^2 = (-1)^2 = 1$. Pero, en este caso, el algoritmo devuelve 9.
e. **Falso**. El algoritmo calcula $(a+b)^2$.

La respuesta correcta es: $resul = (a+b)^2$

Pregunta 7

Correcta

Se puntúa 1,00 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 == 1) return a;
    else if (n % 2 == 0) {
        int r_medios = pot(a, n / 2);
        return r_medios * r_medios;
    }
    else {
        return a * a * pot(a, n - 2);
    }
}
```

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

Seleccione una o más de una:

- a. $T(0) = c_0, T(1) = c_1, T(n) = c_2 + \frac{T(n-2)+T(n/2)}{2}, (n > 1)$, con c_0, c_1 y c_2 constantes.
- b. $T(0) = c_0, T(1) = c_1, T(n) = c_2 + T(n-2), (n > 1)$, con c_0, c_1 y c_2 constantes.
- c. $T(0) = c_0, T(1) = c_1, T(n) = c_2 + T(n/2), (n > 1)$, con c_0, c_1 y c_2 constantes.
- d. Ninguna de las propuestas. ✔ **Cierto**. El coste en el mejor caso se da cuando n es potencia de 2, y en el peor caso cuando n es impar. En ambos supuestos las recurrencias son distintas. Por tanto, la recurrencia dependerá del tipo de caso que se está considerando.
- e. $T(0) = c_0, T(1) = c_1, T(n) = c_2 + T(n-1), (n > 1)$, con c_0, c_1 y c_2 constantes.

a. **Falso**. $\frac{T(n-2)+T(n/2)}{2}$ es la media del coste de las llamadas recursivas en los dos casos recursivos, pero no caracteriza el coste en cada caso.
b. **Falso**. La recurrencia sirve cuando n es impar.
c. **Falso**. La recurrencia sirve cuando n es potencia de 2.
d. **Cierto**. El coste en el mejor caso se da cuando n es potencia de 2, y en el peor caso cuando n es impar. En ambos supuestos las recurrencias son distintas. Por tanto, la recurrencia dependerá del tipo de caso que se está considerando.
e. **Falso**. El tamaño del subproblema, bien se reduce a la mitad, bien en dos unidades, pero nunca en una unidad.

La respuesta correcta es: Ninguna de las propuestas.

Pregunta 8
Correcta
Se puntúa 1,00 sobre 1,00
Marcar pregunta

Considera el siguiente algoritmo recursivo:

$P \equiv \{a \geq 0 \wedge b > 0\}$

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

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

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

Seleccione una o más de una:

- a. $resul = \lfloor log_a(b) \rfloor$, con $\lfloor x \rfloor$ la parte entera de x.
- b. $resul = \lfloor log_b(a) \rfloor$, con $\lfloor x \rfloor$ la parte entera de x.
- c. Ninguna de las que se proponen.
- d. $resul = a/b$ ✔ **Cierto**. Si $a < b, a/b = 0$, que es el resultado devuelto por el algoritmo. Supongamos que $incognita(a-b,b) = \frac{a-b}{b}$. Entonces, el resultado devuelto por el algoritmo es $\frac{a-b}{b} + 1 = \frac{a-b+b}{b} = a/b$.
- e. $resul = a - b$

a. **Falso**. Por ejemplo, si $a = 0$, $\lfloor log_a(b) \rfloor$ no está definido.
b. **Falso**. Por ejemplo, si $a = 8$ y $b = 2$, $\lfloor log_b(a) \rfloor = 3$. Pero el algoritmo devuelve 4.
c. **Falso**. El algoritmo computa a/b.
d. **Cierto**. Si $a < b, a/b = 0$, que es el resultado devuelto por el algoritmo. Supongamos que $incognita(a-b,b) = \frac{a-b}{b}$. Entonces, el resultado devuelto por el algoritmo es $\frac{a-b}{b} + 1 = \frac{a-b+b}{b} = a/b$.
e. **Falso**. Por ejemplo, si $a = 0$ y $b = 1$, $a - b = -1$. Pero el algoritmo devuelve 0.

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

Pregunta 9
Parcialmente correcta
Se puntúa 0,33 sobre 1,00
Marcar pregunta

¿Cuáles de las siguientes afirmaciones son verdaderas?:

Seleccione una o más de una:

- a. Si un algoritmo se implementa mediante varios subprogramas mutuamente recursivos, entonces la recursión es necesariamente no final.
- b. Para implementar un algoritmo recursivo es necesario un lenguaje de programación que soporte recursión. ✖ **Falso**. De hecho, los compiladores para lenguajes que soportan recursión, tales como C++, traducen automáticamente los programas a código ensamblador, que no soportan recursión. De esta forma, todo algoritmo recursivo puede implementarse no recursivamente con ayuda de una pila en la que puedan alojarse los distintos registros de activación.
- c. Si en un algoritmo recursivo aparecen varias llamadas recursivas, entonces la recursión es múltiple. ✖ **Falso**. Cada llamada puede estar en un caso recursivo distinto.
- d. En la práctica, únicamente pueden encadenarse un número finito de llamadas recursivas no finales. ✔ **Cierto**. Si la recursión es no final, será necesario mantener un registro de activación por cada llamada, lo que supone un coste en memoria. Por tanto, si se encadena un número suficientemente grande de llamadas recursivas, se acabará por llenar la pila de registros de activación.
- e. Si un caso recursivo tiene recursión múltiple, entonces en dicho caso hay una llamada recursiva no final. ✔ **Cierto**. Si existe recursión múltiple, al menos deberá producirse primero una llamada recursiva, y una vez devuelto el control de dicha llamada, otra nueva llamada recursiva. Por tanto, la primera llamada recursiva es necesariamente no final, ya que, cuando se completa, aún quedan acciones por realizar.

a. **Falso**. Piénsese, por ejemplo, el caso en el que todas las llamadas a subrpogramas están en posiciones finales (es decir, una vez completadas, no quedan más acciones por realizar).
b. **Falso**. De hecho, los compiladores para lenguajes que soportan recursión, tales como C++, traducen automáticamente los programas a código ensamblador, que no soportan recursión. De esta forma, todo algoritmo recursivo puede implementarse no recursivamente con ayuda de una pila en la que puedan alojarse los distintos registros de activación.
c. **Falso**. Cada llamada puede estar en un caso recursivo distinto.
d. **Cierto**. Si la recursión es no final, será necesario mantener un registro de activación por cada llamada, lo que supone un coste en memoria. Por tanto, si se encadena un número suficientemente grande de llamadas recursivas, se acabará por llenar la pila de registros de activación.
e. **Cierto**. Si existe recursión múltiple, al menos deberá producirse primero una llamada recursiva, y una vez devuelto el control de dicha llamada, otra nueva llamada recursiva. Por tanto, la primera llamada recursiva es necesariamente no final, ya que, cuando se completa, aún quedan acciones por realizar.

Las respuestas correctas son: En la práctica, únicamente pueden encadenarse un número finito de llamadas recursivas no finales., Si un caso recursivo tiene recursión múltiple, entonces en dicho caso hay una llamada recursiva no final.

Pregunta 10
Correcta
Se puntúa 1,00 sobre 1,00
Desmarcar

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

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

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

¿Cuál 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. $O(1)$
- b. $O(log\, n)$
- c. Ninguno de los órdenes propuestos son válidos.
- d. $O(n\, log\, n)$
- e. $O(n)$ ✔ **Cierto**. El coste de `esta_aux` puede determinarse a partir de la recurrencia $T(0) = c_0$, $T(u) = c_1 + 2T(u/2)$, donde $u = (j - i) + 1$. Por tanto, $T(u) \in O(u)$. Dado que, en la llamada inicial, $u = n$, el coste del algoritmo es $O(n)$.

a. **Falso**. Planteando y resolviendo la correspondiente recurrencia, puede verse que el coste es $O(n)$. Sin embargo, al ser un orden no exacto, nos queda la duda de si el coste puede ser de un orden menor. No obstante, en el peor caso (el valor v no está en el vector), el algoritmo visita cada elemento exactamente una vez. En cada visita realiza un trabajo constante. Por tanto, la complejidad es $\Theta(n)$, por lo que no es $O(1)$.
b. **Falso**. Planteando y resolviendo la correspondiente recurrencia, puede verse que el coste es $O(n)$. Sin embargo, al ser un orden no exacto, nos queda la duda de si el coste puede ser de un orden menor. No obstante, en el peor caso (el valor v no está en el vector), el algoritmo visita cada elemento exactamente una vez. En cada visita realiza un trabajo constante. Por tanto, la complejidad es $\Theta(n)$, por lo que no es $O(log\ n)$.
c. **Falso**. El coste del algoritmo es $O(n)$.
d. **Falso**. El coste del algoritmo es $O(n)$, más preciso que $O(n\ log\ n)$.
e. **Cierto**. El coste de `esta_aux` puede determinarse a partir de la recurrencia $T(0) = c_0$, $T(u) = c_1 + 2T(u/2)$, donde $u = (j - i) + 1$. Por tanto, $T(u) \in O(u)$. Dado que, en la llamada inicial, $u = n$, el coste del algoritmo es $O(n)$.

La respuesta correcta es: $O(n)$