

Comenzado el	viernes, 10 de enero de 2025, 13:28
Estado	Finalizado
Finalizado en	viernes, 10 de enero de 2025, 13:32
Tiempo empleado	3 minutos 39 segundos
Calificación	6,00 de 10,00 (60%)

Pregunta 1

Sin contestar

Se puntúa como 0 sobre 1,00

Supuesto $n \geq 0$ y $c > 2$, ¿qué se puede afirmar sobre la terminación del siguiente algoritmo?

```
int f(int n, int c){  
    if ( n > 10 )  
        return c;  
    else  
        return f (n/10, c+n%10);  
}
```

Seleccione una:

- ☐ a. Termina siempre
- ☐ b. No termina únicamente para los valores de n entre 0 y 10
- ☐ c. No termina nunca
- ☐ d. Ninguna de las anteriores.

- a. Falso. Si $0 \leq n \leq 10$ el algoritmo no termina puesto que en esos valores se vuelve a llamar con $n/10$ que de nuevo está entre 0 y 1.
- b. Cierto. El caso base es cuando el parámetro $n > 10$, por lo tanto para esos casos termina siempre. Cuando no termina es si $0 \leq n \leq 10$ puesto que en esos valores se vuelve a llamar con $n/10$ que será un valor entre 0 y 1.
- c. Falso. El caso base es cuando el parámetro $n > 10$, por lo tanto para esos casos termina siempre.
- d. Falso. La respuesta correcta es: No termina únicamente para los valores de n entre 0 y 10

La respuesta correcta es: No termina únicamente para los valores de n entre 0 y 10

Pregunta 2

Sin contestar

Se puntúa como 0 sobre 1,00

¿Cuál es el coste de la siguiente función recursiva?

```
int f(const vector<int>&v, int c, int a, int b){
    if ( b == a + 1 ) {
        return v[a]*c - 3*c;

    } else{
        int m = (a + b)/2;
        int x = f(v, c*2, a, m);
        int y = f(v, c*2+1, m, b);
        for(int i = a; i <= b; ++i)
            x+= v[a];
        return x - y + c;
    }
}
```

Seleccione una:

- ☐ a. $\theta(k \log n)$ con $n = b - a$.
- ☐ b. $\theta(k * N)$ con $N = v.size()$.
- ☐ c. $\theta(n \log n)$ con $n = b - a$.
- ☐ d. Ninguna de las anteriores.

- a. Falso. El parámetro k no influye en el número de llamadas que se hacen, así que el coste asintótico no depende de dicho parámetro. La recurrencia es $T(n) = c_0$ si $n \leq 1$; $T(n) = 2 * T(n/2) + c_1 * n$ si $n > 1$, así que por el teorema de la división $T(n) \in \theta(n \log n)$.
- b. Falso. El parámetro k no influye en el número de llamadas que se hacen, así que el coste asintótico no depende de dicho parámetro. Además, el tamaño de la entrada en la función f es la longitud del intervalo $[a, b]$, es decir, $n = b - a$, y no la longitud del vector. $T(n) = c_0$ si $n \leq 1$; $T(n) = 2 * T(n/2) + c_1 * n$ si $n > 1$, así que por el teorema de la división $T(n) \in \theta(n \log n)$.
- c. Cierto. n está definida como la longitud del intervalo $[a, b]$. Así la recurrencia es $T(n) = c_0$ si $n \leq 1$; $T(n) = 2 * T(n/2) + c_1 * n$ si $n > 1$, así que por el teorema de la división $T(n) \in \theta(n \log n)$.
- d. Falso. La respuesta correcta es: $\theta(n \log n)$ con $n = b - a$.

La respuesta correcta es: $\theta(n \log n)$ con $n = b - a$.



Pregunta 3

Correcta

Se puntúa 1,00 sobre 1,00

Indica cuales de las siguiente afirmaciones son ciertas con respecto a los algoritmos divide y vencerás:

Seleccione una o más de una:

- ☒ a. Dividen el problema original en subproblemas que se pueden resolver en paralelo  Cierto. Cada subproblema se debe poder resolver de forma independiente antes de combinar las soluciones para obtener la solución al problema original.
- ☐ b. Son más eficientes que los algoritmos no divide y vencerás que resuelven el mismo problema
- ☒ c. Dividen el problema original en subproblemas cuyo tamaño es una fracción del original  Cierto. Cada subproblema debe tener como tamaño una fracción del original.
- ☐ d. Sólo se implementan de forma recursiva

- a. Cierto. Cada subproblema se debe poder resolver de forma independiente antes de combinar las soluciones para obtener la solución al problema original.
- b. Falso. Usar divide y vencerás no garantiza mejorar la complejidad.
- c. Cierto. Cada subproblema debe tener como tamaño una fracción del original.
- d. Falso. Divide y vencerás es una estrategia de resolución de problemas. Por ejemplo, la búsqueda binaria se puede implementar de forma iterativa o recursiva.

Las respuestas correctas son: Dividen el problema original en subproblemas que se pueden resolver en paralelo, Dividen el problema original en subproblemas cuyo tamaño es una fracción del original

Pregunta 4

Correcta

Se puntúa 1,00 sobre 1,00

Indica cuál de las siguientes es una función de cota que permite demostrar la terminación de este bucle, suponiendo que $0 \leq n < \text{long}(v)$:

```
int i = 0, j = n;
while (i < j){
    if (v[i]%10 > v[j]%10)
        j-- ;
    i++;
}
```

Seleccione una:

- ☐ a. i .
- ☐ b. $j - i - 3$.
- ☒ c. $n - i + 1$ ✓ Cierto. La variable j empieza en n y disminuye en algunas ocasiones, mientras que la variable i aumenta desde el valor 0 en cada iteración del bucle. En el caso peor, la j no disminuye nunca por lo que el bucle para cuando $i = j = n$, por lo tanto la expresión $n - i + 1$ es una cota válida
- ☐ d. Ninguna de las anteriores.

- a. Falso. La variable i crece en todas las iteraciones, no puede ser una función de cota.
- b. Falso. Puede suceder que la expresión sea negativa: cuando $j = i + 2$ se tiene que $i < j$ pero la expresión es $i + 2 - i - 3 = -1 < 0$. Por ello no puede ser una función de cota.
- c. Cierto. La variable j empieza en n y disminuye en algunas ocasiones, mientras que la variable i aumenta desde el valor 0 en cada iteración del bucle. En el caso peor, la j no disminuye nunca por lo que el bucle para cuando $i = j = n$, por lo tanto la expresión $n - i + 1$ es una cota válida
- d. Falso. La respuesta correcta es: $n - i + 1$.

La respuesta correcta es: $n - i + 1$.


Pregunta 5

Correcta

Se puntúa 1,00 sobre 1,00

Considerando el algoritmo de ordenación rápida o quicksort para un vector de tamaño mayor que 1, ¿cuál de estas afirmaciones es cierta? `

Seleccione una:

- ☐ a. Sólo se puede implementar de manera recursiva.
- ☒ b. Su coste en el caso peor es $\theta(n^2)$.  Cierto. El peor caso de quicksort es cuando el pivote no divide el espacio de búsqueda en dos mitades sino que queda en un extremo. Por ejemplo si elegimos de pivote el primer elemento y el vector está ordenado ya. En esta circunstancia el coste es $\theta(n^2)$.
- ☐ c. Es un algoritmo de partición.
- ☐ d. Ninguna de las anteriores.

- a. Falso. Incorrecta. Quicksort es un algoritmo divide y vencerás que se puede implementar entero iterativamente como cualquier otro algoritmo Divide y Vencerás.
- b. Cierto. El peor caso de quicksort es cuando el pivote no divide el espacio de búsqueda en dos mitades sino que queda en un extremo. Por ejemplo si elegimos de pivote el primer elemento y el vector está ordenado ya. En esta circunstancia el coste es $\theta(n^2)$.
- c. Falso. Incorrecta. Quicksort usa un algoritmo de partición para repartir el espacio entre los elementos menores y mayores que el pivote, pero eso es sólo una parte del mismo. Quicksort es un algoritmo de ordenación divide y vencerás.
- d. Falso. La respuesta correcta es: Su coste en el caso peor es $\theta(n^2)$.

La respuesta correcta es: Su coste en el caso peor es $\theta(n^2)$.

Pregunta 6

Sin contestar

Se puntúa como 0 sobre 1,00

Dado el siguiente código, $0 < r \leq v.size()$, $0 \leq f \leq v.size()$ y v es un vector de enteros que cumple $\forall i: 0 \leq i < r: v[i] = 0$:

```
int n = 0, i = 0, j = r, a = r;
while (a < f){
    if(v[a]%2 == 0)
        ++j;
    else
        ++i;
    if(v[a-r] == 0)
        --j;
    else
        --i;
    if(i < j)
        ++n;
    ++a;
}
```

¿Cuál de estas propiedades es un invariante de este bucle?

Seleccione una:

- ☐ a. $r < a < f$
- ☐ b. $n = \#p, q: 0 \leq p < q \leq a \wedge q - p = r: P(v, p, q)$, donde $P(v, x, y) = (\#s: x \leq s < y: v[s]\%2 = 0) < (\#s: x \leq s < y: v[s]\%2 \neq 0)$
- ☐ c. $n = \#p, q: 0 \leq p < q \leq a \wedge q - p = r: P(v, p, q)$, donde $P(v, x, y) = (\#s: x \leq s \leq y: v[s]\%2 = 0) > (\#s: x \leq s \leq y: v[s]\%2 \neq 0)$
- ☐ d. Ninguna de las anteriores.

- a. Falso. Incorrecto. El invariante debe ser cierto antes de comenzar el bucle y al terminar el bucle. Al comienzo $a = r$ y al final $a = f$
- b. Falso. Incorrecto. El bucle calcula el número de intervalos de longitud r tales que el número de valores pares es mayor que el número de valores impares en el vector v y ese valor se guarda en la variable n . Por lo tanto, en cada vuelta, en n se guarda el número de intervalos de longitud r desde el inicio del vector hasta el índice del bucle, es decir, la variable a . El problema es que el predicado auxiliar $P(v, x, y)$ indica que en el intervalo $[x, y)$ hay más impares que pares ya que $(\#s: x \leq s < y: v[s]\%2 = 0) < (\#s: x \leq s < y: v[s]\%2 \neq 0)$. Por eso es incorrecto.
- c. Falso. Incorrecto. El bucle calcula el número de intervalos de longitud r tales que el número de valores pares es mayor que el número de valores impares en el vector v y ese valor se guarda en la variable n . Por lo tanto, en cada vuelta, en n se guarda el número de intervalos de longitud r desde el inicio del vector hasta el índice del bucle, es decir, la variable a . Aquí llegamos hasta a , al final del bucle $a = f$ y f podría valer hasta $v.size()$, si es así el invariante está mal definido por culpa del predicado auxiliar $P(v, x, y)$ que indica que en el intervalo $[x, y]$ hay más impares que pares: si llamamos con $q = a = f = v.size()$ accederíamos a posiciones erróneas del vector. Por eso es incorrecto.
- d. Cierto. La respuesta correcta es: $(i = \#k: a - r \leq k < a: v[k]\%2 \neq 0) \wedge (j = \#k: a - r \leq k < a: v[k]\%2 = 0)$

La respuesta correcta es: Ninguna de las anteriores.

Pregunta 7

Correcta

Se puntúa 1,00 sobre 1,00

Indica la complejidad del siguiente algoritmo

```
int f(int n, int m){
    int x = 0;
    for (int i = -10; i <= n - 7; i += 2)
        x -= 2;
    for (int j = 1; j < m; j *= 4)
        x -= 3;

    return x;
}
```

Seleccione una:

- ☐ a. No se puede saber.
- ☐ b. $\theta(n * m)$
- ☒ c. $\theta(n + \log m)$ ✓ Cierto. Hay dos bucles independientes. El primero da un número proporcional a n de vueltas, por lo que su coste es (n). Mientras que el segundo da un número logarítmico de vueltas con respecto a m, por lo que su coste es (m). Como no sabemos si n o m es mayor, el coste es (n + m).
- ☐ d. Ninguna de las anteriores.
- a. Falso. Hay dos bucles independientes. El primero da un número proporcional a n de vueltas, por lo que su coste es (n). Mientras que el segundo da un número logarítmico de vueltas con respecto a m, por lo que su coste es (m). Al ser independientes el coste es la suma (n + m), es decir, el máximo entre estas expresiones. Así que sí que se puede saber.
- b. Falso. Hay dos bucles independientes. El primero da un número proporcional a n de vueltas, por lo que su coste es (n). Mientras que el segundo da un número logarítmico de vueltas con respecto a m, por lo que su coste es (m). Al ser independientes el coste no es el producto sino la suma (n + m), es decir, el máximo entre estas expresiones.
- c. Cierto. Hay dos bucles independientes. El primero da un número proporcional a n de vueltas, por lo que su coste es (n). Mientras que el segundo da un número logarítmico de vueltas con respecto a m, por lo que su coste es (m). Como no sabemos si n o m es mayor, el coste es (n + m).
- d. Falso. La respuesta correcta es $\theta(n + \log m)$.

La respuesta correcta es: $\theta(n + \log m)$


Pregunta 8

Correcta

Se puntúa 1,00 sobre 1,00

Indica cuál de las siguientes afirmaciones es correcta

Seleccione una:

- ☐ a. $\theta(n^2) = \theta(n^3)$
- ☐ b. $\Omega(n) \subseteq \Omega(n^2)$
- ☐ c. $O(\log n) \supseteq O(n^4)$
- ☒ d. Ninguna de las anteriores.  Cierto. La respuesta correcta es: $\Omega(n \log n) \subseteq \Omega(n)$

- a. Falso. Por el teorema del límite, como $\lim_{n \rightarrow \infty} \frac{n^2}{n^3} = 0$, $\theta(n^2) \not\subseteq \theta(n^3)$, por lo que no pueden ser iguales.
- b. Falso. Por el teorema del límite, como $\lim_{n \rightarrow \infty} \frac{n}{n^2} = 0$, $\Omega(n) \not\subseteq \Omega(n^2)$.
- c. Falso. Por el teorema del límite, como $\lim_{n \rightarrow \infty} \frac{n^4}{\log n} = \infty$, por lo que $O(n^4) \not\subseteq O(\log n)$.
- d. Cierto. La respuesta correcta es: $\Omega(n \log n) \subseteq \Omega(n)$

La respuesta correcta es: Ninguna de las anteriores.

Pregunta 9

Correcta

Se puntúa 1,00 sobre 1,00


Lucas ha encontrado una baraja de cartas en un cajón. Hay un total de C cartas repartidas en 4 tipos diferentes (c_i es la cantidad de cartas del tipo $i \in \{1, 2, 3, 4\}$). Con esas cartas quiere construir una torre de altura A . Sólo necesita usar una carta por altura y para que la construcción suponga un reto mayor se autoimpone las siguientes restricciones:

- Tiene que usar los 4 tipos de cartas.
- Debe haber siempre más cartas de tipo 1 que del resto de tipos.
- Al final, en la torre, la suma de cartas del tipo 1 y 3 debe ser mayor o igual que la suma de cartas del tipo 2 y 4.

Lucas quiere saber cuántas combinaciones puede obtener antes de ponerse a construir la torre. Para ello usa la técnica de vuelta atrás sabiendo que va a asignar una carta (de las posibles) por cada piso. Su problema es que no tiene ni idea de qué debe usar como marcadores.

¿Cuál de las siguientes opciones es el marcador necesario para resolver este problema?

Seleccione una:

- ☐ a. Este problema no se puede resolver usando marcadores.
- ☐ b. Una variable entera para indicar el total de cartas usadas.
- ☒ c. Un vector de enteros de tamaño  Cierto. Es suficiente saber cuántas cartas de cada tipo hemos usado para comprobar si se han usado cartas de todos los tipos, si hay en cualquier momento más cartas de tipo 1 y si la suma de los tipos 1 y 3 supera o iguala a la de los tipos 2 y 4.
- ☐ d. Ninguna de las anteriores.

- a. Falso. Incorrecta. Un vector de enteros de tamaño T que indique por cada tipo de cartas cuántas se han usado es suficiente para chequear las 3 condiciones que debe cumplir la torre.
- b. Falso. Incorrecto. El total de cartas usadas está dada por el parámetro $k + 1$ y al final por A , la altura de la torre.
- c. Cierto. Es suficiente saber cuántas cartas de cada tipo hemos usado para comprobar si se han usado cartas de todos los tipos, si hay en cualquier momento más cartas de tipo 1 y si la suma de los tipos 1 y 3 supera o iguala a la de los tipos 2 y 4.
- d. Falso. La respuesta correcta es: Un vector de enteros de tamaño 4 que indique por cada tipo de cartas cuántas se han usado.

La respuesta correcta es: Un vector de enteros de tamaño 4 que indique por cada tipo de cartas cuántas se han usado.

Pregunta 10

Sin contestar

Se puntúa como 0 sobre 1,00

Tenemos la siguiente función con su especificación:

$$P: \{0 \leq k \leq n < v.size() \}$$

```
int f(const vector<int>& v, const int k, const int n)
```

$$Q: \{ \#a, b: 0 \leq a < b \leq n \wedge b - a + 1 = k : S(v, a, b) \}$$

donde, $S(v, a, b) = \forall j: a \leq j < b: v[j] \leq v[j+1]$.

¿Cuál de las siguientes combinaciones de parámetros de entrada y salida satisfacen esta especificación?

Seleccione una:

- ☐ a. Llamada $f([-1, 2, 4, 4, 3, 9, 2], 3, 6)$ con resultado 3
- ☐ b. Llamada $f([1, 1, 9, 0, 3, 4, 4], 2, 5)$ con resultado 4
- ☐ c. Llamada $f([1, 1, 9, 0, 3, 4, 4], 0, 3)$ con resultado 1
- ☐ d. Ninguna de las anteriores.

- a. Falso. Para que la entrada cumpla la precondition los valores k y n deben cumplir que son mayores o iguales que 0 y ser menores que $v.size()$. En este caso se cumple $0 \leq 3 \leq 6 < 7$. Por otro lado, la postcondición indica que el resultado es el número de intervalos $[a, b]$ tales que $6 = n \geq b > a \geq 0$ de longitud $k = 3$ ($b - a + 1 = 3$). Sólo hay 2 intervalos así: $[0, 2]$ y $[1, 3]$ y no 3. Es resultado no cumple la postcondición.
- b. Cierto. Para que la entrada cumpla la precondition los valores k y n deben cumplir que son mayores o iguales que 0 y ser menores que $v.size()$. En este caso se cumple $0 \leq 2 \leq 5 < 7$. Por otro lado, la postcondición indica que el resultado es el número de intervalos $[a, b]$ con $5 = n \geq b > a \geq 0$ tales que su longitud es k ($b - a + 1 = k$) y en los que los valores del vector son crecientes. Para el vector $[1, 1, 9, 0, 3, 4, 4]$ hay 4 intervalos así. Por ello se cumple también la postcondición
- c. Falso. Para que la entrada cumpla la precondition los valores k y n deben cumplir que son mayores o iguales que 0 y ser menores que $v.size()$. En este caso se cumple $0 \leq 0 \leq 3 < 7$. Por otro lado, la postcondición indica que el resultado es el número de intervalos $[a, b]$ tales que $3 = n \geq b > a \geq 0$ de longitud $k = 0$ ($b - a + 1 = 0$). No hay valores posibles para a y b que cumplan estas condiciones por lo que la cantidad de intervalos es 0 y no 1. Es resultado no cumple la postcondición.
- d. Falso. La respuesta correcta es: Llamada $f([1, 1, 9, 0, 3, 4, 4], 2, 5)$ con resultado 4

La respuesta correcta es: Llamada $f([1, 1, 9, 0, 3, 4, 4], 2, 5)$ con resultado 4