

Comenzado el	jueves, 19 de junio de 2025, 16:28
Estado	Finalizado
Finalizado en	jueves, 19 de junio de 2025, 16:32
Tiempo empleado	4 minutos 4 segundos
Calificación	4,00 de 10,00 (40%)

Pregunta 1

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

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

Seleccione una:

- ☐ a. $r < a < f$
- ☐ b. $n = \#p, q: 0 \leq p < q < a \wedge q - p = r: P(v, p, q)$, 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. $(i = \#k: a - r \leq k \leq a: v[k]\%2 = 0) \wedge (j = \#k: a - r \leq k \leq a: v[k]\%2 \neq 0)$
- ☐ 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 impares es mayor que el número de valores pares 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 la variable q como máximo tiene el valor $a - 1$ y al final del bucle $a = f$, por lo que el invariante cuenta mal, no considera el último intervalo. Esto es así 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 $q = a - 1 = f - 1$ nunca comprobaríamos el intervalo $[f-r, f)$. Por eso es incorrecto.
- c. Falso. Incorrecto. El bucle calcula el número de intervalos de longitud r tales que el número de valores impares es mayor que el número de valores pares en el vector v y ese valor se guarda en la variable n . Para llevar la cuenta del número de valores pares del intervalo usamos la variable i , mientras que j cuenta el número de variables impares. El intervalo corresponde a los índices $[a - r, a)$ y no $[a - r, a]$ como indican las expresiones ya que al terminar el bucle $a = f$ y f puede ser $v.size()$, en ese caso, a posiciones erróneas del vector. Por eso es incorrecto.
- d. Cierto. La respuesta correcta es: $n = \#p, q: 0 \leq p < q \leq a \wedge q - p = r: P(v, p, q)$, donde $P(v, x, y) = (\#s: x \leq s < y: v[s]\%2 = 0) < (\#s: x \leq s < y: v[s]\%2 \neq 0)$

La respuesta correcta es: Ninguna de las anteriores.

Pregunta 2

Sin contestar

Se puntúa como 0 sobre 1,00

Tenemos la siguiente función con su especificación:

$$P: \{0 \leq n < v.size() \wedge -10 < k < 10\}$$

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

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

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

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

Seleccione una:

- ☐ a. Llamada $f([-1, -2, 0, 0, -6, -10], 0, 2)$ con resultado 3
- ☐ b. Llamada $f([2, 3, 5, 0, 1, -4, -1], 4, 7)$ con resultado 4
- ☐ c. Llamada $f([2, 3, 5, 0, 1, 4, -1], 5, 5)$ con resultado 3
- ☐ d. Ninguna de las anteriores.

- a. Falso. Para que la entrada cumpla la precondition, n deben cumplir que esté en el rango $[0, v.size())$, mientras que k debe estar en el rango $(-10, 10)$. En este caso se cumple tanto que $0 \leq n = 2 < 6$ como $0 \in (-10, 10)$. Por otro lado, la postcondición indica que el resultado es la longitud (dada por la expresión $b - a + 1$) máxima de los intervalos $[a, b]$ (con $a < b$) tales que se cumple también el predicado $S(v, a, b, k)$, que indica que todos los elementos del intervalo deben ser menores estrictos que k. En este caso en el vector $[-1, -2, 0, 0, -6, -10]$ entre las posiciones 0 y $n = 2$ no hay un único intervalo que cumple que todos sus elementos son menores que 0: el que va desde la posición $a = 0$ hasta $b = 1$, de longitud 2, y no 3 como se indica. Por lo que no se cumple la especificación.
- b. Falso. Para que la entrada cumpla la precondition, n deben cumplir que esté en el rango $[0, v.size())$, mientras que k debe estar en el rango $(-10, 10)$. En este caso si bien $4 \in (-10, 10)$, no se cumple que $0 \leq n = 7 < 7$. La llamada no satisface la precondition.
- c. Cierto. Para que la entrada cumpla la precondition, n deben cumplir que esté en el rango $[0, v.size())$, mientras que k debe estar en el rango $(-10, 10)$. En este caso se cumple tanto que $0 \leq n = 5 < 7$ como $5 \in (-10, 10)$. Por otro lado, la postcondición indica que el resultado es la longitud (dada por la expresión $b - a + 1$) máxima de los intervalos $[a, b]$ (con $a < b$) tales que se cumple también el predicado $S(v, a, b, k)$ que indica que todos los elementos del intervalo deben ser menores estrictos que k. En este caso en el vector $[2, 3, 5, 0, 1, 4, -1]$ entre las posiciones 0 y $n = 5$ hay dos intervalos que cumplen que todos sus elementos son menores que 5: el que va desde la posición $a = 0$ hasta $b = 1$, de longitud 2, y el que va desde la posición $a = 3$ hasta la $b = 5$ de longitud 3. Por ello la máxima longitud es 3 y se cumple también la postcondición.
- d. Falso. La respuesta correcta es: Llamada $f([2, 3, 5, 0, 1, 4, -1], 5, 5)$ con resultado 3

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

Pregunta 3


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 = n, j = 1, s = 0;
while (i > 0){
    if (v[i] > v[j] && j <= n){
        s += v[i];
        j++;
    }
    i--;
}
```

Seleccione una:

- ☐ a. $i - 5$.
- ☐ b. j .
- ☐ c. $n - j$.
- ☒ d. Ninguna de las anteriores.  Cierto. La respuesta correcta es: i .

- a. Falso. La variable i decrece en cada iteración, pero la función de cota debe ser mayor o igual que cero siempre que la condición del bucle sea cierta. Si $i = 4$, $i > 0$ pero $i - 5 < 0$, por lo que la expresión no puede ser función de cota.
- b. Falso. La variable j no decrece sino que, según algunos valores del vector, aumenta. No puede ser una función de cota.
- c. Falso. La variable j crece únicamente en algunas iteraciones, dependiendo de los valores de v , la función de cota debe decrecer en cada iteración. No puede ser función de cota.
- d. Cierto. La respuesta correcta es: i .

La respuesta correcta es: Ninguna de las anteriores.

Pregunta 4

Sin contestar

Se puntúa como 0 sobre 1,00

Indica cuál de las siguientes afirmaciones es incorrecta

Seleccione una:

- ☐ a. $O(\log n) \subseteq O(n^4)$
- ☐ b. $O(3^n) \supseteq O(2^n)$
- ☐ c. $\Omega(n^5) \subseteq \Omega(n \log n)$
- ☐ d. Ninguna de las anteriores.

- a. Falso. Por el teorema del límite, como $\lim_{n \rightarrow \infty} \frac{n^4}{\log n} = \infty$, por lo que $O(\log n) \subseteq O(n^4)$.
- b. Falso. Por el teorema del límite, como $\lim_{n \rightarrow \infty} \frac{3^n}{2^n} = \infty$, $O(2^n) \subseteq O(3^n)$.
- c. Falso. Por el teorema del límite, como $\lim_{n \rightarrow \infty} \frac{n^5}{n \log n} = \infty$, $\Omega(n^5) \subseteq \Omega(n \log n)$.
- d. Cierto. La respuesta correcta es: $\Omega(\log n) \subseteq \Omega(n)$

La respuesta correcta es: Ninguna de las anteriores.

Pregunta 5

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. Termina únicamente para los valores de n entre 0 y 9
- ☐ c. No termina únicamente para los valores de n entre 0 y 10
- ☐ 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. Falso. Para esos valores la función llamará a sí misma con $n/10 = 0$ y como $0/10 = 0$ se llamará a sí misma infinitas veces.
- c. 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.
- 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 6

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

- a. 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 $T(n) \in \theta(n \log n)$.
- b. Falso. 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. 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 $T(n) \in \theta(n \log n)$.
- c. 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 $T(n) \in \theta(n \log n)$.
- 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 7

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. Sólo se implementan de forma recursiva
- ☒ b. Dividen el problema original en subproblemas cuyo tamaño es una fracción del original ✓ Cierto. Cada subproblema debe tener como tamaño una fracción del original.
- ☐ c. Son más eficientes que los algoritmos no divide y vencerás que resuelven el mismo problema
- ☒ d. 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.

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

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

Pregunta 8

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 = n + 131; i > n; i -= 1)
        x -= 14;
    for (int j = 16; j <= m + 15; j += 4)
        x -= 3;

    return x;
}
```

Seleccione una:

- ☐ a. $\theta(n \cdot m)$
- ☐ b. $\theta(m^2)$
- ☒ c. $\theta(m)$  Cierto. Hay dos bucles independientes. El primero da siempre un número constante de vueltas independientemente del valor de n, por lo que su coste es $\theta(1)$. Mientras que el segundo da un número de vueltas proporcional a m, por lo que su coste es $\theta(m)$. Por ello el coste total es $\theta(m)$.
- ☐ d. Ninguna de las anteriores.
- a. Falso. Hay dos bucles independientes. El primero da siempre un número constante de vueltas independientemente del valor de n, por lo que su coste es $\theta(1)$. Mientras que el segundo da un número de vueltas proporcional a m, por lo que su coste es $\theta(m)$. Al ser independientes el coste no es el producto sino la suma $\theta(1 + m) = \theta(m)$.
- b. Falso. Hay dos bucles independientes. El primero da siempre un número constante de vueltas independientemente del valor de n, por lo que su coste es $\theta(1)$. Mientras que el segundo da un número de vueltas proporcional a m, por lo que su coste es $\theta(m)$. Al ser independientes el coste no es el producto sino la suma $\theta(1 + m) = \theta(m)$.
- c. Cierto. Hay dos bucles independientes. El primero da siempre un número constante de vueltas independientemente del valor de n, por lo que su coste es $\theta(1)$. Mientras que el segundo da un número de vueltas proporcional a m, por lo que su coste es $\theta(m)$. Por ello el coste total es $\theta(m)$.
- d. Falso. La respuesta correcta es $\theta(m)$.

La respuesta correcta es: $\theta(m)$

Pregunta 9

Sin contestar

Se puntúa como 0 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. Una variable entera para indicar el total de cartas usadas.
- ☐ b. Un vector de booleanos de tamaño C que indique por cada carta si se ha usado o no.
- ☐ c. Un vector de enteros de tamaño 4 que indique por cada tipo de cartas cuántas se han usado.
- ☐ d. Ninguna de las anteriores.

- a. Falso. Incorrecto. El total de cartas usadas está dada por el parámetro $k + 1$ y al final por A , la altura de la torre.
- b. Falso. Incorrecto. No nos ayuda saber qué cartas concretas hemos usado, lo que nos importa es la cantidad de cartas de cada tipo que usamos en 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

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 es en promedio $\theta(n^2)$.
- ☒ c. 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 $\theta(n^2)$.
- ☐ 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. Falso. Incorrecta. Ese es el coste en el peor caso posible, si es que el pivote no divide el espacio de búsqueda en dos mitades. El coste en promedio de quicksort es mejor $\theta(n \log n)$.
- c. 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 $\theta(n^2)$.
- 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 $\theta(n^2)$.