

|                        |                                    |
|------------------------|------------------------------------|
| <b>Comenzado el</b>    | jueves, 19 de junio de 2025, 16:17 |
| <b>Estado</b>          | Finalizado                         |
| <b>Finalizado en</b>   | jueves, 19 de junio de 2025, 16:22 |
| <b>Tiempo empleado</b> | 4 minutos 22 segundos              |
| <b>Calificación</b>    | 2,33 de 10,00 (23,33%)             |


**Pregunta 1**

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. 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  $\theta(n^2)$ .
- ☐ b. Su coste en el mejor de los casos es  $\theta(1)$ .
- ☐ c. Sólo se puede implementar de manera recursiva.
- ☐ d. Ninguna de las anteriores.

- a. 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  $\theta(n^2)$ .
- b. Falso. Incorrecta. El mejor caso de quicksort es cuando el pivote divide el espacio de búsqueda en dos mitades. En este caso es  $\theta(n \log n)$ , nunca  $\theta(1)$ .
- c. Falso. Incorrecta. Quicksort es un algoritmo divide y vencerás que se puede implementar entero iterativamente como cualquier otro algoritmo 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 2**

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

- a. 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 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 pares que impares. Si llamamos con  $q = a - 1 = f - 1$  nunca comprobaríamos el intervalo  $[f-r, f)$ . Por eso es incorrecto.
- b. Cierto. 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  $n$ . Para llevar la cuenta del número de valores pares del intervalo usamos la variable  $j$ , mientras que  $i$  cuenta el número de valores impares. Cada intervalo corresponde a los índices  $[a - r, a)$ , ya que los índices van de 0 a  $v.size()$ . Por ejemplo, al final del bucle  $a = f$  lo que corresponde al último intervalo posible de longitud  $r$   $[f - r, f)$ .
- c. Falso. Incorrecto. La variable  $a$  es siempre  $a \leq f$ , de hecho únicamente se da que  $a = f$  al salir del bucle.
- d. Falso. La respuesta correcta es:  $(i = \#k: a - r \leq k < a: v[s]\%2 \neq 0) \wedge (j = \#k: a - r \leq k < a: v[s]\%2 = 0)$

La respuesta correcta es:  $(i = \#k: a - r \leq k < a: v[s]\%2 \neq 0) \wedge (j = \#k: a - r \leq k < a: v[s]\%2 = 0)$

**Pregunta 3**

Correcta


Se puntúa 1,00 sobre 1,00

Indica la complejidad del siguiente algoritmo

```
int f(int n, int m){
    int z = 0;
    for (int i = n + 194; i > n; i -= 1)
        z += 13;
    for (int j = 21; j <= m + 13; j += 5)
        z += 2;

    return z;
}
```

Seleccione una:

- ☐ a.  $\theta(m^2)$
- ☒ b.  $\theta(m)$   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  $\theta(m)$ .
- ☐ c. No se puede saber.
- ☐ 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. 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  $\theta(m)$ .
- c. 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  $\theta(m)$ . Por ello el coste total es  $\theta(m)$ . Así que sí que se puede saber.
- d. Falso. La respuesta correcta es  $\theta(m)$ .

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

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

Seleccione una:

- ☒ a.  $i$  ✓ Cierto. La variable  $i$  empieza en  $n$  y disminuye en cada vuelta donde además está garantizado que  $i > 0$ . Por eso es una función de cota válida.
- ☐ b.  $j$ .
- ☐ c.  $i - 5$ .
- ☐ d. Ninguna de las anteriores.

- a. Cierto. La variable  $i$  empieza en  $n$  y disminuye en cada vuelta donde además está garantizado que  $i > 0$ . Por eso es una función de cota válida
- 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  $i$  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.
- d. Falso. La respuesta correcta es:  $i$ .

La respuesta correcta es:  $i$ .

**Pregunta 5**

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  $f([10, 13, -2, 0, 7, 1], 8, 5)$  con resultado 5
- ☐ b. Llamada  $f([1, 2, 3, 4, 5, 6], 9, 0)$  con resultado 0
- ☐ c. Llamada  $f([-1, -2, 0, 0, -6, -10], 0, 2)$  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  $0 \leq n = 5 < 6$  como  $8 \in (-10, 10)$ . 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  $[10, 13, -2, 0, 7, 1]$  entre las posiciones 0 y  $n = 5$  hay un intervalo que cumple que todos sus elementos son menores que 8: el que va desde la posición  $a = 2$  hasta  $b = 5$ , de longitud 4, y no 5 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 tanto  $0 \leq n = 0 < 6$  como  $9 \in (-10, 10)$ , así que se satisface la precondition. 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  $0 \leq a < b \leq n = 0$ ) 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, no existen valores posibles para a y b que cumplan que  $0 \leq a < b \leq n = 0$ , por lo que el intervalo es vacío. Como la función no está definida en el rango vacío esto hace que la postcondición sea incorrecta. Por lo que no se cumple la especificación.
- c. 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.
- d. Cierto. La respuesta correcta es: Llamada  $f([2, 3, 5, 0, 1, 4, -1], 5, 5)$  con resultado 3

La respuesta correcta es: Ninguna de las anteriores.

**Pregunta 6**

Sin contestar

Se puntúa como 0 sobre 1,00

Dado la siguiente función recursiva:

```
int fun(const vector<int>&v, int i, int f){
    if ( i+1 == f && v[i]%2 != 0)
        return 1;
    else if ( i+1 == f && v[i]%2 == 0)
        return 0;
    else{
        int m = (f+i)/2;
        return fun(v, i, m) + fun(v, m, f);
    }
}
```

Si  $v = 11, 0, 2, 5, 1, 9$ , ¿Qué devuelve la llamada  $fun(v, 0, 5)$ ?

Seleccione una:

- ☐ a. 0
- ☐ b. 2
- ☐ c. 3
- ☐ d. Ninguna de las anteriores.

- a. Falso. La función devuelve el número de valores impares que hay en el intervalo  $[i, f)$ , al llamar con  $i = 0$  y  $f = 5$ . Hay 3 valores impares en ese intervalo: 11, 5 y 1
- b. Falso. La función devuelve el número de valores impares que hay en el intervalo  $[i, f)$ , al llamar con  $i = 0$  y  $f = 5$ . Hay 3 valores impares en ese intervalo: 11, 5 y 1
- c. Cierto. La función devuelve el número de valores impares que hay en el intervalo  $[i, f)$ , al llamar con  $i = 0$  y  $f = 5$ . Hay 3 valores impares en ese intervalo: 11, 5 y 1
- d. Falso. La respuesta correcta es: 3

La respuesta correcta es: 3

**Pregunta 7**

Incorrecta

Se puntúa -0,33 sobre 1,00

La tía Josefina tiene un negocio de dulces navideños y necesita repartir  $P$  paquetes con sus exquisiteces usando el menor número de camiones de reparto (para ahorrar costes). Todos los camiones de reparto son idénticos: tienen la misma capacidad y su alquiler cuesta lo mismo. Cada paquete tiene un volumen  $v_i$ , peso  $w_i$  y características diferentes que impiden que podamos meter todos los dulces juntos: por ejemplo, no se pueden apilar más de 3 cajas con dulces de hojaldre porque se estropean, etc.

Tu misión es minimizar el dinero que Josefina se gastará usando el menor número de camiones. Como ya sabes mucho de algoritmos te planteas si podrías hacer una poda del espacio de búsqueda. ¿Cuál de las siguientes sería una poda válida?

Seleccione una:

- ☐ a. Podar con la cantidad actual de camiones alquilados
  - ☒ b. No existe una poda para este tipo de problema ✗ Falso. Para un problema de minimización una poda correcta es el valor de la solución parcial, en este caso la cantidad de camiones alquilados por el momento
  - ☐ c. Podar considerando que cada paquete de dulces que queda por repartir necesita un nuevo camión de reparto
  - ☐ d. Ninguna de las anteriores.
- a. Cierto. Para un problema de minimización una poda correcta es el valor de la solución parcial, en este caso la cantidad de camiones alquilados por el momento
- b. Falso. Para un problema de minimización una poda correcta es el valor de la solución parcial, en este caso la cantidad de camiones alquilados por el momento
- c. Falso. Eso no funcionaría como poda de minimización ya que estaríamos sobreaproximando la cantidad de camiones que necesitamos. Una poda de minimización debe ser una cota inferior al coste, no una superior
- d. Falso. La respuesta correcta es: Podar con la cantidad actual de camiones alquilados

La respuesta correcta es: Podar con la cantidad actual de camiones alquilados

**Pregunta 8**

Incorrecta

Se puntúa -0,33 sobre 1,00

Indica cuál de las siguientes afirmaciones es correcta

Seleccione una:

- ☐ a.  $O(n^{20}) \supseteq O(2^n)$
- ☐ b.  $\Omega(n \log n) \subseteq \Omega(n)$
- ☒ c.  $\Omega(n) \subseteq \Omega(n^2)$  ✗ Falso. Por el teorema del límite, como  $\lim_{n \rightarrow \infty} \frac{n}{n^2} = 0$ ,  $\Omega(n) \not\subseteq \Omega(n^2)$ .
- ☐ d. Ninguna de las anteriores.

- a. Falso. Por el teorema del límite, como  $\lim_{n \rightarrow \infty} \frac{2^n}{n^{20}} = \infty$ ,  $O(2^n) \not\subseteq O(n^{20})$ .
- b. Cierto. Por el teorema del límite, como  $\lim_{n \rightarrow \infty} \frac{n \log n}{n} = \infty$ ,  $\Omega(n \log n) \subseteq \Omega(n)$ .
- c. Falso. Por el teorema del límite, como  $\lim_{n \rightarrow \infty} \frac{n}{n^2} = 0$ ,  $\Omega(n) \not\subseteq \Omega(n^2)$ .
- d. Falso. La respuesta correcta es:  $\Omega(n \log n) \subseteq \Omega(n)$

La respuesta correcta es:  $\Omega(n \log n) \subseteq \Omega(n)$

**Pregunta 9**

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

- a. 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  $n < = 1$ ;  $T(n) = 2 * T(n/2) + c_1 * n$  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  $n < = 1$ ;  $T(n) = 2 * T(n/2) + c_1 * n$  si  $n > 1$ , así que por el teorema de la división  $T(n) \in \theta(n \log n)$ .
- c. Falso.  $n$  está definida como la longitud del intervalo  $[a, b]$ . Así la recurrencia es  $T(n) = c_0$  si  $n < = 1$ ;  $T(n) = 2 * T(n/2) + c_1 * n$  si  $n > 1$ , así que por el teorema de la división  $T(n) \in \theta(n \log n)$  y no  $\theta(n)$ .
- d. Cierto. La respuesta correcta es:  $\theta(n \log n)$  con  $n = b - a$ .

La respuesta correcta es: Ninguna de las anteriores.



**Pregunta 10**

Incorrecta

Se puntúa 0,00 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. Son más eficientes que los algoritmos no divide y vencerás que resuelven el mismo problema ✗ Falso. Usar divide y vencerás no garantiza mejorar la complejidad.
- ☒ 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. 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.
- ☒ d. Sólo se implementan de forma recursiva ✗ Falso. Divide y vencerás es una estrategia de resolución de problemas. Por ejemplo, la búsqueda binaria se puede implementar de forma iterativa o recursiva.

a. Falso. Usar divide y vencerás no garantiza mejorar la complejidad.

b. Cierto. Cada subproblema debe tener como tamaño una fracción del original.

c. Cierto. Cada subproblema se debe poder resolver de forma independiente antes de combinar las soluciones para obtener la solución al problema 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 cuyo tamaño es una fracción del original, Dividen el problema original en subproblemas que se pueden resolver en paralelo