

Fundamentos de Algoritmia

Examen de junio

Curso 2021/2022

NOMBRE:

Observaciones:

- En el test, para cada pregunta hay una única respuesta correcta. Cada respuesta **correcta** vale **0,1 puntos** y cada respuesta **incorrecta** resta **0,033 puntos**.

1. Si la precondition de un algoritmo es **false**, ¿qué afirmación es correcta?

- (a) Cualquier postcondición será válida.
- (b) Si se ejecuta el algoritmo dicha ejecución no terminará.
- (c) Si se ejecuta el algoritmo se producirá un error en tiempo de ejecución.
- (d) Ninguna postcondición será válida.

a

2. Indica cuál de las siguientes afirmaciones es incorrecta:

- (a) $O(\log n) \subset O(n)$.
- (b) $O(2^n) \subset O(n^2)$.
- (c) $2 \in O(1)$.
- (d) $n \log(n) \in O(n^2)$.

b

3. Indica la complejidad del siguiente algoritmo:

```
int c = 0;
for (int i = 0; i < n; i += 2) c += 1;
for (int j = 0; j < n ; ++j) c += 3;
```

- (a) $\Theta(\log n)$.
- (b) $\Theta(n)$.
- (c) $\Theta(n^2)$.
- (d) Ninguna de las anteriores.

b

4. ¿Qué significa el siguiente predicado para un vector no vacío de naturales?

$$\forall i : 1 \leq i < v.size() : v[i-1] \neq v[i]$$

- (a) Todos los valores del vector son diferentes.
- (b) No existen dos valores iguales en el vector.
- (c) Los valores del vector están ordenados en orden creciente.
- (d) Ninguna de las anteriores.

d

5. Dada la especificación

```
{0 ≤ a.size}
fun contarImpares(vector<int> a) dev (int c)
{c = #i : 0 ≤ i < a.size : a[i] % 2 = 1}
```

y el siguiente algoritmo:

```
int contarImpares(std::vector<int> const& a) {
    int c = 0; int k = a.size()-1;
    while (k >= 0)
    {
        if (a[k] % 2 == 1) {c = c + 1;}
        k = k - 1;
    }
    return c;
}
```

indica si el algoritmo es correcto con respecto a la especificación y en tal caso cuál es el invariante que permite demostrar la corrección del bucle.

- (a) Es correcto con invariante $\{-1 \leq k < a.size \wedge c = \#i : 0 \leq i < k : a[i] \% 2 = 1\}$.
- (b) Es correcto con invariante $\{-1 \leq k \leq a.size \wedge c = \#i : k \leq i < a.size : a[i] \% 2 = 1\}$.
- (c) Es correcto con invariante $\{-1 \leq k \leq a.size \wedge c = \#i : k < i < a.size : a[i] \% 2 = 1\}$.
- (d) Ninguna de las anteriores.

c

6. Indica cuál de las siguientes propiedades sobre los órdenes de complejidad no es correcta, siendo f y g funciones de coste cualesquiera:

- (a) $\mathcal{O}(f + g) = \mathcal{O}(\max(f, g))$.
- (b) $\mathcal{O}(c \cdot f) = c \cdot \mathcal{O}(f)$
- (c) $\mathcal{O}(\log_a f) = \mathcal{O}(\log_b f)$
- (d) Ninguna de las anteriores.

d

7. Indica cuál es una función de cota para este algoritmo:

```
{x > 0 }
int i = x;
while (i >= 0)
{
    if (i%2 == 1) {i = i + 1;}
    i = i - 1;
}
{i%2 = 1}
```

- (a) $i + 1$
- (b) $x + i$
- (c) $\max(0, i - 1)$
- (d) Ninguna de las anteriores.

d

8. Dados los algoritmos de búsqueda lineal y de búsqueda binaria, indica cual de las siguientes afirmaciones es cierta (n indica el número de elementos del vector en que se realiza la búsqueda):

- (a) Ambos algoritmos tienen el mismo orden de complejidad en el caso peor.
- (b) El algoritmo de búsqueda lineal tiene coste $\mathcal{O}(1)$ y el de búsqueda binaria $\mathcal{O}(\log(n))$.
- (c) Ambos algoritmos se pueden aplicar sobre los mismos vectores de entrada.
- (d) Ninguna de las anteriores.

d

9. La siguiente especificación:

$P : v.size \geq 0 \wedge v = V$

void f (vector<int> v)

$Q : \forall k : 0 \leq k < v.size - 1 : v[k] \leq v[k + 1] \wedge permutacion(v, V)$

siendo el predicado $permutacion(v, w) \equiv v.size() = w.size() \wedge \forall k : 0 \leq k < v.size() : (\#x : 0 \leq x < v.size() : v[x] = v[k]) = (\#x : 0 \leq x < v.size() : w[x] = v[k])$

Se puede implementar con el siguiente algoritmo

- (a) Algoritmo quicksort o de ordenación rápida.
- (b) Algoritmo de partición.
- (c) Algoritmo de búsqueda binaria.
- (d) Ninguna de las anteriores.

a

10. Indica el coste de un algoritmo cuya recurrencia es:

$$T(n) = \begin{cases} c_0 & \text{if } n \leq 2 \\ T(n/2) + c_1 & \text{if } n > 2 \end{cases}$$

- (a) $\mathcal{O}(\log n)$
- (b) $\mathcal{O}(n)$
- (c) $\mathcal{O}(n \log n)$
- (d) $\mathcal{O}(n^2)$

a

1	2	3	4	5	6	7	8	9	10