Fundamentos de Algoritmia

Examen de enero — 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. Dada la especificación

$$\{v.size \geq 0\}$$
$$\text{fun xxx(vector<int> } v\text{) dev int } r$$
$$\{r = \max p, q : 0 \leq p \leq q \leq v.size \wedge \forall i : p \leq i < q : v[i] = 0 : q - p\}$$

y el vector de entrada $v = [5, 4, 0, 4, 3, 0, 0, 0, 0, 5, 3, 0, 0, 5]$, ¿cuál es el valor de r según la especificación?

(a) $r = 7$.
(b) $r = 0$.
(c) $r = 4$.
(d) Ninguna de las anteriores.

c

---

2. Un algoritmo óptimo que busca el máximo en un vector ordenado de n elementos tiene complejidad en el caso peor:

(a) $\Theta(\log n)$.
(b) $\Theta(1)$.
(c) $\Theta(n)$.
(d) Ninguna de las anteriores.

b

---

3. Indica la complejidad del siguiente algoritmo:

```
int c = 0;
for (int i = 1; i < n; i *= 2)
    for (int j = 0; j < m+2; ++j)
        c += 4;
```

(a) $\Theta(1)$.
(b) $\Theta(m \log n)$.
(c) $\Theta(nm)$.
(d) Ninguna de las anteriores.

b

---

4. Dado un vector a de n enteros, con $n \geq 1$, y una variable booleana b, el predicado

$$b = \exists w : 0 \leq w < n : (\exists k : 0 \leq k : a[w] = 2 * k + 1)$$

significa que la variable b toma el valor cierto si y solo si:

(a) Hay al menos una posición en el vector que contiene un número impar positivo.
(b) Nunca toma el valor cierto.
(c) Todas las posiciones del vector son impares.
(d) Ninguna de las anteriores.

a

---

5. Dada la especificación

$$\{a.size \geq 0\}$$
$$\text{fun contarPares(vector<int> } a\text{) dev int } c$$
$$\{c = \#i : 0 \leq i < a.size : a[i] \,\%\, 2 = 0\}$$

y el siguiente algoritmo:

```
int contarPares(vector<int> const& a) {
    int c=0; int k=-1;
    while (k<a.size()-1)
    {
        if (a[k+1] % 2 == 0) {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 $\{-1 \leq k < a.size \land c = \#i : 0 :\leq i < k : a[i] \,\%\, 2 = 0\}$.
(b) Es correcto con invariante $\{-1 \leq k < a.size \land c = \#i : 0 :\leq i \leq k : a[i] \,\%\, 2 = 0\}$.
(c) Es correcto con invariante $\{-1 \leq k \leq a.size \land c = \#i : 0 :\leq i < n : a[i] \,\%\, 2 = 0\}$.
(d) Ninguna de las anteriores.

b

---

6. Indica cuál de las siguientes afirmaciones sobre el algoritmo *quicksort* (ordenación rápida) es **falsa**:

(a) Si los valores del vector están ordenados en orden creciente tiene un orden de complejidad cuadrático respecto al tamaño del vector.
(b) Si los valores del vector presentan una distribución uniforme en un intervalo de va-lores, el algoritmo tiene una complejidad $n \log n$ siendo n el número de elementos del vector.
(c) El algoritmo tiene una complejidad $n \log n$ siendo n el número de elementos del vector tanto en el caso peor como en el caso medio.
(d) El algoritmo tiene una complejidad cuadrática respecto al número de elementos del vector en el caso peor.

c

---

7. Dos algoritmos que tienen el mismo orden de complejidad:

(a) Se comportan de forma semejante para tamaños de entrada grandes.
(b) Se comportan de forma semejante para tamaños de entrada pequeños.
(c) Tardan exactamente el mismo tiempo en ejecutarse.
(d) Todas de las anteriores.

a

---

8. Indica cuál de los siguientes es un requisito imprescindible para poder utilizar el algoritmo de la búsqueda binaria sobre un vector:

(a) El vector debe tener al menos un elemento.
(b) Los elementos del vector deben ser números enteros.
(c) Los valores del vector deben estar ordenados (según un orden bien definido).
(d) Todas las anteriores.

c

---

9. Indica qué función de cota utilizarías para probar la terminación del siguiente bucle:

```
int k = 0; int N = 100;
for (int i = N-1; i > -N; --i) ++k;
```

(a) $f(i, N) = i$.
(b) $f(i, N) = i + N$.
(c) $f(i, N) = i - N$.
(d) $f(i, N) = N - i$

$\boxed{\text{b}}$

---

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

$$T(n) = \begin{cases} c_0 & \text{si } n == 0 \\ T(n-1) + n & \text{si } n > 0 \end{cases}$$

(a) $\mathcal{O}(n)$.
(c) $\mathcal{O}(n \log n)$.
(b) $\mathcal{O}(n^2)$.
(d) Ninguna de las anteriores.

$\boxed{\text{c}}$

1	2	3	4	5	6	7	8	9	10