Fundamentos de Algoritmia

Examen de enero. Tipo A. Curso 2022/2023

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

   $\{n = \text{longitud}(v)\}$
   fun foo(int $v[\,]$, int n) dev int r
   $\{r = \max p, q : 0 \leq p \leq q \leq n \wedge \forall i : p \leq i < q - 1 : v[i+1] = v[i] + 1 : q - p\}$

   y el vector de entrada $v = [5, 1, 2, 4, 6, 7, 8, 9, 10, 12, 11, 0]$, ¿cuál es el valor de r según la especificación?

   (a) $r = 7$.
   (b) $r = 5$.
   (c) $r = 4$.
   (d) Ninguna de las anteriores.

   b

---

2. Para calcular el mínimo de los valores de un vector de tamaño n que contiene números enteros pertenecientes al intervalo [1..1000]:

   (a) Se puede calcular utilizando divide y vencerás con una complejidad $\Theta(\log n)$.
   (b) Un algoritmo eficiente ordena el vector y obtiene el elemento en la posición 0.
   (c) Si el vector es vacío el mínimo es siempre 0.
   (d) La mejor complejidad que se puede obtener es del orden $\Theta(n)$.

   d

---

3. Indica la complejidad del siguiente algoritmo:

```
int c = 0;
for (int j = 0; j < n+2; j+=2)
   for (int i = 1; i < m; i *= 3)
      c += 4;
```

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

   c

---

4. ¿Cuál de los siguientes predicados especifica que un vector a de n elementos está ordenado de forma creciente?

   (a) $\forall w : 0 \leq w < n - 1 : a[w] \leq a[w+1]$.
   (b) $\forall w : 0 \leq w < n : a[w] \leq a[w+1]$.
   (c) $\exists w : 0 \leq w < n - 1 : a[w] \leq a[w+1]$.
   (d) Ninguna de las anteriores.

   a

---

5. Indica cuál de las propiedades es un invariante del siguiente bucle:

```
int y=1; int j=n;
while (j>0) {
    y=y*x;
    j--;
}
```

(a) $y = x^j$. (b) $0 < j \leq n$. (c) $y = x^{n-j}$. (d) Ninguna. **c**

---

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

$$T(n) = \begin{cases} c_0 & \text{si } n = 0 \\ 2T(n/2) + n & \text{si } n > 0 \end{cases}$$

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

---

7. Dados dos algoritmos A y B con órdenes de complejidad $\Theta(f(n))$ y $\Theta(g(n))$ respectivamente, si $\Theta(f(n)) \subset \Theta(g(n))$:
   (a) El algoritmo A siempre tiene un tiempo de ejecución menor que el algoritmo B independientemente del tamaño de entrada.
   (b) El algoritmo A tiene un tiempo de ejecución menor que el algoritmo B solo para tamaños de entrada grandes.
   (c) El tamaño de entrada a partir del cual el algoritmo A tiene un tiempo de ejecución menor que el algoritmo B depende de la constante multiplicativa del algoritmo.
   (d) El algoritmo A siempre tiene un tiempo de ejecución mayor que el algoritmo B, independientemente del tamaño de entrada. **c**

---

8. Indica cuál de las siguientes respuestas no es correcta respecto al algoritmo de búsqueda binaria que comprueba si un valor se encuentra en una secuencia de valores.
   (a) El algoritmo de búsqueda binaria solo se puede aplicar sobre secuencias de valores ordenadas.
   (b) Si la secuencia de valores no está ordenada es más eficiente ordenarla y buscar el elemento con búsqueda binaria que realizar una búsqueda secuencial.
   (c) La búsqueda binaria sobre un vector vacío siempre devuelve como resultado `false`.
   (d) La búsqueda binaria en un vector ordenado siempre es más eficiente que la búsqueda secuencial para secuencias de valores grandes. **b**

---

9. El orden en el que se recorren las ramas del árbol de exploración en un algoritmo de vuelta atrás influye en las podas que se realizan y por lo tanto en el tiempo de ejecución del algoritmo:
   (a) Siempre.
   (b) Nunca.
   (c) Solo podría influir en los problemas de optimización.
   (d) Nunca podría influir en los problemas de optimización. **c**

---

10. Un algoritmo que obtiene las combinaciones de k elementos que pueden formarse con n elementos diferentes utilizando la técnica de vuelta atrás tiene complejidad:

(a) $\Theta(n)$. (b) $\Theta(n * k)$. (c) $\Theta(k^n)$. (d) $\Theta(n^k)$. **d**
