B12SO

Normas de valoración del examen:

- La nota del examen representa el 80% de la valoración final de la asignatura (el 20% restante corresponde a las prácticas).
- Cada cuestión contestada correctamente vale 1 punto.
- Cada cuestión contestada incorrectamente baja la nota en 0.3 puntos.
- Debe obtenerse un mínimo de 3 puntos en las cuestiones para que el problema sea valorado (con 3 cuestiones correctas y alguna incorrecta el examen está suspenso).
- La nota total del examen debe ser al menos de 4.5 para aprobar.
- **Las cuestiones se responden en una hoja de lectura óptica**.

1. Indique cuál de las siguientes afirmaciones es cierta:

    a. La programación dinámica se puede emplear en muchos de los problemas que se resuelven utilizando el esquema divide y vencerás.
    b. La programación dinámica tiene como principal ventaja que supone un decremento tanto en coste computacional como en espacio de almacenamiento.
    c. El esquema de programación voraz se aplica a problemas de optimización en los que la solución se puede construir paso a paso comprobando, en cada paso, la secuencia de decisiones tomadas anteriormente.
    d. El esquema de ramificación y poda es el más indicado a utilizar en el caso que se deseen encontrar todas las soluciones posibles a un problema dado.

2. Los residentes de una ciudad no quieren pavimentar todas sus calles (que son de doble sentido), sino sólo aquellas que les permitan ir de una intersección a otra cualquiera de la ciudad con comodidad. Quieren gastarse lo menos posible en la pavimentación, teniendo en cuenta que el coste es directamente proporcional a la longitud de las calles que hay que pavimentar. El alcalde querría saber qué calles tiene que pavimentar para gastarse lo menos posible. ¿Cuál de los siguientes esquemas es más eficiente de los que puedan resolver el problema correctamente?

    a. Esquema voraz.
    b. Esquema de programación dinámica.
    c. Esquema de vuelta atrás.
    d. Esquema de ramificación y poda.

3. En el recorrido mediante doble hashing, la función h'(x) debe cumplir:
    a. Necesariamente debe ser $h'(x) \neq 0$.
    b. h'(x) debe tener la forma $h'(x) = 2h(x) + c$, con c constante.
    c. Se debe cumplir que $h'(x) < h(x) + c$, con c constante.
    d. Ninguna de las anteriores es cierta.

4. Considérese el vector [10,7,7,4,3,1] que representa un montículo. ¿Cuál sería la representación resultante de insertar en este montículo el valor 11 usando la función flotar?

    a. [11,7,7,4,3,1,10].
    b. [10,7,11,4,3,1,7].
    c. [10,7,11,4,3,1,7].
    d. Ninguna de las opciones anteriores.

5. Dado un grafo con *n* vértices y *a* aristas sobre el que se pueden realizar las siguientes operaciones:

    - *esAdyacente*: comprueba si dos vértices son adyacentes.
    - *borrarVértice*: Borra el vértice especificado y todas sus aristas.
    - *añadirVértice*: Añade un nuevo vértice al grafo.

    ¿Cuál es la complejidad de cada una de las operaciones anteriores según se utilice una matriz de adyacencia (*MA*) o una lista de adyacencia (*LA*)?

a)

	MA	**LA**
esAdyacente	O(1)	O(1)
borrarVértice	O(n)	O(n)
añadirVértice	O(n)	O(1)

b)

	MA	**LA**
esAdyacente	O(1)	O(n)
borrarVértice	O(1)	O(n+a)
añadirVértice	O(1)	O(1)

c)

	MA	**LA**
esAdyacente	O(1)	O(n)
borrarVértice	O(n)	O(n+a)
añadirVértice	O(n)	O(1)

d)

	MA	**LA**
esAdyacente	O(1)	O(1)
borrarVértice	O(n)	O(n)
añadirVértice	O(n)	O(n)

6. Suponga un algoritmo que dadas dos listas de tamaños *n* y *m*: $A = a_1,\ldots a_n$ y $B = b_1,\ldots,b_m$ dé como resultado la lista de aquellos elementos que pertenecen a *A* pero no a *B*. Se asume que *A* y *B* no contienen elementos duplicados, pero no que éstos estén acotados. Indique cuál de los siguientes costes se ajusta más al del algoritmo que puede resolver el problema con menor coste:

    a. O(n log m + m log m).
    b. O(n log m + m^2).
    c. Máx (O (n log n), O(m log m)).
    d. O(nm).

Problema (4 puntos). Teseo se adentra en el laberinto en busca de un minotauro que no sabe dónde está. Se trata de implementar una función *ariadna* que le ayude a encontrar el minotauro y a salir después del laberinto. El laberinto debe representarse como una matriz de entrada a la función cuyas casillas contienen uno de los siguientes tres valores: 0 para "camino libre", 1 para "pared" (no se puede ocupar) y 2 para "minotauro". Teseo sale de la casilla (1,1) y debe encontrar la casilla ocupada por el minotauro. En cada punto, Teseo puede tomar la dirección Norte, Sur, Este u Oeste siempre que no haya una pared. La función *ariadna* debe devolver la secuencia de casillas que componen el camino de regreso desde la casilla ocupada por el minotauro hasta la casilla (1,1).

La resolución de este problema debe incluir, por este orden:

1. Elección del esquema más apropiado, el esquema general y explicación de su aplicación al problema (0,5 puntos).
2. Descripción de las estructuras de datos necesarias (0.5 puntos solo si el punto 1 es correcto).
3. Algoritmo completo a partir del refinamiento del esquema general (2,5 puntos solo si el punto 1 es correcto). Si se trata del esquema voraz debe hacerse la demostración de optimalidad.
4. Estudio del coste del algoritmo desarrollado (0.5 puntos solo si el punto 1 es correcto).