Examen. 6 de febrero de 2.007

Resolver cada pregunta en una hoja distinta.
No hay que entregar esta hoja con el examen.
Indicar el tipo de examen en la primera hoja.

1. (2 puntos) Escribir una especificación formal axiomática del TAD **TablaCerrada[C,V]**, de tablas de dispersión cerrada con claves de tipo **C** y valor **V**, con las operaciones *crear* (crea una tabla de un tamaño dado), *esVacía*, *añadir* (se añade a la tabla una asociación clave/ valor), *está* (indica si una clave dada se encuentra en la tabla), *recuperar* (devuelve el valor asociado a una clave en la tabla), *borrar* (elimina la asociación de la tabla dada su clave), *tamaño* (devuelve el número de asociaciones existentes en la tabla) y *estáLlena* (dice si la tabla está llena o no). Hay que tener en cuenta que si se inserta una clave que ya está en la tabla, la operación *añadir* sustituye el valor asociado por el nuevo valor. Suponer que nunca se intenta insertar una nueva asociación cuando la tabla está llena. ¿Cómo debería variar la especificación formal (por ejemplo, para la operación *está*) si no se hace esa suposición?

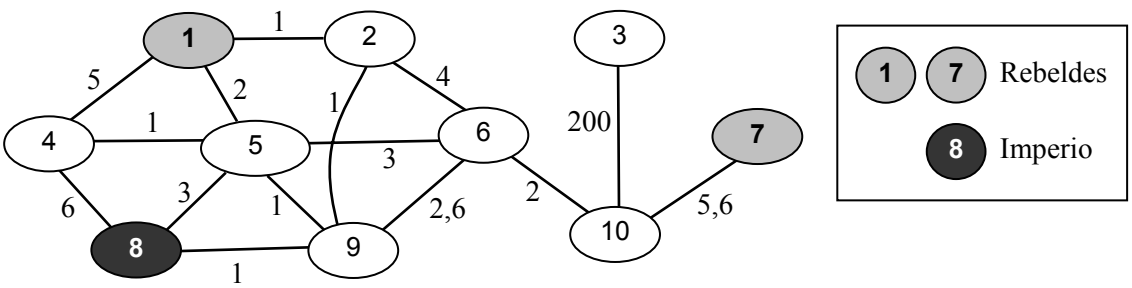
Nota: La pregunta 1 no deben hacerla los alumnos que tengan aprobada la práctica de especificaciones formales.

2. (2,6 puntos) Sobre una estructura de árboles trie queremos añadir una operación de listado ordenado. Dicha operación recibe un trie y dos palabras como parámetro **p1** y **p2**, siendo **p1** alfabéticamente menor o igual que **p2**. El procedimiento debe listar todas las palabras del trie que estén alfabéticamente entre **p1** y **p2** (ambas inclusive).

   Programar dicho procedimiento utilizando las operaciones genéricas sobre nodos trie: **Consulta (n: trie, c: carácter): trie** (devuelve el hijo del nodo **n** para el carácter **c**, o NULO si no existe), **Inserta (n: trie, c: carácter, m: trie)** (añade al nodo **n** el hijo **m** asociado al carácter **c**), **NuevoTrie: trie** (devuelve un nuevo nodo vacío) y un iterador del tipo **para cada carácter c hijo del nodo n hacer** (iterador de letras, en orden alfabético). Tener en cuenta, por ejemplo, que: "CAL" < "CALA" < "CALABAZA" < "CALABAZATE". Suponer las operaciones típicas con cadenas (concatenación, acceso, etc.), incluidas las de comparación, "<".

3. (3 puntos) ¡El Imperio Galáctico está en peligro! Algunos sistemas planetarios han sido invadidos por las tropas rebeldes. Otros muchos sistemas permanecen neutrales. Pero antes o después deberán aliarse a uno de los dos bandos. Los malvados rebeldes van a enviar mensajeros a todos los sistemas neutrales, para que se sumen a su causa. Lo mismo harán las tropas del Imperio. Los mensajeros siempre parten de un sistema controlado por su bando, y pueden pasar por sistemas neutrales o controlados por ellos mismos. El primero que llegue a cada sistema neutral, conseguirá sumarlo a sus aliados. Pero si ambos bandos llegan al mismo tiempo, se producirá una batalla y ese sistema y los mensajeros que han llegado quedarán completamente destruidos en una terrible explosión neutrónica de antimateria. El objetivo de ambos es conseguir el mayor número de aliados.

   Suponer que los dos bandos siguen una estrategia óptima. Escribir un algoritmo que diga, para cada sistema neutral, si será controlado por los rebeldes, por el Imperio, o acaba explotando, y el instante en que lo será. Los datos del problema son: existen **n** sistemas en total. El array **G**[1...**n**], contiene valor **G[i]**=1 si el sistema **i** está controlado por los rebeldes, -1 por el Imperio, y 0 si es neutral. La matriz **M** de tamaño **n**x**n** indica el tiempo de los caminos intergalácticos, siendo **M[i, j]** el número de eones para ir del sistema **i** al **j**. El número de mensajeros no está limitado. Si se utiliza alguno de los algoritmos vistos en clase, se deberá especificar de forma detallada.

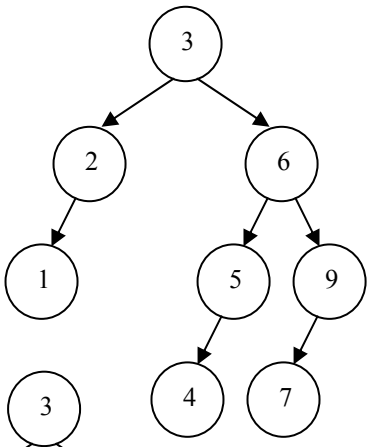


4. (2,4 puntos) Indicar las respuestas que son válidas para cada una de las siguientes preguntas. Tener en cuenta que puede haber una, varias o ninguna respuesta correcta por pregunta (es decir, la respuesta no es necesariamente única). Una pregunta sólo se dará por correcta si se señalan todas las afirmaciones correctas. No es necesario justificar las respuestas.

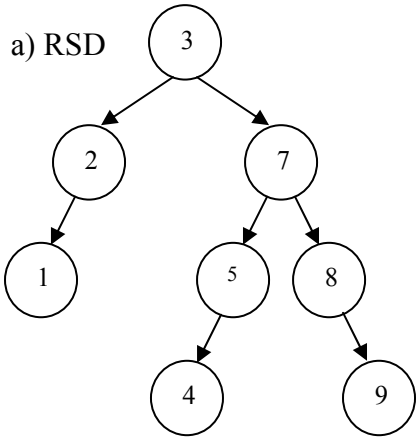
   (1) Los árboles AVL:
   a. Son un tipo particular de árbol de búsqueda.
   b. Pueden ser considerados como árboles B de orden $p = 2$.
   c. Todo subárbol del mismo es un árbol AVL.
   d. Si una inserción requiere reequilibrar el árbol, las operaciones de rotación hacen que la complejidad en el peor caso sea mayor que en un ABB no balanceado.

   (2) Las tablas de dispersión:
   a. Van degradando su eficiencia al aumentar la ocupación de la tabla.
   b. No requieren comparaciones entre claves en las búsquedas.
   c. Sólo se pueden utilizar si el universo de las claves es finito y numerable.
   d. Son poco adecuadas cuando se necesitan muchas operaciones de acceso ordenado.

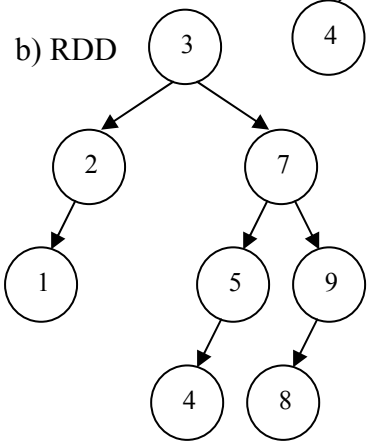
   (3) Suponiendo que tenemos el siguiente árbol AVL, indicar cuáles serían las rotaciones aplicadas y el árbol resultante, después de borrar el elemento 6 y seguidamente insertar el 8. **Nota:** dado que en ciertas operaciones hay variantes, es posible que varios de los árboles puedan ser válidos. Si es así, hay que indicarlos todos.



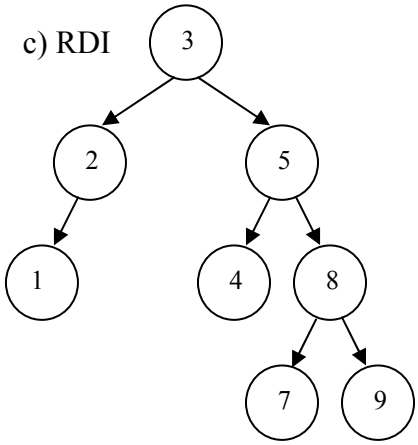
a) RSD



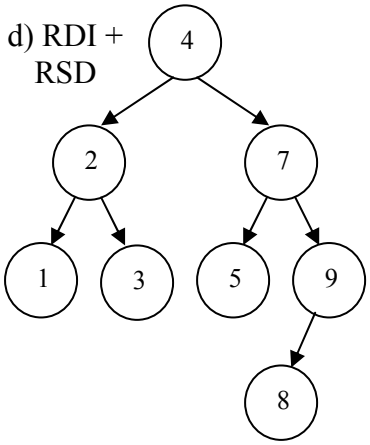
b) RDD



c) RDI

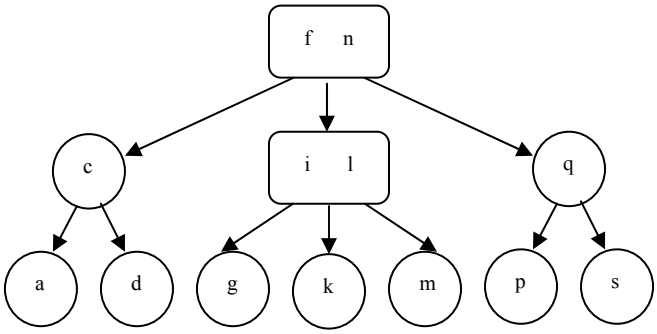


d) RDI + RSD

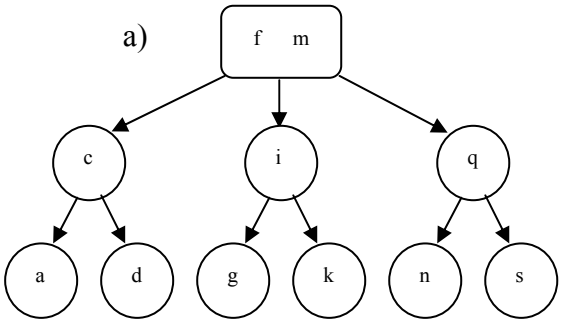


(4) Deseamos aumentar las operaciones del TAD **grafo dirigido** con una operación *Invertir* que, dado un grafo, lo convierte en otro con el sentido de los arcos invertido. ¿Con qué implementación sería más simple el código que realice esta operación?
   a. Con listas de adyacencia.
   b. Con matrices de adyacencia.
   c. En ambas estructuras sería igual de ineficiente.

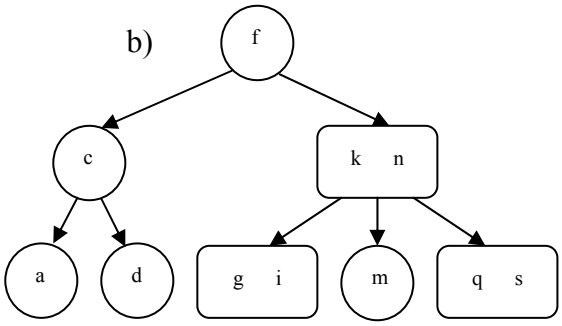
(5) Indicar el árbol B correcto resultante de ejecutar el borrado de los elementos **l** y **p** (en ese orden) en el árbol B de orden $p = 3$ dado:



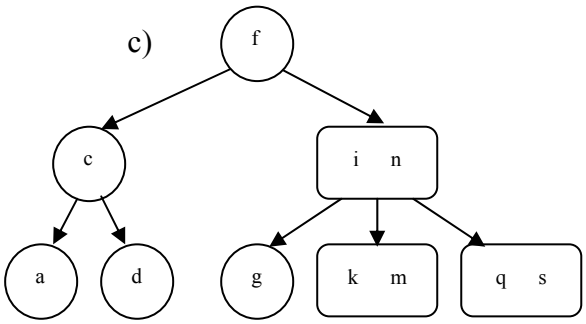
a)



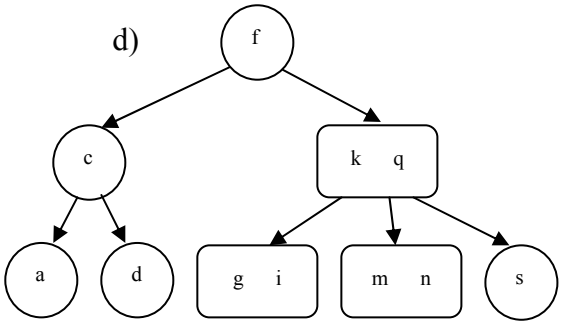
b)



c)



d)



(6) En la estructura eficiente para el TAD **relación de equivalencia**, mediante punteros al padre, con balanceo de árboles y compresión de caminos:
   a. La técnica de compresión de caminos es irrelevante si sólo se producen operaciones de unión y no hay ninguna de consulta.
   b. Para que los árboles no aumenten de altura, se coloca el más profundo como hijo del menos profundo.
   c. La altura de los árboles sólo aumenta cuando se unen dos raíces de igual profundidad.
   d. Realmente las relaciones de equivalencia no se pueden considerar como tipos de datos abstractos, ya que los TAD son, por ejemplo, los conjuntos, las pilas, colas, listas, etc.

Examen. 6 de febrero de 2.007

Resolver cada pregunta en una hoja distinta.
No hay que entregar esta hoja con el examen.

1. (3 puntos) Un algoritmo de divide y vencerás tiene la siguiente ecuación de recurrencia:

$$t(n) = 7 \cdot t(n-2) + 6 \cdot t(n-3) + 2n^2 + n \cdot 3^n$$

Siendo los casos base: $t(n) = 2n + 1$, para $n < 5$.

a) Calcular el orden de complejidad del algoritmo, expresándolo con la notación asintótica más adecuada.
b) Explicar lo que habría que hacer para calcular la o-pequeña del tiempo de ejecución de este algoritmo. No es necesario calcularla, pero sí dejar claro cómo hacerlo.
c) Escribir un algoritmo sencillo que tenga ese tiempo de ejecución. Estudiar su ocupación de memoria, dando la función **m(n)**, memoria ocupada para tamaño **n**.

2. (2 puntos) Un tranvía murciano tiene un recorrido lineal: existen **n** paradas situadas secuencialmente, siendo **1** la primera y **n** la última. Por simplicidad, supondremos que el tranvía tarda un tiempo unitario entre cada par de paradas. Sea **a** la parada actual del tranvía. En cada instante podemos movernos a la siguiente parada (**a**+1), o a la anterior (**a**–1). Inicialmente, el tranvía se encuentra en la parada **b**.

   El array **P**[1, ..., **n**] indica en cada posición **P**[**i**] el número de pasajeros que se encuentran inicialmente en la parada **i**. El tranvía debe pasar por las paradas, recogiendo a todos los pasajeros en el tiempo mínimo. El objetivo del problema es planificar los movimientos del tranvía (avanzar/retroceder) de manera que se minimice el tiempo total de espera de los viajeros (es decir, la suma de los tiempos de espera de cada pasajero). Por ejemplo, si el tranvía está inicialmente en la parada **b**= 4, el tiempo de espera para los pasajeros de **P**[4] es 0. Si a continuación se mueve a la parada 5, los **P**[5] esperan 1 instante cada uno, y así sucesivamente.

   Diseñar y programar un algoritmo voraz que encuentre una buena solución para el problema, devolviendo la ruta del tranvía y el tiempo total de espera resultante. Mostrar la solución del algoritmo para los siguientes casos: **P**= (1, 0, 0, 0, 2) con **b**= 2, y con **b**= 3; y **P**= (2, 100, 0, 10, 1) con **b**= 3.

3. (2,5 puntos) En el problema de la mochila 0/1 tenemos 4 objetos con pesos **p**= (1, 3, 4, 5) y beneficios **b**= (2, 5, 6, 6). La capacidad de la mochila es **M**= 7.

   Aplicar el algoritmo de programación dinámica visto en clase sobre el ejemplo. Deducir razonadamente la forma de la ecuación de recurrencia, con sus casos base. Mostrar la tabla resultante para este caso concreto. A partir de ella, encontrar los objetos que forman parte de la solución, explicando el proceso. ¿Cómo se puede saber si la solución óptima es única? ¿Coincide la solución con la que devolvería el algoritmo voraz?

4. (2,5 puntos) Resolver el problema del ejercicio 2 por backtracking o por ramificación y poda. Se deberán utilizar los esquemas vistos en clase, que se pueden dar por supuestos. Definir la forma de representar la solución y las funciones genéricas del esquema.

Nota: Los alumnos que tengan convalidada la Práctica 4 (o los que tengan aprobada la asignatura de plan antiguo "Laboratorio de Programación") pueden elegir entre convalidar la pregunta 2 o la 4. En cualquier caso, la pregunta convalidada valdrá 2 y la otra 2,5.