ÁRBOLES GENERALES, ABB, AVL , APO – FAQ – v3.0 Sep 2016

1.- ¿Por qué no se puede implementar un árbol general con un vector de posiciones relativas?

Porque no podemos conocer su grado (número máximo de hijos), luego no podemos prever la posición que deberían ocupar sus hijos.

Nota: No es lo mismo el por qué no se puede que el cuándo es adecuado. Ver pregunta 8 de Árboles Binarios.

2.- ¿Se podrían usar las listas doblemente enlazadas en los árboles generales mediante listas de hijos? **Editada.**

Sí, ya que una lista doblemente enlazada posee características mejoradas de una lista enlazada, usada en nuestra representación. Por lo anterior, si usamos una lista doblemente enlazada, vamos a poder cumplir con todas las especificaciones del TAD.

3.- ¿Existe alguna operación claramente ineficiente en los árboles generales representados mediante listas de hijos? -> *Examen 2015.* **Revisar. Editada.**

La operación constr, ya que tiene que recorrer todo el vector para poner los padres a null, y las de insertar, porque tiene que buscar la primera posición en esté a null, luego las dos en el peor caso serán de O(n).

4.1- ¿Puede determinarse un árbol general a partir del inorden y postorden?

No se puede, harían falta los 3 tipos de orden (PreOrden, InOrden y PostOrden), ya que podría darse el caso de:

Post(A)=Post(B) ^ In(A)=In(B)

4.2- ¿Puede reconstruirse un árbol unívocamente dado su inorden? **New. Revisar.**

No, ya que no tenemos forma de saber cuál es el nodo raíz ni qué elementos pertenecen al subárbol izquierdo ni cuáles al derecho. Para reconstruir un árbol a partir del inorden, será necesario otro recorrido cualquiera (preorden o postorden).

4.3- ¿Puede reconstruirse un árbol unívocamente dado su recorrido en preorden y su recorrido en postorden? **New. Revisar.**

No se puede, harían falta los 3 tipos de orden (PreOrden, InOrden y PostOrden), ya que podría darse el caso de:

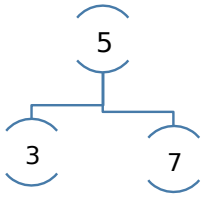
Post(A)=Post(B) ^ In(A)=In(B)

No, ya que no tenemos forma de saber cuál es el nodo raíz ni qué elementos pertenecen al subárbol izquierdo ni cuáles al derecho. Para reconstruir un árbol a partir del inorden, será necesario otro recorrido cualquiera (preorden o postorden).

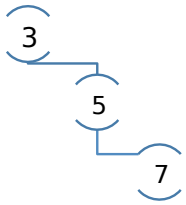
5.- ¿Puede reconstruirse un ABB de forma unívoca dado su recorrido en inorden?
Editada completamente. Respuesta renovada.

No, ya que si partimos de una lista con este recorrido, cualquier árbol de búsqueda que contuviera exactamente los elementos de la lista daría lugar al mismo recorrido en inorden, al consistir este en una lista ordenada (Martí-Oliet, Ortega-Mallén, y Verdejo-López 2004, p.187).

A:



B:



- In(A) = 3, 5, 7 ^
  In(B) = 3, 5, 7 =>
  In(A) = In(B)

el inorden ordena los elementos del ABB

6.- ¿Puede reconstruirse un ABB de forma unívoca dado su recorrido en preorden?
Editada completamente. Respuesta anterior decía que NO, y la correcta es SI.

Si. Si la lista no es vacía (árbol vacío), el elemento más a la izquierda de la lista será la raíz del árbol. Como es de búsqueda, los elementos en el hijo izquierdo serán menores. En el recorrido en preorden, el hijo izquierdo se recorre de forma recursiva después de la raíz, por lo que todos los elementos menores formarán el recorrido en preorden del hijo izquierdo. Por último se recorre el hijo derecho de forma recursiva, por lo que el resto de elementos mayores formarán el recorrido en preorden del hijo derecho. Conociendo así los recorridos de ambos hijos, podemos construirlos con sendas llamadas recursivas. (Martí-Oliet, Ortega-Mallén, y Verdejo-López 2004, p.187).

7.- ¿Qué consiguen los árboles binarios de búsqueda (ABB)?

Consiguen que las operaciones de inserción y de búsqueda en el caso promedio sean de orden O(log n) cuando están equilibrados, aunque en el peor de los casos seguirá siendo de O(n) (árbol degenerado en una lista).

8.- ¿Qué condición tienen que cumplir los elementos de un árbol para poder realizar las búsquedas con un coste menor que O(n)? **Misma que pregunta 11 de Árboles Binarios.**

Que tengan establecida una relación de orden entre sí y que se posicionen en el árbol equilibrándolo.

9.- ¿Qué condiciones debe cumplir un ABB para que la búsqueda sea menor que O(n)?
Editada. Revisar.

Que el ABB esté lo más equilibrado posible. Destacar los AVL (ABB con autoequilibrado), donde la búsqueda, la inserción y la eliminación son siempre de orden O(log n).

10.- ¿Es siempre la eliminación de elementos en un ABB de orden mayor que O(n)?
Editada. Revisar.

No, nunca es de orden mayor a O(n). Si tenemos un ABB equilibrado, la operación de eliminar nunca será mayor que de O(log n), y si el ABB ha degenerado en una lista, la operación será de orden O(n).

11.- ¿En qué condiciones puede un árbol ser un APO y ABB simultáneamente?
Editada. Revisar.

Aparte de cuando el árbol esté vacío o sólo tenga el nodo raíz, nunca.

12.- ¿Qué aportan los AVL frente a los ABB?

En un ABB, las operaciones de búsqueda, inserción y eliminación son, en el caso medio (cuando están equilibrados) de orden O(log n) y en el peor de los casos de orden O(n) (árbol degenerado en una lista). Un AVL, por definición, es un ABB equilibrado, por lo que elimina el peor de los casos anteriormente citado, luego garantiza que las operaciones sean de orden O(log n).

13.- Definición de AVL y factor de equilibrio.

Se define el factor de equilibrio de un nodo como la altura del subárbol derecho menos la altura del subárbol izquierdo de dicho nodo. Se define un árbol binario equilibrado como un árbol binario (AVL) en el cual el factor de equilibrio de cada nodo es -1, 0 ó 1. Se define un árbol AVL como un árbol binario de búsqueda equilibrado.

14.- ¿Influye el orden de inserción de los datos en la altura de AVL?

No, no influye. Esto es debido a que por definición, un árbol AVL tiene la propiedad de auto equilibrado. Es decir, en todo momento para todos los nodos, la altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa, siendo esto independiente al orden de inserción de los datos.

15.- ¿Por qué se exige un orden exacto en la inserción de elementos en el AVL?

En los AVL no se exige un orden exacto en la inserción de elementos, ya que la propia operación de inserción se encarga de mantener el árbol equilibrado.

16.- ¿Influye el orden de inserción de los elementos para el desequilibrio de un APO?
-> ***Examen 2014***

En los APO no influye el orden de inserción de los elementos para el desequilibrio, ya que, por definición, es un árbol completo, los cuales siempre están equilibrados. El orden de inserción no sirve para nada.

17.- ¿Puede reconstruirse un APO de forma unívoca dado su recorrido en preorden?

Sí, dado que un APO es un árbol completo en el que el valor de cualquier nodo es menor que el de sus descendientes. Con el preorden obtenemos la raíz y el número de nodos y, por lo tanto, la altura del árbol. Con estos datos y a partir de la definición de APO podemos reconstruir el árbol.

18.- ¿Es posible obtener coste O(n) en la eliminación de un nodo cualquiera de un APO?

No es posible eliminar cualquier nodo de un APO, ya que solamente se puede eliminar la raíz del mismo. Destacar que esta eliminación siempre toma un tiempo O(log2 n), puesto que en el árbol ningún camino tiene más de <<1+log2 n>> nodos y el proceso de hundir intercambiando con los hijos toma un tiempo constante por nodo.

19.- ¿Es posible obtener coste O(n) en la inserción/eliminación de la raíz en un APO?

No. Un APO por definición es un árbol completo parcialmente ordenado. Por tanto, cualquiera de las dos operaciones toma un tiempo O(log2 n), puesto que en el árbol ningún camino tiene más de <<1+log2 n>> nodos, ya que flotan y hunden respectivamente.

20.- ¿Influye el número de elementos de un APO en su desequilibrio?

No, ya que un APO por definicion, es un arbol completo, es decir, que solamente pueden faltarle nodos en el ultimo nivel y por la parte derecha, luego el factor de equilibrio va a ser -1 en el peor caso.

21.- ¿Tiene sentido el concepto de un árbol terciario de búsqueda?

Sí. La generalización de los ABB para árboles generales son los árboles B. Un árbol terciario de búsqueda sería un árbol B de orden m=3 (siendo m el número máximo de hijos de cada nodo) y k=2 (siendo k el número de elementos que contiene como máximo cada uno de los nodos).

22.- Dado que interesa reducir al máximo la altura de un árbol B, ¿por qué no aumentamos "indefinidamente" el número de hijos del árbol? Razona la respuesta. Editada

El coste del acceso a bloques para localizar un registro en un fichero es, para un árbol b m-ario, de $O(\log_m(N))$. Al aumentar indefinidamente el número de hijos, aumentaríamos de forma indefinida el número de claves, aumentando el tamaño de cada nodo de forma también indefinida haciéndolos lentos de leer y procesar, superando los tamaños de los bloques de memoria, siendo necesario más de un acceso a memoria secundaria para leerlo. Usualmente el valor se coloca de forma tal que cada nodo ocupe un bloque de disco.

23- ¿Por qué interesa que los árboles B tengan poca altura?

El tener poca altura implica maximizar el número de nodos hijo de cada nodo interno, por lo que las operaciones para balancear el árbol se reducen, aumentando la eficiencia. Además, el tiempo de búsqueda en un árbol B es proporcional al número de niveles que posee.

24- Se insertan un conjunto de elementos en un árbol B en un determinado orden. ¿La altura de ese árbol B resultado es independiente del orden de inserción?

Si y no, si vamos a insertar un conjunto de elementos, y en el árbol hay claves libres donde puedan alojarse sin necesidad de promocionar, no hay problema, si no, la altura si se vería afectada.

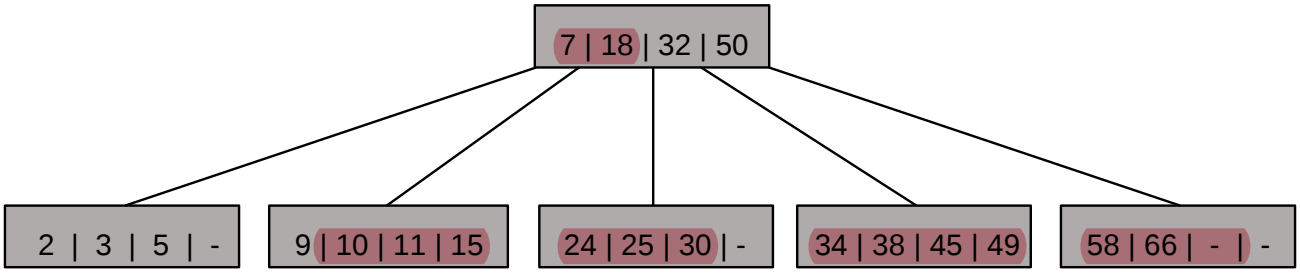
25- Explicar el por qué se exige un equilibrio perfecto en los AVL. -> *Examen 2014*

En los AVL no se exige un equilibrio perfecto; el factor de desequilibrio puede ser de 1, 0 o -1. Gracias a esta forma de equilibrio (o balanceo), la complejidad de una búsqueda en estos árboles se mantiene siempre en orden de complejidad O(log n).

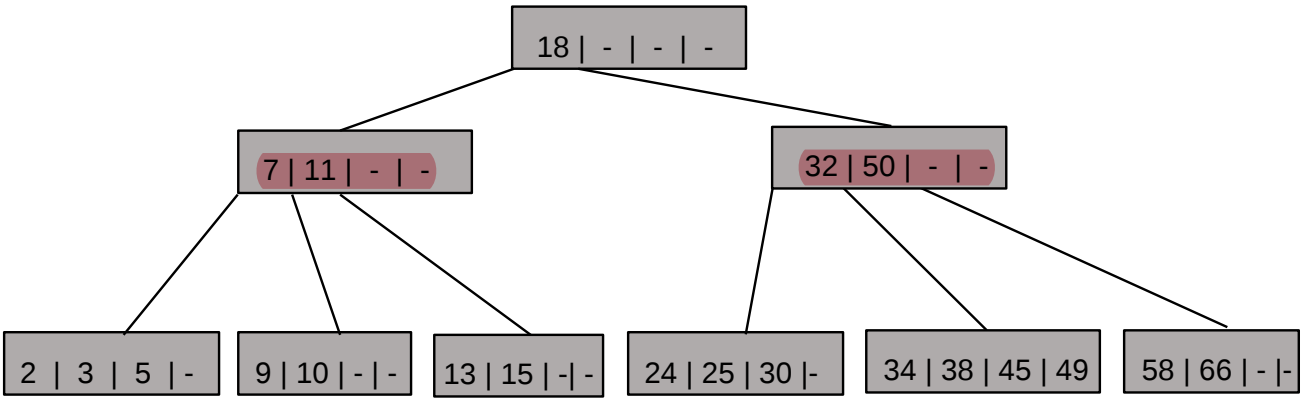
26.- ¿Por qué interesa reducir al máximo la altura de un árbol B? New.

Porque al maximizar el número de nodos hijo de cada nodo, haciendo disminuir la altura del árbol, las operaciones para balancearlo se reducen, y aumenta la eficiencia. Indicar que lo ideal es que un nodo ocupe un bloque de memoria, y el número de hijos sea consecuente a este tamaño.

27- Insertar en el siguiente árbol B el elemento 13, seguido del 39. -> *Examen 2015*



1º- Hacemos división 9 | 10 | 11 13 | 15 y promocionamos 11.
Luego realizaremos división 7 | 11 18 | 32 | 50 y promocionamos 18.



2º- Hacemos división 34 | 38 | 39 45 | 49 y promocionamos 39.