jvc93
www.wuolah.com/student/jvc93

2351

Preguntas%20ED2.pdf

Preguntas Teoría

2° Estructuras de Datos no Lineales

Grado en Ingeniería Informática

Escuela Superior de Ingeniería
UCA - Universidad de Cádiz

PREGUNTAS TEORÍA EXÁMENES

1.- ¿Es posible obtener coste O(n) en la eliminación de un nodo cualquiera de un APO?
En un APO o montículo no se puede eliminar cualquier nodo del mismo, sino que se elimina aquel que tiene mayor prioridad, es decir, la raíz del APO. O en el caso de un APO MIN-MAX también podría eliminarse (suponiendo que el montículo tenga un número de nodos mayor de 2), el mayor de los hijos de la raíz, pero en ningún caso obtendremos ese coste.

2.- ¿Es posible obtener coste O(n) en la inserción/eliminación de la raíz en un APO?
No, ya que un APO o montículo es siempre un árbol completo, es decir, si le faltan nodos serán del último nivel y por la derecha. Luego, por esta razón, es seguro que no se puede producir la degeneración del árbol en una lista y por lo tanto el coste de la inserción será $O(\log_2 n)$.

3.- ¿Qué aportan los AVL frente a los ABB?
La búsqueda en un ABB de n elementos requiere $O(\log_2 n)$ operaciones en el caso medio (árbol completo) y en el peor caso (árbol degenerado en lista) será de O(n) operaciones. Los AVL son árboles binarios de búsqueda en el cuál las alturas de los dos subárboles nunca difieren en más de una unidad, es decir, están equilibrados. Por ello se evita entrar en el peor caso y se asegura que tanto las búsquedas de elementos como las inserciones y eliminaciones de nodos se pueden efectuar en $O(\log_2 n)$.

4.- ¿Tiene sentido el concepto de un árbol terciario de búsqueda?
Sí, teniendo en cuenta que la generalización de los ABB para árboles generales son los denominados árboles B, un árbol terciario de búsqueda sería un árbol B de orden m = 3 y k = 2, siendo m el número máximo de hijos de cada nodo y k el número de elementos o claves que tiene como máximo cada uno de los nodos.

6.- ¿Qué condición tienen que cumplir los elementos de un árbol para poder realizar las búsquedas con un coste menor que O(n)?
Para realizar búsquedas en orden menor que O(n) es necesario que los elementos cumplan una relación de orden entre sí y que esté equilibrado.

7.- ¿Es siempre la eliminación de elementos en un ABB de orden mayor que O(n)?
No, nunca será mayor que O(n), ya que ese es el peor caso, cuando el árbol esté degenerado en una lista.

8.- ¿Existe alguna operación claramente ineficiente en los árboles generales representados mediante listas de hijos?
Utilizando la implementación de árboles generales mediante listas de hijos el acceso al hermano derecho de un nodo es poco eficiente, ya que el tiempo es proporcional al número de hermanos de un nodo. Luego HermDrcho (nodo n, Arbol A) resulta ineficiente.

15.- 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.
Sabiendo que 'm' es el número máximo de hijos que puede tener cada nodo del árbol, no podemos hacer 'm' arbitrariamente grande, puesto que si 'm' es demasiado grande, el tamaño de un nodo podrá exceder el tamaño de un bloque y será necesario más de un acceso a memoria secundaria para leer un nodo. Además es más lento leer y procesar un bloqué más grande, así que es necesario buscar el valor óptimo de 'm'.

16.- Las operaciones insertar y eliminar del TAD Binario representado mediante un vector, ¿son de O(1)?
En la primera idea sobre esta implementación, la inserción era poco eficiente, ya que había que buscar una celda cuyo padre fuese NODO_NULO, mientras que la eliminación sólo requería marcar la celda como libre colocando NODO_NULO.
Se mejoró de forma que todas las celdas libres del vector estaban al final, así se insertaría en la primera celda libre (ya que conocemos el número de nodos) en tiempo constante. A cambio, en la eliminación se tardaría un poco más de tiempo, ya que habría que mover todos los nodos para cubrir el hueco libre. Para solucionar esto, se moverá el último nodo a la posición donde se encuentra el hueco libre, quedando por tanto la operación de eliminación de orden constante también.

17.- APO, ¿influye el orden de inserción de los elementos?
Para que un APO siga siéndolo, debemos controlar siempre que se siga cumpliendo la condición que los caracteriza, es decir, que cualquier nodo debe ser menor o igual que sus descendientes, y que el último nivel (que es el único que puede no estar completo del todo) debe rellenarse de izquierda a derecha para que sea un árbol completo. Puede haber más de un elemento que cumpla estas condiciones y ocupen lugares distintos en el APO según los insertemos.

18.- ¿Influye el orden de inserción de los datos en la altura de un AVL?
La altura de los AVL depende del número de nodos que vayamos insertando no del orden en que se insertan los mismos. Para un mismo número de nodos, la altura de un AVL, se quedará igual o como mucho, aumentará en uno, según el orden de inserción de los elementos, ya que debe seguir siendo un AVL y cumpliendo por tanto la condición de equilibrio.

19.- ¿En qué situaciones es conveniente utilizar un vector de posiciones relativas?
Esta implementación es recomendable para árboles completos, ya sean binarios, terciarios, cuaternarios, etc., siempre que conozcamos el número de nodos.

20.- ¿Por qué interesa que los árboles B tengan poca altura?
Porque si nuestro árbol tiene poca altura, reduciremos el número de accesos a memoria.

24.- ¿Cuántos tipos de recorridos de árboles en anchura existen? Explica sus diferencias.
Existe un único recorrido de árboles en anchura (se procesan de izquierda a derecha todos los nodos de un nivel antes que los del siguiente, empezando por el primer nivel, es decir, por la raíz), sólo que se consideran dos versiones, una iterativa y otra recursiva. A diferencia de esto, existen tres tipos de recorridos en profundidad para árboles: preorden, inorden y postorden.

25.- ¿Es verdad que la inserción y la eliminación de nodos en un árbol binario no se puede conseguir a un coste O(1), cuando se utiliza una representación vectorial con índice al padre, hijo izquierdo e hijo derecho?
Sí se puede conseguir, mirar la explicación de la pregunta 16.

NOOOO!!!!!
26.- Si dispones de un soporte de almacenamiento direccionable, ¿en qué casos optarías por una organización secuencial para un fichero?
En aquellos casos en los que sea frecuente un acceso secuencial a los datos del fichero, ya que para este tipo de accesos, es la mejor organización, pero nunca para aquellos casos en los que sean frecuentes consultas puntuales, ya que en este caso, el acceso sería muy lento. También en aquellos casos en los que se precise aprovechar el espacio de almacenamiento, ya que la organización secuencial, tiene la ventaja de ocupar únicamente el espacio justo para los datos. QUITAR

27.- ¿Por qué se exige un orden exacto en la inserción de elementos en el AVL?
No se exige un orden exacto en la inserción, ya que el AVL resultante después de cualquier inserción seguirá cumpliendo las propiedades de los AVL. La operación Insertar de los AVL es la que se encargará de reorganizar el AVL si hubiese algún problema.

28.- ¿Influye el orden de inserción de los elementos para el desequilibrio de un APO?
El orden en que se insertan elementos, influye en el desequilibrio de los ABB, pero los APO son árboles completos, es decir árboles con todos los niveles llenos a excepción del último, que puede no estar lleno, pero que se rellena de izquierda a derecha. Luego serán árboles equilibrados, ya que ningún nodo tendrá un desequilibrio mayor que uno, y éstos serán los del penúltimo nivel si el árbol no está del todo completo.

29.- ¿Por qué surgen los árboles?
Surgen por la necesidad de representar situaciones de la vida cotidiana en las que los elementos están organizados en una jerarquía de diferentes capas o niveles, ya que las estructuras vistas anteriormente (pilas, colas y listas) únicamente servían para representar secuencialidad entre los elementos. Por otra parte, también solucionan un problema, solucionable mediante las estructuras de datos lineales, pero que no es computable en un tiempo razonable mediante esas estructuras, que es el problema de la búsqueda de un elemento en una colección de los mismos. Los árboles, permiten realizar búsquedas en orden logarítmico, mientras que con las estructuras lineales, las búsquedas eran de orden O(n).

30.- ¿Qué consiguen los árboles en la búsqueda?
Los árboles, permiten realizar búsquedas en orden logarítmico, mientras que con las estructuras lineales, las búsquedas eran de orden O(n).

31.- Las operaciones de insertar y eliminar en los árboles binarios, ¿cuándo son de orden O(1)?
Son de orden constante cuando se trata de la representación vectorial. Ver pregunta 16.

32.- Ventajas de la representación de árboles binarios con celdas enlazadas frente a la representación con matrices.

La representación mediante celdas enlazadas, es más eficiente en cuanto a espacio ocupado, ya que al estar implementada mediante punteros, siempre ocupará el espacio que necesite, mientras que la representación mediante matrices, puede no utilizar todo el espacio que tiene reservado para el árbol.
Otra ventaja puede ser la facilidad para moverse hacia arriba y hacia abajo en el árbol.

42.-¿Influye el número de elementos de un APO en su desequilibrio?
Sí ya que si todos los nodos hojas están en el último nivel el desequilibrio es 0 y en caso contrario 1.

43.- ventajas de la representación de árboles binarios con celdas enlazadas frente a matrices.
La representación de celdas enlazadas es necesaria cuando no se conocen el número máximo de nodos que puede tener el árbol representado a priori (puede faltar espacio). También es bastante más eficiente en aquellos casos en los que sabiendo el número máximo de nodos, no se utilicen todos o la mayoría de ellos frecuentemente, se desperdicie espacio.

44.- ¿por qué no se puede implementar un árbol general con un vector de posiciones relativas?
Porque el árbol general puede tener cualquier grado y en la representación con posiciones relativas las relaciones entre los nodos(Hijos/padre) van en relación del grado. si no se sabe el grado, ¿cómo se donde esta los hijos de un nodo?

41- ¿Se podría usar las listas doblemente enlazadas en los árboles generales mediante listas de hijos?
No, porque en el TAD no es necesario movernos hacia el hermano Izquierdo, ya que no lo contempla la especificación.

42. ¿A partir de dos recorridos cualesquiera conocidos, podemos conocer el Árbol ?
No, no vale cualquiera, sólo vale si uno de los dos es el recorrido en Inorden, el otro puede ser preorden o posorden.

43- ¿ Qué aporta los AVL a los ABB?
Aseguran que las operaciones de inserción () , eliminación() y búsqueda se realizan siempre en el peor caso en orden logarítmico.

44 - ¿puede tener coste O(n) eliminar raíz en un APO?
No, ya que para eliminar la raíz se intercambia en el vector con el elemento que ocupa la última posición del vector y se hunde y la operación hundir es de coste logarítmico.

45 - ¿condiciones debe tener un ABB para que la búsqueda sea menor que O(n)?
Que este equilibrado (AVL).

5.- ¿Por qué el algoritmo de Kruskall asegura que no se producen ciclos?
Porque el resultado de aplicar el algoritmo de Kruskall es un árbol generador (o de extensión) de coste mínimo y, por definición, un árbol es un grafo no dirigido conexo y acíclico. Por lo tanto, no se pueden producir ciclos.

9.- Dado el TAD Partición, decir qué combinación entre estructura de datos y estrategia es necesaria para que tanto la búsqueda como la unión sean de O(1).
No existe ninguna estructura de datos que haga posible que ambas operaciones sean de orden O(1) simultáneamente. Podemos optar por hacer que únicamente la búsqueda sea de O(1) utilizando para ello, por ejemplo, un vector de pertenencia como estructura de datos o hacer que únicamente la unión sea de O(1) usando para ello un bosque de árboles como estructura de datos y una estrategia de unión por tamaño o unión por altura, con lo que se conseguirá que la operación de búsqueda sea de orden O(log n) en el peor caso.
En todo caso, utilizando unión por altura o por tamaño y búsqueda por compresión de caminos, después de muchas ejecuciones podría ser que ambas hubiera conseguido tener un tiempo de O(1).

10.- Dado el algoritmo de Kruskall implementado mediante el TAD Partición, ¿son los mismos árboles los de la partición y los del algoritmo?
El TAD Partición se utiliza como una ayuda en la implementación del algoritmo de Kruskall. Una partición no es nunca un árbol aunque se represente mediante bosque de árboles. Sólo es cierto que los vértices de la partición serán los mismos que los que forman el árbol.

11.- ¿Es necesario ordenar las aristas en el algoritmo de Kruskall? ¿Sería correcto en caso de no ordenarlas?
Sí, es necesario ya que el algoritmo considera la lista de aristas del grafo en orden creciente de costes para crear un árbol de expansión de costes mínimos, y se coge la arista de menor coste de todas.

12.- ¿Por qué se coloca infinito en la diagonal principal de la matriz de costes en el algoritmo de Floyd?
La diagonal principal de la matriz de costes del algoritmo de Floyd no se inicializa a infinito sino que lo hace a cero, ya que así se representa que todo vértice v tiene un camino hacia sí mismo de longitud igual a cero. Luego A[i][i] = 0 para todo i.

13.- Explica las razones por las que no es necesario marcar los nodos visitados al realizar el recorrido de un grafo.
Precisamente es al revés, dado que no existe secuencialidad entre los nodos de un grafo (como en las listas) ni tampoco hay una jerarquía definida entre ellos (como en los árboles), nada nos impide que podamos entrar en un ciclo. Por tanto es necesario marcar de alguna forma cada nodo recorrido como visitado para evitar recorrerlo más de una vez.

14.- ¿Por qué surgen los grafos?
Por la necesidad de tener una estructura de datos cuyos nodos no estén organizados de manera secuencial (como en las listas) ni tampoco estén relacionados de manera jerárquica (como en los árboles). En este sentido, lo único que diremos es que es una estructura que conecta los nodos de una red mediante aristas.

21.- El algoritmo de Dijkstra, ¿funciona correctamente con valores negativos?
El algoritmo de Dijsktra, es un algoritmo voraz, es decir, se va quedando con la mejor solución hasta el momento. Si se permitiesen los costes negativos, podría ocurrir, que llegados a un punto de la traza del algoritmo, en el que tendríamos una solución concreta (se supone que la mejor hasta el momento), nos encontrásemos con un camino más corto, originado por estos costes negativos, lo cual sería incoherente ya que hemos dicho que se va quedando con la mejor solución, y si este camino fuese realmente mejor (más corto), ya lo habríamos cogido en un punto anterior de la traza del algoritmo. Además, con los costes negativos, podrían originarse ciclos.

22.- En el TAD Partición, ¿es posible emplear la unión en altura y la compresión de caminos a la vez?
Sí. Son dos técnicas que hacen cosas distintas. La unión por altura, realiza la operación Unión() de manera que el árbol con menos altura siempre será hijo del árbol con más altura, mientras que la compresión de caminos, lo que hace es que en cada búsqueda, al subir de nivel hacemos que el nodo por el que pasamos sea hijo de la raíz y así nos aproximamos a la solución ideal de dos niveles. Por tanto, se emplean las dos cosas a la vez, de hecho, el uso de estas dos técnicas de manera simultánea, se utiliza para conseguir después de muchas ejecuciones mejores tiempos para las operaciones Encontrar() y Unión().

23.- Comente la siguiente afirmación: "Prim y Kruskall resuelven el mismo problema y dan la misma solución".
La afirmación es cierta. Ambos algoritmos resuelven el problema de conectar todos los nodos de una red con un coste mínimo y dan como solución un árbol generador de coste mínimo. La diferencia está en el modo de hacerlo. Kruskall ordena desde el principo las aristas de menor a mayor coste y utiliza el TAD Partición, resultando por tanto más eficiente que el algoritmo de Prim en una constante multiplicativa. Sin embargo, Prim tiene que ir buscando las aristas de coste mínimo a lo largo del algoritmo.

33.- ¿Qué pasaría si el algoritmo de Prim y Kruskall operaran sobre un grafo dirigido?
Los algoritmos de Prim y Kruskall obtienen un árbol generador de coste mínimo para el grafo que comprenda todas las líneas posibles de comunicación de la red. Si el grafo es dirigido, entonces no considera los dos sentidos de circulación entre nodos, únicamente consideraría un sentido.

34.- ¿Por qué no se permiten los costes negativos en Floyd?
Porque podríamos encontrar un camino de un nodo hacia sí mismo con coste menor, lo cual es absurdo.

35.- Condición que debe cumplir un grafo no dirigido para que Kruskall obtenga el resultado.
Que el grafo además de no dirigido, sea conexo y no procesar dos veces el mismo nodo.

37.- ¿Por qué Kruskall no devuelve un grafo?

El resultado de aplicar el algoritmo de Kruskall es un conjunto de aristas que forman un árbol generador (o de extensión) de coste mínimo. Con este árbol generador de coste mínimo, tendremos la forma de conectar todos los nodos de una red con el menor coste.

38.-¿Existe una estructura de datos que permita ejecutar simultáneamente las operaciones de unión y encontrar en un tiempo constante en las particiones?
No.

39.-¿Qué se consigue con la técnica de unión por tamaño ?¿y con la técnica de unión por altura ?
Ambas técnicas aseguran un coste, en el peor caso, de log(n) para la operación encontrar.

40.-¿En la representación mediante bosques de árboles con unión por altura, por qué las raíces de los árboles se representan con números negativos?
Para diferenciar los representantes canónicos. Además el valor absoluto del número negativo representa la altura del árbol más 1.

41.- AVL ..... factor de equilibrio.
El factor de equilibrio o balance de un nodo se define como la altura del subárbol derecho menos la altura del subárbol izquierdo correspondiente. El factor de equilibrio de cada nodo en un árbol equilibrado será 1,-1 , 0.