AED-Enero-Junio-2017.pdf

albeerto_ro

Algoritmos y Estructuras de Datos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería
Universidad de Santiago de Compostela

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA — ETSE Escola Técnica Superior de Enxeñaría

ALGORITMOS Y ESTRUCTURAS DE DATOS

EXAMEN 2017

ALBERTO RUIBAL OJEA

1 EJERCICIO 1

Describe la estructura interna de la implementación de los nodos de un árbol binario de búsqueda ordinario, un árbol AVL, un árbol B y un árbol B+. ¿Cuál es la principal ventaja de un árbol AVL frente a un árbol de búsqueda ordinario? ¿Y de un árbol B+ frente a un árbol AVL?

Un ***ÁRBOL BINARIO DE BÚSQUEDA ORDINARIO (ABB)*** tiene punteros a sus hijos izquierdo y derecho y la información del nodo. La inserción es recursiva, ya que se parte del nodo raíz y se comprueba si el elemento a insertar es menor o mayor, bajando por la rama correspondiente hasta encontrar su sitio. Al eliminar un nodo, es necesario reestructurar el árbol:

- Si el nodo es hoja, se suprime del árbol sin más.
- Si el nodo tiene un único hijo, se intercambia posición con el hijo y se elimina del árbol.
- Si el nodo tiene dos hijos, se busca el menor de los descendientes del hijo derecho, se sustituye por dicho nodo y se elimina del árbol. Otra opción es sustituir el nodo de mayor valor de los descendientes del subárbol izquierdo.

En un ***ÁRBOL EQUILIBRADO (AVL)*** se tienen los mismos elementos que en el ABB, pero a mayores se le añade el factor de equilibrio del nodo para saber si es necesario reestructurar el árbol (almacena la diferencia de altura de cada subárbol). Será AVL mientras el valor absoluto de todos los factores de carga sea menor que 2. Para insertar, se inserta el nodo como un nodo hoja (factor equilibrio es cero) y se recalculan los factores de equilibrio de los nodos superiores hasta encontrar uno con valor absoluto 2, en donde se hace necesario reestructurar el árbol (rotaciones simples II, DD; rotaciones compuestas DI, ID). Para eliminar un nodo, se sigue el mismo algoritmo que para ABB, pero incluyendo las reestructuraciones necesarias.

En los ***ÁRBOLES B Y B+,*** los nodos son páginas a las que se acede en bloque y que contienen punteros a sus hijos (que pueden ser más de 2), las claves contenidas en la página y el número de huecos ocupados.

- En los B, al agrupar los nodos en páginas se reduce considerablemente la altura del árbol y ahorra bloques en el disco, realizando las búsquedas, inserciones y eliminaciones con mayor rapidez. Por otra parte, cuando hay subdivisiones o unificaciones se requieren más accesos a disco (aunque puedan ser menos que un convencional).
- En los B+, todos los elementos se encuentran en páginas hoja, por lo que algunos aparecen duplicados al actuar como índices. Ocupan algo más en memoria, pero suponen un aspecto positivo al evitar la reorganización del árbol.

La principal ***VENTAJA DE UN AVL FRENTE A UN ABB*** es que el primero garantiza una búsqueda binaria y el árbol siempre está equilibrado, en cambio en el caso de ABB en función del orden de inserción de los elementos el orden de complejidad aumenta (por ejemplo, si se intenta de menor a mayor sería O(n)).
Respecto a las ***VENTAJA DE UN B+ FRENTE A UN AVL***, árbol B+ genera menos nodos y un árbol con una altura menor al AVL, por tanto, el acceso a disco es menos costoso.

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en ing.es

Que te den **10 € para gastar** es una fantasía. ING lo hace realidad.

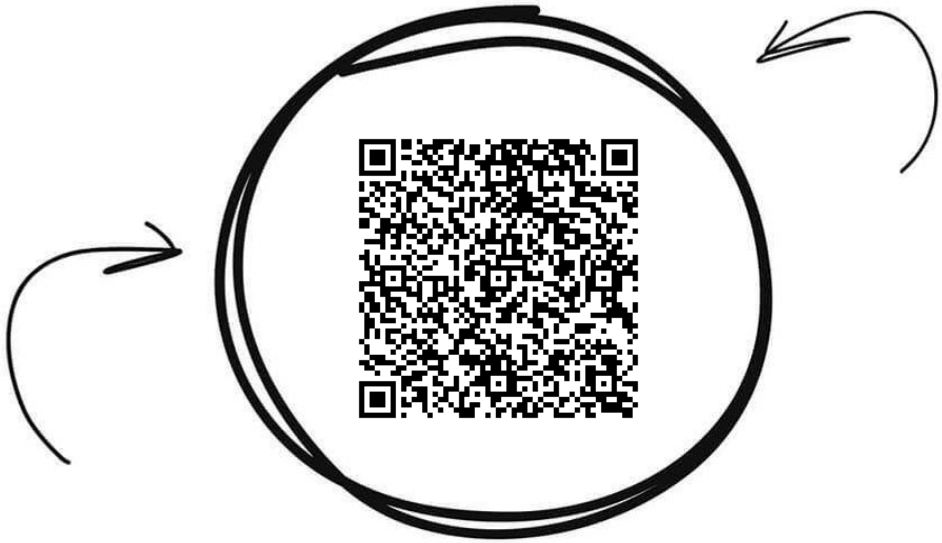
Abre la **Cuenta NoCuenta** con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Quiero el cash

Consulta condiciones **aquí**

do your thing

Algoritmos y Estructuras de...



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

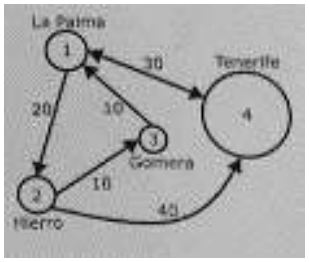
2 EJERCICIO 2

Explica cómo se ordenan los elementos en un montículo binario. ¿Para qué es útil una estructura de este tipo? Pon un ejemplo de aplicación.

En un montículo binario de mínimos los elementos se ordenan de forma que la clave de los hijos de un nodo es mayor o igual que la del padre (en caso de máximos sería menor o igual). Un ejemplo de uso sería para implementar colas de prioridad (como la gestión de enfermos en un servicio de urgencias) ya que siempre se saca el elemento de mayor prioridad (la raíz), por lo tanto, los nodos salen en función de su clave.

3 EJERCICIO 3

Observa el siguiente grafo de conexiones entre todas las islas de la provincia de Tenerife. Para cada conexión (que puede ser de dirección única), además de saber la isla de origen y la isla destino, se conoce el coste del viaje.



A. **La estructura interna de un grafo puede representarse de dos formas, mediante estructuras estáticas o mediante estructuras dinámicas. Indica en este caso cuál de las dos usarías y explica por qué.**
   Con una estructura estática (matriz de adyacencia), se obtiene eficiencia a la hora de obtener los costes asociados a un arco y la comprobación de adyacencia es inmediata, pero no se permite eliminar nodos del grafo y constan de muchos ceros.
   Con una estructura dinámica (lista de adyacencia), ocupan menos memoria, pero la representación es más compleja al contar con punteros y es ineficiente para encontrar los arcos que llegan a un nodo (hay que recorrer todas las listas buscando si en cada una hay un arco al elemento buscado).
   USARÍA UNA ESTRUCTURA ESTÁTICA debido a que el número de islas de Tenerife no va a cambiar.

B. **Para cada par de islas se desea saber cuál es el camino de coste mínimo que las une. Indica qué algoritmo usarías.**
   El algoritmo de Floyd, ya que devuelve una matriz donde cada campo contiene el coste mínimo de los caminos de i a j.

C. **En caso de que todos los arcos fueran bidireccionales, indica qué buscarías para encontrar los puntos de la red que, si fallan, producen un fallo general de la red de comunicaciones.**
   Algoritmo de búsqueda de puntos de articulación, ya que se realiza un recorrido en profundidad recursivo del grafo, que se representa con un árbol de expansión.

D. **En caso de que todos los arcos fueran bidireccionales, ¿Cómo podríamos saber si podemos eliminar alguna de las conexiones minimizando el coste? ¿Qué algoritmo aplicarías?**
   Generando un árbol de expansión de coste mínimo, ya que el peso de sus arcos suma el menor valor posible y se generan el menor número de caminos posible que unan todos los nodos.
   Algoritmo de Kruskal o algoritmo de Prim.

4 EJERCICIO 4

Decide las estructuras de datos más apropiadas (pilas, colas, listas, árboles, grafos) para simular los procesos que sigue un buscados de Internet para recopilar e indexar información de la Web, y la información para poder calcular el grado de autoridad de los sitios web.

A. La información de los sitios web encontrados por los robots. Esta tiene que guardarse temporalmente en memoria de la forma más eficiente posible teniendo en cuenta que los robots están enviando continuamente nueva información que se va procesando para organizarla e indexarla. Tenéis que determinar qué estructura de datos es la más apropiada para esto.

Una tabla hash debido a que es necesario insertar y buscar de forma eficiente sin necesidad de tener los elementos ordenados. Proporcionan un tiempo de búsqueda constante, independiente del número de elementos, aunque no estén ordenados.

B. La indexación de las páginas web. Hay que indexar los sitios web por orden alfabético de tal forma de que la búsqueda de estos sea lo más rápido posible. Hay que seleccionar una estructura de datos para crear este indexador de tal forma que se permita la introducción de sitios web nuevos conservando el orden alfabético y, si ya existen, actualizar el número de visitas que nuestros robots hayan realizado.

Un árbol binario de búsqueda (ABB, AVL, ***B***, B+, etc.) para que la búsqueda sea más eficiente y aun así se pueda mantener un orden.

C. Los enlaces entre sitios web. Los robots almacenan los enlaces que salen de cada sitio web en un fichero. Decide cuál es la estructura de datos más apropiada para guardar esta información de forma que permita poder calcular la autoridad de cada sitio web de forma sencilla.

En un grafo dirigido, para utilizar el grado de entrada/salida para reconocer los sitios web más referenciados.

5 EJERCICIO 5

Explica las ventajas y desventajas en cuanto a eficiencia (tiempo, almacenamiento) en las operaciones de inserción, búsqueda y ordenación cuando se usa una tabla hash, un vector o una lista enlazada para almacenar un número dado de datos N. Indica qué diferencias habría entre usar una tabla hash con recolocación y una tabla hash con encadenamiento.

Suponiendo que un vector es un array estático:

- En operaciones de inserción
  Las tablas hash nos proporcionan una inserción *O(1)*, en cambio las listas enlazadas y un vector en el peor de los casos es *O(n)* ya que hay que desplazar los bloques.
- Búsqueda
  Las tablas hash nos proporcionan una búsqueda *O(1)*, mientras que el resto en el peor de los casos *O(n)* (recorrer toda la lista en el caso del vector).
- Ordenación
  Una tabla hash no se puede ordenar debido a que su posición está establecida a través de una función matemática. La lista enlazada en el peor de los casos es *O(n)*, en cambio en un vector es *O(n)* y probablemente el algoritmo de ordenación prefiere poder acceder en *O(1)*.

En una tabla hash con recolocación sería necesario tener una tabla 2N (buscando un primo cercano) para tener un factor de carga 0.5. En cambio, con encadenamiento llegaría con una que sea 4/3 para tener un factor de carga de 0.75.

6 EJERCICIO 6

Dado el esquema general de un algoritmo voraz, indica qué representarían C, S, x y las funciones en negrita en el caso del algoritmo de Kruskal.

```
voraz(C: CjtoCandidatos; var S: CjtoSolucion)
      S := ∅
      mientras (C ≠ ∅) && NO solución(S) hacer
            X := seleccionar(C)
            C := C – {x}
            si factible(S, x) entonces
                  insertar(S, x)
            finsi
      finmientras
      si NO solución(S) entonces
            devolver “No se puede encontrar solución)
      finsi
```

C *es el conjunto de aristas que no se han seleccionado*.
S *es el conjunto de aristas solución*.
X *es la arista escogida de C que puede ser añadida a S si es factible*.

Solución(S): Función que comprueba si un conjunto de candidatos S es una solución al problema.
Seleccionar(C): Función que selecciona el mejor elemento de un conjunto C de candidatos.
Factible(S, x): Función que indica si a partir de un conjunto de candidatos S, y añadiendo otro x es posible llegar a una solución.
Insertar(S, x): Función que añade un elemento x al conjunto S de candidatos seleccionados para la solución.
Objetivo(S): Función que dada una solución S devuelve el coste asociado a la misma.

7 EJERCICIO 7

Explica el proceso de exploración del espacio de solución de la estrategia de ramificación y poda en un problema de optimización el que ha sido utilizado para definir la solución óptima es la minimización de costes.

Hay dos estrategias para la exploración del espacio de búsqueda.

- ***DE PODA***
  En cada nodo, a partir de una solución parcial, se generan cotas inferiores y superiores del beneficio que se puede obtener siguiendo por los hijos del nodo, pudiendo podar si no es óptimo.
  Por ejemplo, si estamos en un caso de maximización y tenemos almacenada en una variable de poda C la CI más grande que se ha obtenido hasta el momento, si el CS del nodo es menor que C pues no tiene sentido seguir por el nodo. Por lo tanto, se poda.

- ***DE RAMIFICACIÓN:*** nos especifica el tipo de recorrido
  Generalmente se usa una lista de nodos visitados para representar los nodos que han sido generados, pero no explorados.
  Hay varios tipos de recorridos como por profundidad, anchura, etc.
  Estrategias de ramificación:
  - *LC-FIFO*: Seleccionar de LNV el nodo que tenga menor coste estimado, y en caso de empate escoger el primero que se ha introducido.
  - *LC-LIFO*: Seleccionar de LNV el nodo que tenga menor coste estimado, y en caso de empate escoger el último que se ha introducido.
  - *MB-LIFO*: Seleccionar de LNV el nodo que tenga mayor beneficio estimado, y en caso de empate escoger el primero que se ha introducido.
  - *MB-LIFO*: Seleccionar de LNV el nodo que tenga mayor beneficio estimado, y en caso de empate escoger el último que se ha introducido.