Desorden Acotado Optimo*

Ricardo A. Baeza-Yates
Depto. de Ciencias de la Computación
Universidad de Chile
e-mail: rbaeza@dcc.uchile.cl
Blanco Encalada 2120
Santiago, Chile

Héctor Soza Pollman
Depto. de Ingeniería de Sistemas y Computación
Universidad Católica del Norte
e-mail: hsoza@ucn.cl
Av. Angamos 0610, Casilla 1280
Antofagasta, Chile

Resumen

En este trabajo presentamos, para la organización de archivo Desorden Acotado (*Bounded Disorder*), la determinación de un modelo que relaciona el tamaño del bucket de overflow secundario (medido en cantidad de registros que este bucket puede almacenar) con el tamaño de los buckets primarios del nodo de datos (supuesto distinto al de overflow) y con la cantidad de estos buckets, considerando el efecto del índice. Este modelo se obtiene al minimizar el costo de inserción de datos en la estructura, y permite determinar el tamaño óptimo del bucket de overflow. **Keywords:** Bounded Disorder, B-trees, Hashing.

1 Introducción

La organización de archivo de Desorden Acotado o Bounded Disorder (BD) en inglés, propuesta originalmente por Litwin y Lomet [2], se considera una estructura que provee un acceso eficiente a los datos almacenados, debido a que reduce la altura del índice integrando el acceso a ellos a través del uso de hashing [5]. Un archivo BD consiste de un índice de árbol B (*B-tree*), en que cada nodo de datos está organizado como una tabla de hashing de m buckets ($m > 0$) con una capacidad de almacenar b registros (o claves) cada uno y un bucket de overflow adicional, que se supondrá con una capacidad de c registros. Esta estructura se muestra en la Figura 1 y combina las ventajas de acceso de hashing con la posibilidad de realizar acceso secuencial ordenado de los árboles B.

Para insertar una nueva clave se usa el índice del árbol-B para hallar el nodo de datos correspondiente, y se usa una función de hashing para determinar el bucket adecuado e insertarlo. Si éste está lleno se intenta su inserción en el bucket de overflow, y si éste también está lleno se divide el nodo de datos en dos nodos de datos, lo que se llama una división (split) [2]. Un primer análisis de la eficiencia de esta estructura lo realiza Lomet [3], sobre una variante de esta estructura, el cual es mejorado por Ramakrishna y Mukhopadhyay [4] considerando un método alternativo para el crecimiento de los datos del archivo, pero sólo considerando la conducta de un nodo de datos. Baeza-Yates [1] completa este análisis introduciendo el índice en el modelo, y usando el *fringe analysis* [6]. En todos estos casos se ha

*Este trabajo ha sido parcialmente financiado por el Proyecto FONDECYT No. 1990627.

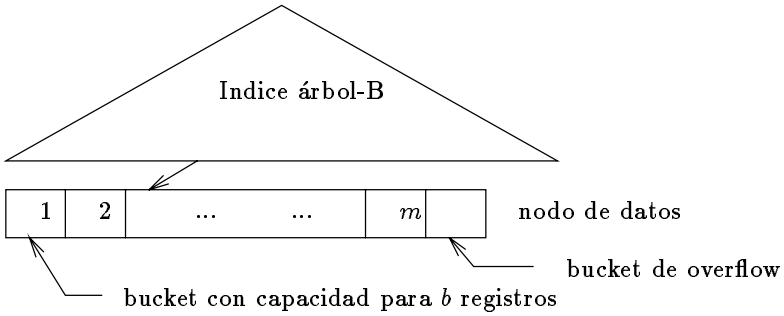

Figura 1: Ejemplo de Desorden Acotado.

considerado que los datos se almacenan en los nodos hojas de un árbol-B, formados por $m+1$ buckets primarios contiguos, de los que el último es el bucket de overflow, todos del mismo tamaño (b registros por bucket). En este artículo presentamos un estudio para determinar una relación óptima entre el tamaño del bucket de overflow c, el del bucket primario b y la cantidad m de buckets primarios minimizando el costo de inserción de datos en la estructura. En la estructura original $c = b$. Se considera que la unidad de espacio es la necesaria para almacenar la clave (llave) primaria de la base de datos más la información adicional asociada a cada elemento. En la Sección 2 se describen brevemente los resultados más importantes del método BD. En la Sección 3 se describe el análisis desarrollado y sus resultados, para luego presentar las conclusiones y extensiones.

2 Desorden Acotado

En lo que sigue se usará la misma notación de [1]. Sea $2L$ la menor cantidad de registros en un nodo de datos en el momento en que una división puede ocurrir y $H \leq 2L$ la mayor cantidad posible. Luego se tienen nodos de datos que tienen entre L y H claves. Para la versión actual de BD se observa que $L = \frac{b+c}{2}$ y $H = mb + c$. Sea s_j la probabilidad de una división dado que hay j claves en un nodo de datos, de modo que $s_j = 0$ para $j < 2L$ o $j > H$, y $s_j > 0$ para $j \geq 2L$ y $j \leq H$. Además $s_H = 1$. Sea $p_j(n)$ la probabilidad de hallar una clave en un nodo que contiene j claves, cuando hay n claves insertadas en el archivo. Asintóticamente sobre n, estas probabilidades p_j convergen a la solución del siguiente sistema de ecuaciones lineales, en que $p_L(L) = 1$ y $p_j(L) = 0$ para $j \neq L$:

$$(j+1)p_j = j(1 - s_{j-1})p_{j-1} + js_{2j-2}p_{2j-2} + 2js_{2j-1}p_{2j-1} + js_{2j}p_{2j}$$

con $j = L, ..., H$ y $p_j = 0$ para $j < L$ o $j > H$ (ver [1]). Entonces, la probabilidad de insertar una clave en un bucket de overflow es:

$$Pr_{Overf}(n) = \sum_{j=2L}^{H} p_j f_j^{m,b,c}$$

Las funciones $f_j^{m,b,c}$ son las probabilidades de insertar una clave en un bucket de un nodo de datos que está lleno. Estas probabilidades están determinadas en [4] y han sido modificadas considerando un tamaño c para el bucket de overflow, además de los parámetros j, m y b ya conocidos. La probabilidad de tener una división durante una inserción es:

$$Pr_{Split}(n) = \sum_{j=2L}^{H} s_j p_j$$

La utilización de almacenamiento esperada en los nodos de datos es:

$$U(n) = (H \sum_{j=L}^{H} p_j/j)^{-1}$$

Se desea determinar una relación entre el tamaño del bucket primario b, el del bucket de overflow c y la cantidad de buckets primarios m. Para ello se intentó determinar esta relación optimizando algunas de estas medidas de desempeño. Al intentar minimizar la utilización de almacenamiento de la estructura se observó que $c/b \to \infty$, lo cual conduce a que se alcance para la utilización el valor de $\ln(2) = 0.6931$, como en los árboles B^+, pero no se determina ninguna relación óptima para los parámetros mencionados. Se consideró minimizar el tiempo de búsqueda de un dato en la estructura, pero no se alcanzó un óptimo debido a que los valores de esta medida son siempre crecientes para los distintos parámetros considerados. Al considerar el costo de inserción, y tomando en cuenta que esta medida considera el tiempo de transferencia, si se logró determinar una relación c/b óptima para esta medida de desempeño, como función de los parámetros b, c y m, como se describe a continuación.

3 Costo de Inserción

En lo que sigue supondremos que:

- el nodo de datos entero puede leerse y escribirse en un acceso (es decir, hay contiguidad física).
- el índice del archivo BD está en memoria principal (representa menos del 1% de los datos).
- el tiempo de transferencia, dado por $tt(registros) = \frac{registros}{R}$ se expresa en unidades de tiempo para acceder a un bucket. Si A es el acceso a un bucket (ms), RT la razón de transferencia (MB) y TM el tamaño del registro (KB), entonces: $R = \frac{A*RT}{TM}$. Se verifica que $10 \leq R \leq 160$ para casos reales (ver [1]).

El costo de inserción $I_c(n)$ es el número de accesos a un nodo de datos durante una inserción, y se define como:

$$\begin{aligned} I_c(n) = 2 \ & (1 - Pr_{Overf}(n) - Pr_{Split}(n))(1 + tt(b)) \\ + \ & (Pr_{Overf}(n) - Pr_{Split}(n))(1 + tt(b) + 2(1 + tt(c))) \\ + \ & Pr_{Split}(n)(1 + tt(b) + 1 + tt(c) + 3(1 + tt(mb + c))) \end{aligned}$$

La primera línea de la ecuación anterior corresponde al costo de insertar en un nodo de datos que aún tiene espacio libre (sin considerar overflow ni división del nodo), lo que se hace con dos accesos: uno para leer el bucket y otro para escribirlo con la nueva información). La segunda línea corresponde a la inserción en el bucket de overflow (sin considerar una división del nodo) y considera el acceso al bucket primario más los accesos al bucket de overflow para almacenar el dato. Finalmente, la tercera línea considera la inserción posterior a la división del nodo de datos, la cual combina los accesos al nodo completo y la escritura de uno de los dos nuevos nodos de datos. Se supone que el índice está en memoria principal asi que no se necesita ningún acceso extra para actualizar el correspondiente nodo índice. Simplificando esta expresión se obtiene:

$$I_c(n) = 2 + 2tt(b) + Pr_{Overf}(n)(1 - tt(b) + 2tt(c)) + Pr_{Split}(n)(3tt(mb + c) - tt(c) - 2tt(b))$$

Esta expresión tiene un mínimo ya que el tiempo de transferencia aumenta con b al mismo tiempo que las probabilidades de overflow o split disminuyen. Para los cálculos se consideraron los valores de $R = 10$ y $R = 20$ respectivamente. A medida que éste valor crece disminuye la posibilidad de alcanzar un mínimo para $I_c(n)$. La Figura 2 muestra un ejemplo del mínimo para $I_c(n)$, considerando $R = 10$. A la izquierda está el caso $b = 5$ y a la derecha $b = 10$.

La Figura 3 muestra la relación entre $I_c(n)$ y m para $b = 5, 10, 15$ y la Figura 4 la relación entre $I_c(n)$ y b para $m = 5, 10, 15$, en ambos considerando $R = 10$ y $R = 20$, respectivamente. Es importante observar que, debido a que la resolución exacta de los sistemas de ecuaciones lineales mencionados en el Capítulo 2 depende del tamaño de éste, a medida que el valor de m o b crecían eran menos los valores de $I_c(n)$ calculables. Por ejemplo para $b = 5$ se alcanza hasta $m = 50$ pero para $b = 15$ sólo se puede calcular hasta $m = 26$. Análogamente para el caso en que m está fijo. En la Figura 3 se observa que los

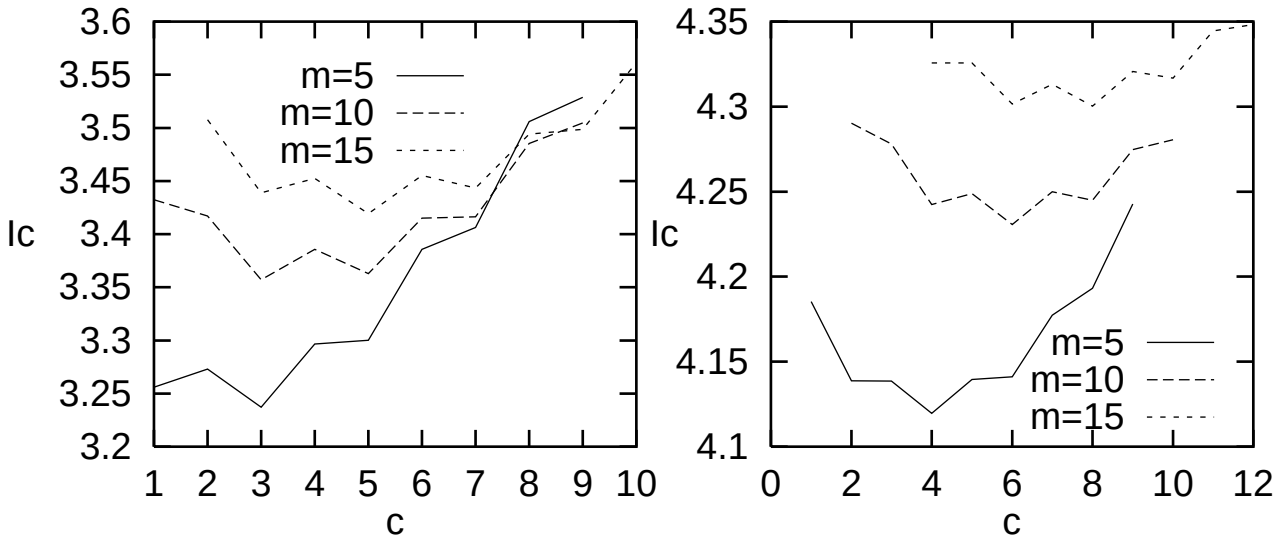

Figura 2: $I_c(n)$ versus c con $R = 10$ y $b = 5$ (izq.) y $b = 10$ (der.).

valores óptimos de $I_c(n)$ crecen lentamente con m para valores fijos de b, y se observa una convergencia hacia un valor fijo de $I_c(n)$ para m grande. En la Figura 4 en cambio, se observa que hay un crecimiento definido (casi lineal) de $I_c(n)$ con respecto a b para valores fijos de m y todos en el mismo rango de valores, de lo cual se deduce que para un valor de R dado, e independiente del parámetro m, los valores de $I_c(n)$ son crecientes e iguales para los valores de b considerados. A partir de los resultados obtenidos, fijando el parámetro b se obtuvo la relación entre c y m en Figura 5, y fijando el parámetro m se obtuvo la relación entre c y b en Figura 6. Se observa que en estas figuras aparecen discontinuidades crecientes a medida que m crece (ver Figura 5) y menores a medida que b crece (ver Figura 6). Se desarrolló un ajuste para los datos obtenidos, que relaciona c con m y b. Este ajuste se obtuvo usando mínimos cuadrados para relaciones entre tres parámetros y considerando todos los datos obtenidos. Para $R = 10$ el ajuste es:

$$c = 0.410794 m^{0.65078} b^{0.500547}$$

y para $R = 20$ es:

$$c = 0.162068 m^{0.81049} b^{0.624575}$$

Los ajustes mencionados se presentan en las Figuras 5 y 6 considerando un rango de valores entre 5 y 50 para m y b respectivamente, lo cual permite proyectar la relación existente entre estos parámetros en dicho rango. Se observa en las Figuras que estos ajustes representan una aproximación razonable de las relaciones obtenidas para c versus m y versus b respectivamente en los rangos considerados.

4 Conclusiones y Trabajos Futuros

Se analizó la estructura de archivo Desorden Acotado, para determinar una relación entre el tamaño del bucket de overflow c, la cantidad de buckets primarios m y la cantidad de registros b que estos buckets pueden almacenar, minimizando el costo de inserción de los datos en la estructura. Con los resultados se graficó la relación entre el costo de inserción y los parámetros m y b mencionados, manteniendo el otro fijo así como se graficó la relación entre c y cada uno de estos parámetros. Se estableció además un ajuste experimental de los datos obtenidos, que representa una buena aproximación para la relación entre los tres parámetros considerados. Como trabajo a futuro se propone desarrollar el mismo estudio considerando dos o más expansiones parciales de los nodos de datos en el momento en que el bucket de overflow se completa, como se propone en [2, 1].

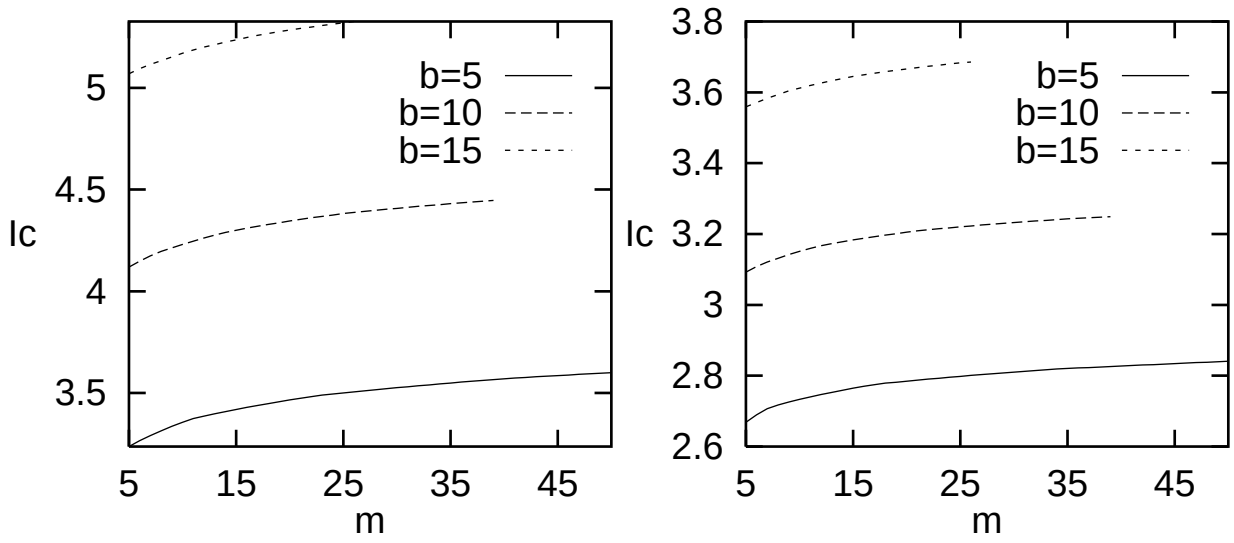

Figura 3: $I_c(n)$ versus m con b fijo para $R = 10$ (izq.) y $R = 20$ (der.).

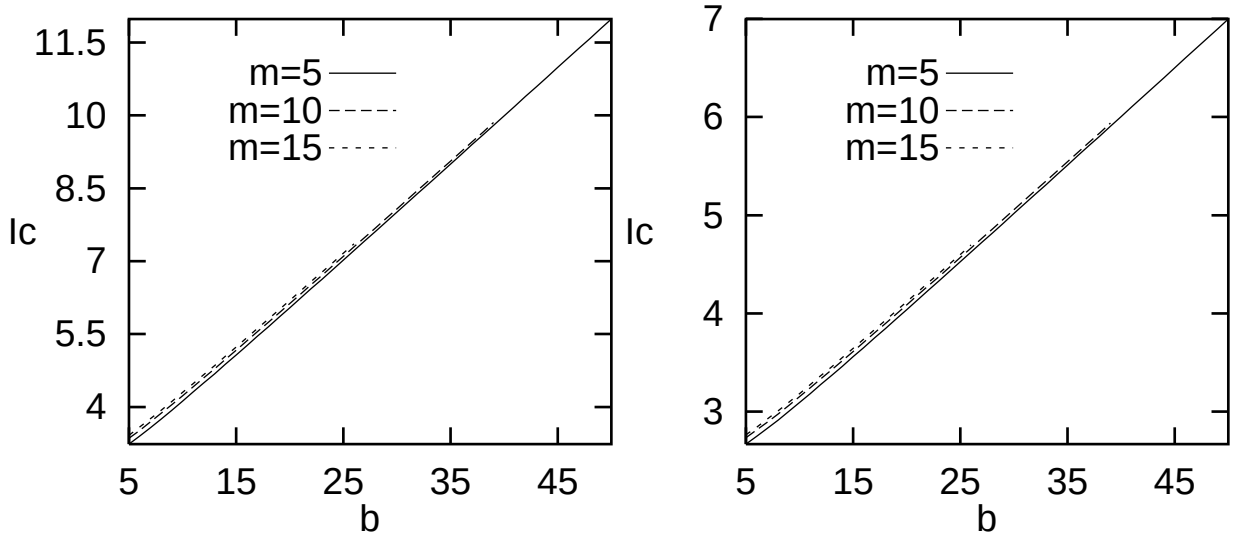

Figura 4: $I_c(n)$ versus b con m fijo para $R = 10$ (izq.) y $R = 20$ (der.).

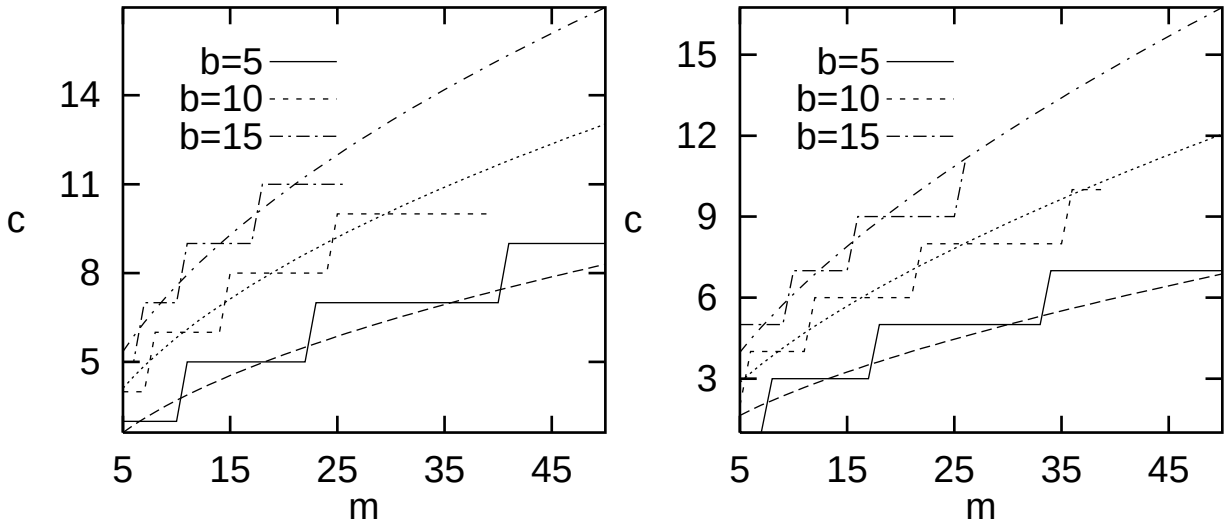

Figura 5: c versus m con b fijo para $R = 10$ (izq.) y $R = 20$ (der.).

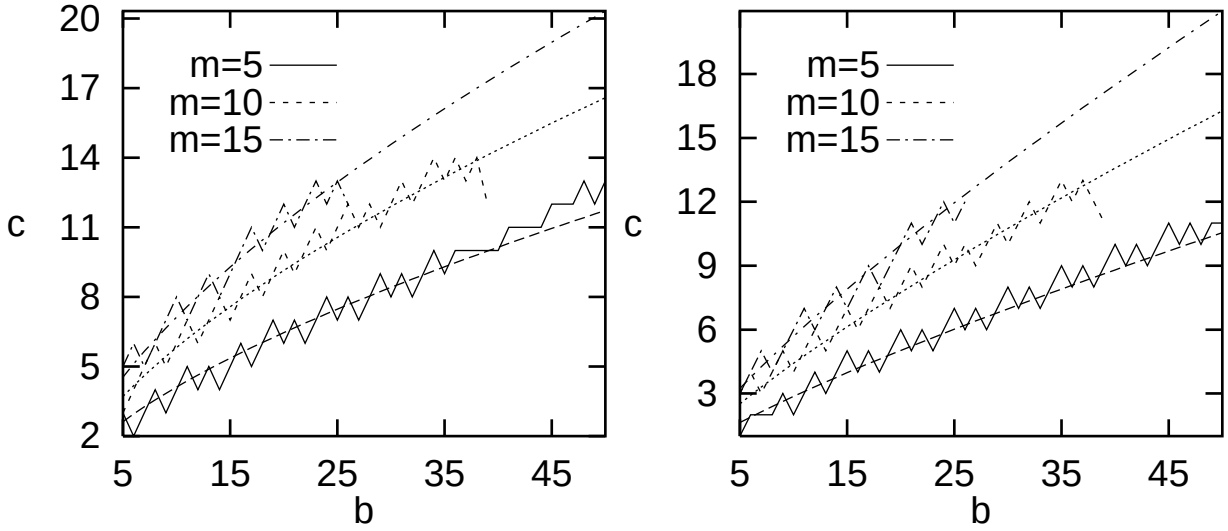

Figura 6: c versus b con m fijo para $R = 10$ (izq.) y $R = 20$ (der.).

Referencias

[1] R.A. Baeza-Yates, *Bounded disorder: the effect of the index,* Theoretical Computer Science **168** (1996) 21-38.

[2] W. Litwin, D. Lomet, *A new method for fast data searches with keys,* IEEE Software **4** (1987) 16-24.

[3] D. Lomet, *A simple bounded disorder file organization with good performance,* ACM TODS **13** (1988) 525-551.

[4] M. Ramakrishna, P. Mukhopadhyay, *Bounded Disorder file organization,* IEEE Trans. Knowledge Eng. **6** (1994) 79-85.

[5] A. Tharp, W. Boswell, B^+Trees, bounded disorder and adaptive hashing, Information Systems **16** (1991) 65-71.

[6] B. Eisenbarth, N. Ziviani, G. Gonnet, K. Mehlhorn and D. Wood, *The theory of fringe analysis and its application to 2-3 trees and B-trees,* Inform. and Control **55** (1982) 125-174.