Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 4 - "Algoritmos en Memoria Secundaria"

Profesores: Pablo Barceló
Gonzalo Navarro
Auxiliar: ~~Manuel~~ Ariel Cáceres Reyes

9 de abril del 2018

P1. Relación

Dada una relación binaria $\mathcal{R} \subseteq \{1, \ldots, k\} \times \{1, \ldots, k\}$, representada como un archivo secuencial de sus pares (i, j), construya algoritmos eficientes (considerando que $N \gg M$) para determinar si la relación es:

a) Refleja

b) Simétrica

P2. Desordenar

Se nos pide desordenar un archivo ordenado de números reales. Teniendo una función `rand(t)` que nos da un entero aleatorio uniforme entre 1 y t, debemos leer un archivo de disco de largo N y producir otro de largo N, donde se encuentren los mismos elementos permutados.
Su programa debe producir cualquier permutación con la misma probabilidad (para simplificar, no describa cómo permutar uniformemente en memoria interna).

a) Resuelva el problema en $\mathcal{O}(n)$ I/Os para el caso en que $N \leq M^2/B$.

b) Resuelva el problema en el mismo tiempo de ordenar en disco, para el caso general.

P3. Join y Split en B-tree

a) La operación de `join` de B-trees toma como entrada dos B-trees T_1 y T_2 , y una llave x tal que toda llave en T_1 es menor que x y toda llave en T_2 es mayor que x, y entrega como resultado un B-tree contiene a x y a todas las llaves en T_1 y en T_2.
Explique cómo implementar la operación de `join` entre T_1 y T_2 utilizando a lo más $\mathcal{O}(1 + |h(T_1) - h(T_2)|)$ accesos a memoria secundaria, donde $h(T)$ representa la altura de la raíz de T.

b) La operación de `split` de B-trees toma como entrada un B-tree T, y una llave x en T , y entrega como resultado dos B-trees T_1 y T_2 tal que T_1 contiene todas las llaves en T que son estrictamente menores que x y T_2 contiene todas las llaves en T que son estrictamente mayores que x.
Explique cómo implementar esta operación con $\mathcal{O}(B \cdot h(T))$ accesos a memoria secundaria.

"Someone is sitting in the shade today because someone planted a tree a long time ago. "
Warren Buffett

Soluciones

P1. Relación

Para esta pregunta asumiremos que no se repiten pares.

a) Basta tener un contador iniciado en 0, escanear todo el input bloque por bloque y cuando encontramos un par de la forma (i, i) aumentamos el contador. Cuando terminamos de escanear respondemos que la relación es refleja si el contador es igual a k, no en caso contrario. Como solo escaneamos el input una vez hacemos $n = \frac{N}{B}$ llamadas a disco. ⚡

b) Con el siguiente algoritmo:

- Hacer una copia de los pares, pero con sus coordenadas invertidas. $\mathcal{O}(n)$.
- Ordenar tanto el original como la copia lexicográficamente (es decir, por la primera coordenada y en caso de empate mirar la segunda coordenada para decidir). $\mathcal{O}(n \log_m n)$, con $m = \frac{M}{B}$.
- Comparar que tanto la copia como el original sean iguales. $\mathcal{O}(n)$

Por lo tanto, el costo del algoritmo es $\mathcal{O}(n \log_m n)$ llamadas a disco.

P2. Permutar

a)
1. Haremos operaciones sobre N/M archivos (las operaciones serán hechas con buffers de tamaño $\leq B$ por lo que ocuparemos $\leq BN/M \leq M$ memoria principal).
2. Al azar distribuimos uniformemente los elementos del input a los archivos (M elementos en cada archivo).
3. Por cada archivo: lo leemos, lo permutamos en memoria principal y lo volvemos a escribir a memoria secundaria.
4. Finalmente, vamos eligiendo al azar un archivo (dejamos de considerarlos si quedan vacíos) y sacamos el primer elemento de el y lo ponemos en el output.

b) Por la restricción ya no podemos procesar N/M archivos al mismo tiempo, por lo que solo lo haremos con M/B archivos y resolveremos los subproblemas que quedan recursivamente. Así para permutar un archivo de tamaño N haremos lo siguiente:

1. Haremos operaciones sobre M/B archivos (las operaciones serán hechas con buffers de tamaño $\leq B$ por lo que ocuparemos $\leq BM/B \leq M$ memoria principal).
2. <Mismo que en parte anterior>.
3. Por cada archivo: lo permutamos recursivamente.
4. <Mismo que en parte anterior>.

Así obtenemos la ecuación de recurrencia $T(N) = mT(N/m) + \mathcal{O}(n)$ para las operaciones en memoria secundaria, cuya solución es $\mathcal{O}(n \log_m n)$.

"Someone is sitting in the shade today because someone planted a tree a long time ago. "
Warren Buffett

P3. Split y Merge en B-tree

Para resolver esta pregunta agregaremos a los nodos del B-tree su altura, lo que no tiene costo adicional en operaciones de memoria secundaria, pues estos árboles *crecen por arriba* (cuando creamos una nueva raíz su altura es igual a 1 más la altura de sus hijos).

a) El enfoque general será hacer *merge* de dos nodos de T_1 y T_2 igual altura. En caso que tengan la misma altura, estos nodos serán las raíces de T_1 y T_2.Por otro lado, si $h(T_1) < h(T_2)/h(T_1) > h(T_2)$ los nodos serán la raíz de T_1/T_2 y el hijo más izquierdo/derecho de T_2/T_1 de altura $h(T_1)/h(T_2)$. Al hacer el *merge* de ambos nodos, si nos quedan más de B elementos en el nodo, hacemos un *split* basado en la mediana de los elementos y esta mediana sube recursivamente en el árbol, como sucede en la inserción normal de B-trees. El costo en operaciones de disco será entonces el de encontrar los nodos de igual altura $\mathcal{O}(|h(T_1) - h(t_2))$ más el de posiblemente hacer *split* de la raíz del árbol más alto $\mathcal{O}(1)$.

b) Primero buscamos x en el B-tree a costo $\mathcal{O}(h(T))$, luego convertimos al padre de la hoja donde se encuentra x en T_1 y T_2 respectivamente, en este momento de altura 1. Al devolvernos de la búsqueda vamos haciendo `join` de T_1 con cada uno de los árboles que están a la izquierda del camino que se tomó para hacer la búsqueda de x y de T_2 con los árboles que están a la derecha del camino. Finalmente, notando que se hacen $\mathcal{O}(B)$ `join` por nivel y que los `join` se realizan entre árboles cuya diferencia de altura es a lo más 1, el costo total de este `split` será $\mathcal{O}(B \cdot h(T))$.

"Someone is sitting in the shade today because someone planted a tree a long time ago. "
Warren Buffett