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

Auxiliar 3 - "Memoria Externa"

Profesores: Pablo Barceló
Gonzalo Navarro
Auxiliar: Dustin Cobas

P1. Relación Binaria

Dada una relación binaria $\mathcal{R} \subseteq \{1, \ldots, k\} \times \{1, \ldots, k\}$, representada como una secuencia de sus pares (i, j) donde asumiremos que no hay pares repetidos. Considerando que la relación $\mathcal{R}$ es almacenada en un archivo en memoria externa y $N \gg M$, construya algoritmos eficientes para determinar si $\mathcal{R}$ es:

a) Reflexiva.

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.

Soluciones

P1. Relación Binaria

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 $n = \frac{N}{B}$ accesos a la memoria externa.

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 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 $\mathcal{O}(n \log_m n)$ accesos a la memoria externa.

P2. Desordenar

a) Teniendo en cuenta la condición $N \leq M^2/B$, creamos $m = M/B$ archivos en memoria externa, cada uno con tamaño $\leq M$. Por cada archivo tendremos un buffer de tamaño B en memoria principal. Distribuimos aleatoriamente los N elementos del input entre los m archivos, de manera que cada archivo se considerará en la repartición siempre que no haya alcanzado los M elementos. Luego, leemos cada uno de los archivos, lo permutamos y lo reescribimos en memoria externa. Dado que los elementos del input ya están distribuidos uniformemente entre los archivos creados, la salida será la concatenación de los mismos.

En términos de acceso a la memoria externa, nuestro algoritmo solo hace unas pocas pasadas lineales por los datos, teniendo $\mathcal{O}(N/B) = \mathcal{O}(n)$ operaciones de acceso a memoria externa.

b) En el caso general, al dividir el input en m archivos, estos pueden tener tamaño $> M$, por lo que la fase de cargar cada archivo en memoria y permutarlo no sería posible. Podemos resolver este problema aplicando el mismo procedimiento a cada uno de los m archivos, obteniendo así m^2 archivos, y repitiendo este paso recursivamente hasta obtener m^i archivos con tamaño $\leq M$, los cuales pueden ser permutados en memoria principal. Nuestro algoritmo genera entonces un árbol de recursión de altura $i = \log_m n$, ejecutando $\mathcal{O}(n)$ accesos a memoria externa por nivel, por lo que la complejidad total es igual a la de ordenar en disco.