Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC4102 Diseño y Análisis de Algoritmos
Prof: Gonzalo Navarro B.

Diccionarios en Memoria Secundaria
Tarea 2
CC4102 - Diseño y Análisis de Algoritmos

Nicolás Salas V.
Daniel Rojas C.

15 de noviembre de 2015

Índice general

1. Introducción

El objetivo de esta tarea -y por tanto de este informe- es estudiar estructuras de datos en memoria secundaria.

En el contexto de esta tarea, con memoria secundaria se quiere decir esencialmente *disco*[1]. Esto es, sabiendo que algunas estructuras de datos son mejores para disco que otras, lo que interesa es verificar:

1. Si efectivamente son buenas estructuras para disco.

2. Dar una medida del espacio de las estructuras ocupan en disco.

Antes de continuar, es bueno dejar en claro algunas cosas y supuestos que pueden parecer obvios, pero siempre es mejor aclarar. Muchas de las cosas que en este informe se explican se toman por sentadas muchas veces sin dar prueba de ello. Dichas pruebas escapan al alcance de este informe, pero el hecho de declarar algunos resultados sin sus correspondientes pruebas no hace que sean menos verdaderos. De nuevo, dichas demostraciones **escapan** al alcance de este informe, sobre todo en esta sección introductoria.

No obstante, para los objetivos, resultados e interpretaciones que sí incumben a este informe, por cierto que se presentarán pruebas experimentales y teóricas cuando sea necesario.

En Memoria Secundaria, siempre se asume que la cantidad de datos es gigantesca y por tanto mucho, mucho mayor que la cantidad de datos que cabe en memoria principal, de manera que los algoritmos y estructuras de datos convencionales usados para resolver en memoria secundaria pueden o no ser los mejores. Un buen ejemplo de esto es *MergeSort*, que no es el mejor algoritmo para ordenamiento en memoria principal, pero ciertamente que es de los mejores en memoria secundaria.[2]

Entonces, se sabe[3] que las operaciones en disco son *bastante* más lentas en memoria secundaria que en memoria principal. Por lo tanto, para los objetivos de este informe no se considerará ningún procesamiento en memoria principal, la razón de esto es precisamente lo anterior: como el una lectura o escritura de memoria secundaria es bastante más lenta que una lectura o escritura de memoria principal, el tiempo asociado al procesamiento en memoria principal es nulo comparado con el tiempo que toma cualquier operación en memoria secundaria. Esto es lo que se conoce como **I/O Model**.

El enfoque de este experimento consiste en medir las operaciones en disco y la efectividad de las estructura de datos que se van a probar según I/O Model.

---

[1] Es un poco intrincado decir que memoria secundaria *es* disco. Pero digamos que una operación de memoria secundaria es una operación en disco. Con operación hablamos de Lectura o Escritura.

[2] Lo dejamos sin pruebas, pero una pequeña búsqueda en `https://scholar.google.cl/scholar?q=mergesort` puede llevar a pruebas fehacientes de que lo que decimos es cierto.

[3] De nuevo, esto es cierto, pero lo declaramos sin pruebas.

1.1. Estructuras de Datos

Hasta ahora se ha dado una discusión bastante grande de qué es memoria secundaria, cómo opera y el modelo en el que se va a circunscribir el experimento. Las estructuras de datos que se van a comparar en este informe son:

- Árboles B
- Hashing Extendible
- Hashing Lineal

1.2. Experimento

Para ver qué estructuras son mejores según las medidas de rendimiento antes expuestas

2. Hipótesis

hipo

3. Diseño Experimental

3.1. Lenguaje de Preferencia

Para implementar las estructuras y hacer los experimentos se ha escogido el lenguaje `C`. Esto porque, en general, los experimentos en memoria secundaria tienden a ser muy largos y un lenguaje como C permite minimizar el tiempo de ejecución, de manera que se pueden obtener los resultados rápidamente, al contrario de lo que pasaría con `Java`. Se puede también dar una discusión de por qué `C` y no `C++`, en este caso simplemente se usa C porque no se necesita la orientación a objetos para resolver un problema de memoria secundaria.

3.2. Limitaciones de la Implementación

El trabajo de experimentar con memoria secundaria es, en general, bastante más complejo que lo que en este experimento particular se hace. Hacer un experimento formal para memoria secundaria involucra aspectos de Sistemas Operativos que escapan al alcance de un curso de Diseño y Análisis de Algoritmos de pregrado. En particular, se requeriría un driver para algún sistema operativo que pudiera escribir un bloque *real* de disco y eliminar una cantidad arbitraria de bytes de un archivo.

Considerando las limitaciones que ofrecen las llamadas a sistema de C, el esquema será el siguiente:

- Una estructura será un sólo archivo gigante con sus datos en disco
- Escribir y modificar un bloque se hará vía las llamadas `fwrite` y `fseek` (que internamente usan la llamada a sistema `write` y algunas otras más).

Aunque el análisis parezca pesimista, realmente estas limitaciones no son de tanta urgencia como parece. En este trabajo se medirá numero de lecturas y escrituras desde/hacia disco y la cantidad de espacio que ocupa cada estructura, de manera que no es tan relevante si se escriben o no bloques reales de disco.[4]

Asimismo, el espacio no tiene que ver con la cantidad de los bloques, sino que con su contenido, y esto es específico de la implementación como también independiente de dónde esté la información.

3.3. •

Para entender mejor los resultados que se presentan en la siguiente sección, en esta sección se describen todos los detalles de implementación de las estructuras de datos que se usan en los experimentos.

Es natural que los resultados puedan variar mucho según la forma en la que se implementan las estructuras de datos en memoria secundaria, y es justamente por eso que leer esta sección es de especial importancia para entender los resultados de este experimento *particular* con las implementaciones aquí descritas.

3.4. Árbol B (B-Tree)

3.5. Hashing Extendible

3.6. Hashing Lineal

Política 1

Política 2

4. Resultados

5. Análisis e Interpretación de los Datos

6. Conclusiones

7. Anexos

[4] Y es de esperar, en todo caso, que el sistema operativo (Linux, al menos) detecte el tamaño de la escritura y lo haga como se espera de todas maneras.