Tarea 1

Mergesort externo V/S Quicksort externo

Profesores: Benjamín Bustos, Gonzalo Navarro
Auxiliares: Sergio Rojas, Pablo Skewes

1. Indicaciones administrativas

- Fecha de entrega: jueves 8 de mayo a las 23:59.
- Grupos de a lo más 3 personas **de la misma sección**.
- Lenguaje a utilizar: C, C++ o Java.

2. Contexto

En este curso se estudia el algoritmo de ordenamiento Mergesort para memoria secundaria. En esta tarea se introducirá el algoritmo de ordenamiento Quicksort externo. Se buscará comparar ambos algoritmos en términos de tiempo de ejecución y cantidad de accesos a disco para poder evaluar cuál es más eficiente.

3. Quicksort Externo

Quicksort externo es la versión de Quicksort para memoria secundaria. La idea general es particionar el arreglo A en a subarreglos del mismo tamaño, de tal forma de que todos los elementos de un arreglo sean menores que los elementos de los arreglos siguientes. De esta forma, se pueden ordenar recursivamente cada subarreglo, terminando cuando estos tengan tamaño M (donde se puede ordenar el bloque en memoria principal en $O(1)$ I/Os). Esto se esperaría que tuviera un costo de $O\left(n \log_a\left(\frac{n}{m}\right)\right)$ I/Os, donde deberíamos proveer $a-1$ pivotes para hacer la separación en los subarreglos.

3.1. Algoritmo

Dado un arreglo A de tamaño N, el algoritmo de Quicksort externo se puede describir como sigue:

- Si $N \leq M$, ordenar el arreglo en memoria principal y devolver.
- Si $N > M$, leer un bloque de A aleatorio, elegir $a-1$ de sus elementos al azar y ordenarlos, de esta forma, esos $a-1$ elementos serán nuestros pivotes.
  - Particionar el arreglo A en a subarreglos, de tal forma de que estos subarreglos sean separados por los pivotes elegidos.
  - Realizar recursivamente Quicksort externo en cada uno de los subarreglos.
  - Finalmente, concatenar los subarreglos ordenados para obtener un único arreglo, de tal forma de que este arreglo resultante esté completamente ordenado.

Notar que los subarreglos no necesariamente tienen el mismo tamaño, ya que los pivotes pueden estar desbalanceados.

4. Descripción de la tarea

4.1. Objetivos

En esta tarea tendrán que:

- Implementar el algoritmo de Mergesort externo.

- Implementar el algoritmo de Quicksort externo.
- Comparar ambos algoritmos en términos de tiempo de ejecución y cantidad de accesos a disco.
- Analizar los resultados obtenidos y sacar conclusiones.

4.2. Metodologías obligatorias

Si bien hay libertad para implementar lo pedido en la tarea, se espera que se sigan ciertas metodologías obligatorias. Estas son:

4.2.1. Implementación de algoritmos

No se aceptará el uso de librerías externas para la implementación de los algoritmos. Se espera que cada uno de los algoritmos sea implementado desde cero. Para Mergesort externo, básese en lo expuesto en cátedra y en el apunte del curso, en la siguiente sección explicaremos cómo definir la aridad. Para Quicksort externo, básese en la descripción del algoritmo dada en este enunciado.

4.2.2. Trabajar en disco

Para trabajar en disco, se guardará como un archivo binario (.bin) el arreglo de tamaño N. Este archivo se deberá leer y escribir en bloques de tamaño B. Es incorrecto realizar una lectura o escritura de tamaño distinto a B. Para esto, se recomienda usar las funciones `fread`, `fwrite` y `fseek` de C. Se recomienda también usar un buffer para evitar leer y escribir cada vez que se accede a un elemento del arreglo. El buffer debe ser de tamaño B es decir, tendremos en memoria principal todo el bloque solicitado.

4.2.3. Arreglo

Ya como el arreglo se encuentra en un archivo .bin en memoria secundaria, se requerirá que implementen una función que pueda interpretar un bloque del archivo binario y llevarlo a un arreglo de números, esto para poder trabajar en memoria principal.

4.2.4. Uso de Docker

Para obtener los resultados en un contexto controlado con memoria principal limitada, se requerirá que utilicen un contenedor de Docker. Este ya se encuentra implementado y tendrán acceso a él en **(link)**, junto a documentación de cómo trabajar con él. Notar que para poder trabajar con Docker, deben instalarlo desde su página oficial (https://www.docker.com/).

Para verificar que instalaron bien Docker, pueden ejecutar el siguiente comando en su terminal:

```
docker --version
```

Luego pueden seguir las instrucciones del contenedor entregado con normalidad.

Recomendamos probar su proyecto en primera instancia sin el contenedor de Docker, esto para que puedan revisar errores de lógica de sus algoritmos más facilmente, sin límite de memoria. Una vez que estén seguros de sus implementaciones, obtener los resultados que les soliciramos con el contenedor, limitando la memoria.

4.2.5. Valores de M y B

El valor de M para la experimentación final será de 50 MB, el valor de B será el tamaño real de un bloque en disco. Debido al contexto de esta tarea, no debería existir ningún número que los bits que lo representan queden en dos bloques distintos.

4.3. Experimentación

4.3.1. Obtención de resultados

Se deberá generar 5 secuencias de números enteros de 64 bits de tamaño total N, con $N \in \{4M, 8M, ...60M\}$ (es decir, 5 secuencias de tamaño $4M$, 5 secuencias de tamaño $8M$, ...), insertarlos desordenadamente en un arreglo y guardarlo en binario en disco. Por cada uno de esos arreglos en binario, se tendrá que realizar lo siguiente:

- Ordenar mediante Mergesort externo con la aridad obtenida, guardando el tiempo de ejecución y la cantidad de accesos a disco.
- Ordenar mediante Quicksort externo con el número de pivotes obtenido, guardando el tiempo de ejecución y la cantidad de accesos a disco.

Debido al tamaño de los arreglos, este proceso puede demorar mucho tiempo. Se recomienda entonces ejecutar la experimentación días antes de la entrega.

Con los datos obtenidos, tendrán que calcular el promedio de tiempo de ejecución y la cantidad de accesos a disco para cada uno de los algoritmos con sus distintos parámetros (promediar los 5 datos obtenidos para Mergesort y los 5 datos obtenidos para Quicksort) y graficar lo siguiente:

- Un gráfico de tiempo de ejecución en función de N de los dos algoritmos.
- Un gráfico de cantidad de accesos a disco en función de N de los dos algoritmos.

Para conseguir lo anterior, es importante que tengan definido el valor de a, para eso, se recomienda realizar lo siguiente:

4.3.2. Obtención de a

Impondremos que la cantidad a de subarreglos para Quicksort externo será el mismo valor de la aridad utilizada para Mergesort externo. Notar que para que los pivotes quepan en un bloque de disco necesitamos que la suma de sus tamaños sea menor o igual a B. Para definir entonces a, realizaremos lo siguiente con Mergesort externo:

- Generar una secuencia de tamaño $60M$ de números enteros de 64 bits, insertarlos desordenadamente en un arreglo y guardarlo en binario en disco.
- Siendo b la cantidad de números de 64 bits que caben en un bloque de tamaño B, se deberá realizar una búsqueda binaria de $a \in [2, b]$, de tal forma de encontrar la aridad a que genera, para Mergesort externo, los mejores resultados para el arreglo definido en el paso anterior.

Con esto obtendremos la mejor aridad a de Mergesort externo, la cual también definirá la cantidad de subarreglos para Quicksort externo.

5. Entregables

Se deberá entregar el código y un informe donde se explique el experimento en estudio. Con esto se obtendrá una nota de código ($NCod$) y una nota de informe ($NInf$). La nota final de la tarea será el promedio simple entre ambas notas ($NT_1 = 0.5 \cdot NCod + 0.5 \cdot NInf$).

5.1. Código

La entrega de código debe ser hecha en C o en C++. Tiene que contener:

- **(0.3 pts)** README: Archivo con las instrucciones para ejecutar el código, debe ser lo suficientemente explicativo para que cualquier persona solo leyendo el README pueda ejecutar la totalidad de su código (incluyendo librerías no entregadas por el equipo docente que potencialmente se deban instalar).
- **(0.2 pts)** Firmas: Cada estructura de datos y función debe tener una descripción de lo que hace y una descripción de sus parámetros de entrada y salida.
- **(1.0 pts)** Uso de Disco: El uso de disco debe ser correcto. Se deben realizar lecturas y escrituras por bloque. Se debe guardar los archivos en binario. Se debe poder realizar la interpretación del binario en memoria principal.
- **(1.5 pts)** Implementación de Mergesort Externo: Debe ser creada por ustedes, no se aceptará el uso de librerías externas. Base su implementación con lo expuesto en cátedra y en el apunte del curso.
- **(1.5 pts)** Implementación de Quicksort Externo: Debe ser creada por ustedes, no se aceptará el uso de librerías externas. Base su implementación con lo expuesto en este enunciado.
- **(0.5 pts)** Experimento: Creación de los arreglos de tamaño N.
- **(0.5 pts)** Obtención de resultados: La forma en el que se obtienen los resultados (a, tiempo de ejecución y accesos a disco) es correcta.
- **(0.5 pts)** Main: Un archivo o parte del código (función main) que permita ejecutar la construcción y experimentos.

5.2. informe

El informe debe ser claro y conciso. Se recomienda hacerlo en LaTeX o Typst. Debe contener:

- **(0.8 pts)** Introducción: Presentación del tema en estudio, resumir lo que se dirá en el informe y presentar una hipótesis.
- **(0.8 pts)** Desarrollo: Presentación de algoritmos, estructuras de datos, cómo funcionan y por qué. Recordar que los métodos ya son conocidos por el equipo docente, lo que importa son sus propias implementaciones.
- **(2.4 pts)** Resultados: Especificación de los datos que se utilizaron para los experimentos, la cantidad de veces que se realizaron los tests, con qué inputs, que tamaño, etc. Se debe mencionar en qué sistema operativo y los tamaños de sus cachés y RAM con los que se ejecutaron los experimentos. Se deben mostrar gráficos/tablas y mencionar solo lo que se puede observar de estos, se deben mostrar los valores y parámetros que se están usando.
- **(1.2 pts)** Análisis: Comentar y concluir sus resultados. Se hacen las inferencias de sus resultados.
- **(0.8 pts)** Conclusión: Recapitulación de lo que se hizo, se concluye lo que se puede decir con respecto a sus resultados. También ven si su hipótesis se cumplió o no y analizan la razón. Por último, se menciona qué se podría mejorar en su desarrollo en una versión futura, qué falta en su documento, qué no se ha resuelto y cómo se podrían extender.

Todo lo mencionado debe estar en sus informes en las secciones en las que se señalan, la falta de algún aspecto o la presencia de algún aspecto en una sección equivocada hará que no se tenga la totalidad del puntaje.