Algoritmos en memoria secundaria

IIC2283

Algoritmos en memoria secundaria

Hasta ahora hemos supuesto que la entrada de un algoritmo cabe en la memoria principal (RAM)

- No hemos considerado el costo asociado a acceder a un dato, ni hemos mencionado que este costo podría ser distinto dependiendo del lugar donde es almacenado

Algoritmos en memoria secundaria

Hasta ahora hemos supuesto que la entrada de un algoritmo cabe en la memoria principal (RAM)

- No hemos considerado el costo asociado a acceder a un dato, ni hemos mencionado que este costo podría ser distinto dependiendo del lugar donde es almacenado

Al utilizar un algoritmo en un computador es posible que el tamaño de la entrada sea mayor que el tamaño de la memoria principal

Algoritmos en memoria secundaria

Hasta ahora hemos supuesto que la entrada de un algoritmo cabe en la memoria principal (RAM)

- No hemos considerado el costo asociado a acceder a un dato, ni hemos mencionado que este costo podría ser distinto dependiendo del lugar donde es almacenado

Al utilizar un algoritmo en un computador es posible que el tamaño de la entrada sea mayor que el tamaño de la memoria principal

- Tenemos entonces que usar algún sistema de almacenamiento secundario (por ejemplo, un disco duro)

Algoritmos en memoria secundaria

El acceso a los datos en memoria principal es más rápido que en memoria secundaria

Algoritmos en memoria secundaria

El acceso a los datos en memoria principal es más rápido que en memoria secundaria

- Es posible que un acceso a memoria secundaria sea equivalente a miles de operaciones en memoria principal

Algoritmos en memoria secundaria

El acceso a los datos en memoria principal es más rápido que en memoria secundaria

- Es posible que un acceso a memoria secundaria sea equivalente a miles de operaciones en memoria principal
- Por esto la memoria principal es más cara y más pequeña que la memoria secundaria

Algoritmos en memoria secundaria

El acceso a los datos en memoria principal es más rápido que en memoria secundaria

- Es posible que un acceso a memoria secundaria sea equivalente a miles de operaciones en memoria principal
- Por esto la memoria principal es más cara y más pequeña que la memoria secundaria

La diferencia de velocidad entre memoria principal y secundaria puede ser tan grande que debe ser tomada en cuenta al diseñar un algoritmo que puede tener entradas muy grandes

Algoritmos en memoria secundaria

El acceso a los datos en memoria principal es más rápido que en memoria secundaria

- Es posible que un acceso a memoria secundaria sea equivalente a miles de operaciones en memoria principal
- Por esto la memoria principal es más cara y más pequeña que la memoria secundaria

La diferencia de velocidad entre memoria principal y secundaria puede ser tan grande que debe ser tomada en cuenta al diseñar un algoritmo que puede tener entradas muy grandes

- En general queremos diseñar algoritmos que minimicen el número de accesos a memoria secundaria

Algoritmos en memoria secundaria

Para poder cuantificar el número de accesos a memoria secundaria necesitamos un modelo de computación que los considere.

Algoritmos en memoria secundaria

Para poder cuantificar el número de accesos a memoria secundaria necesitamos un modelo de computación que los considere.

- Este modelo debe tomar en cuenta cómo se realiza el acceso a memoria secundaria en un computador

Algoritmos en memoria secundaria

Para poder cuantificar el número de accesos a memoria secundaria necesitamos un modelo de computación que los considere.

- Este modelo debe tomar en cuenta cómo se realiza el acceso a memoria secundaria en un computador

El análisis que vamos a realizar es relevante en cualquier escenario donde tengamos dispositivos de almacenamiento con distintas velocidades de acceso

Algoritmos en memoria secundaria

Para poder cuantificar el número de accesos a memoria secundaria necesitamos un modelo de computación que los considere.

- Este modelo debe tomar en cuenta cómo se realiza el acceso a memoria secundaria en un computador

El análisis que vamos a realizar es relevante en cualquier escenario donde tengamos dispositivos de almacenamiento con distintas velocidades de acceso

- Bajo el supuesto de que los dispositivos más lentos tienen mayor capacidad de almacenamiento

Algoritmos en memoria secundaria

Para poder cuantificar el número de accesos a memoria secundaria necesitamos un modelo de computación que los considere.

- Este modelo debe tomar en cuenta cómo se realiza el acceso a memoria secundaria en un computador

El análisis que vamos a realizar es relevante en cualquier escenario donde tengamos dispositivos de almacenamiento con distintas velocidades de acceso

- Bajo el supuesto de que los dispositivos más lentos tienen mayor capacidad de almacenamiento
- Podemos incluir más de dos dispositivos de almacenamiento

Un modelo de computación para memoria secundaria

En un computador un acceso a memoria secundaria no lee un dato sino que un bloque de datos

Un modelo de computación para memoria secundaria

En un computador un acceso a memoria secundaria no lee un dato sino que un bloque de datos

- Un bloque corresponde a un conjunto de datos físicamente contiguos en memoria secundaria

Un modelo de computación para memoria secundaria

En un computador un acceso a memoria secundaria no lee un dato sino que un bloque de datos

- Un bloque corresponde a un conjunto de datos físicamente contiguos en memoria secundaria
- El acceso a un bloque no es más costoso que el acceso a un dato

Un modelo de computación para memoria secundaria

En un computador un acceso a memoria secundaria no lee un dato sino que un bloque de datos

- Un bloque corresponde a un conjunto de datos físicamente contiguos en memoria secundaria
- El acceso a un bloque no es más costoso que el acceso a un dato
  - Por eso conviene leer o escribir un bloque de datos en lugar de un dato individual

Un modelo de computación para memoria secundaria

En un computador un acceso a memoria secundaria no lee un dato sino que un bloque de datos

- Un bloque corresponde a un conjunto de datos físicamente contiguos en memoria secundaria
- El acceso a un bloque no es más costoso que el acceso a un dato
  - Por eso conviene leer o escribir un bloque de datos en lugar de un dato individual
- En general, el tamaño de un bloque es mucho más pequeño que el tamaño de la memoria principal

Un modelo de computación para memoria secundaria

Consideramos entonces dos constantes:

Un modelo de computación para memoria secundaria

Consideramos entonces dos constantes:

- B: Número de datos en un bloque leído o escrito en memoria secundaria

Un modelo de computación para memoria secundaria

Consideramos entonces dos constantes:

- B: Número de datos en un bloque leído o escrito en memoria secundaria
- M: Número de datos en memoria principal

Un modelo de computación para memoria secundaria

Consideramos entonces dos constantes:

- B: Número de datos en un bloque leído o escrito en memoria secundaria
- M: Número de datos en memoria principal
  - Este número puede ser menor que el tamaño total de la memoria principal, ya que puede representar la cantidad de memoria asignada por el sistema operativo para el funcionamiento de un algoritmo

Un modelo de computación para memoria secundaria

Consideramos entonces dos constantes:

- B: Número de datos en un bloque leído o escrito en memoria secundaria
- M: Número de datos en memoria principal
  - Este número puede ser menor que el tamaño total de la memoria principal, ya que puede representar la cantidad de memoria asignada por el sistema operativo para el funcionamiento de un algoritmo

Por ejemplo, vamos a estudiar un algoritmo para ordenar una lista de números enteros que no cabe en memoria principal

- B y M son entonces el número de enteros en un bloque y en memoria principal, respectivamente

Un modelo de computación para memoria secundaria

El acceso a memoria secundaria se realiza a través de dos procedimientos:

Un modelo de computación para memoria secundaria

El acceso a memoria secundaria se realiza a través de dos procedimientos:

- **LeerMemoriaSecundaria**(I, pos): Dado un puntero I a un archivo (en memoria secundaria) y un número entero $pos \geq 1$, retorna el bloque que está en la posición pos de I

Un modelo de computación para memoria secundaria

El acceso a memoria secundaria se realiza a través de dos procedimientos:

- **LeerMemoriaSecundaria**(I, pos): Dado un puntero I a un archivo (en memoria secundaria) y un número entero $pos \geq 1$, retorna el bloque que está en la posición pos de I
- **EscribirMemoriaSecundaria**(O, pos, $bloque$): Dado un puntero O a un archivo, un número entero $pos \geq 1$ y un bloque de datos $bloque$, escribe $bloque$ en la posición pos de O

Un modelo de computación para memoria secundaria

El acceso a memoria secundaria se realiza a través de dos procedimientos:

- **LeerMemoriaSecundaria**(I, pos): Dado un puntero I a un archivo (en memoria secundaria) y un número entero $pos \geq 1$, retorna el bloque que está en la posición pos de I
- **EscribirMemoriaSecundaria**(O, pos, $bloque$): Dado un puntero O a un archivo, un número entero $pos \geq 1$ y un bloque de datos $bloque$, escribe $bloque$ en la posición pos de O

En ambos procedimientos suponemos que el primer bloque de un archivo está en la posición 1.

Ordenando una lista en memoria secundaria

Ejercicio

Describa el algoritmo Mergesort para ordenar de menor a mayor una lista de números enteros (suponiendo que la lista cabe en memoria principal)

Ordenando una lista en memoria secundaria

Ejercicio

Describa el algoritmo Mergesort para ordenar de menor a mayor una lista de números enteros (suponiendo que la lista cabe en memoria principal)

Vamos a extender este algoritmo para el caso en que la lista tiene N enteros y la memoria principal puede almacenar M enteros con $M < N$

- Recuerde que en este caso B es el número (máximo) de enteros en un bloque leído o escrito en memoria secundaria

Un procedimiento para leer desde memoria secundaria

En el archivo apuntado por I, el siguiente procedimiento lee k bloques desde la posición pos y almacena los enteros en estos bloques en la lista L

LeerArchivo(I, pos, k, L)
    $L := \emptyset$
    $i := pos$
    $bloque :=$ **LeerMemoriaSecundaria**(I, i)
    while $i \leq pos + k - 1$ **and** $bloque \neq \emptyset$ **do**
        Append(L, **TransformarLista**($bloque$))
        $i := i + 1$
        $bloque :=$ **LeerMemoriaSecundaria**(I, i)

Un procedimiento para leer desde memoria secundaria

En el archivo apuntado por I, el siguiente procedimiento lee k bloques desde la posición pos y almacena los enteros en estos bloques en la lista L

LeerArchivo(I, pos, k, L)
    $L := \emptyset$
    $i := pos$
    $bloque :=$ **LeerMemoriaSecundaria**(I, i)
    while $i \leq pos + k - 1$ **and** $bloque \neq \emptyset$ **do**
        Append(L, **TransformarLista**($bloque$))
        $i := i + 1$
        $bloque :=$ **LeerMemoriaSecundaria**(I, i)

Nótese que este procedimiento funciona bajo la restricción $k \cdot B \leq M$

- Puesto que la lista L es almacenada en memoria principal

Un procedimiento para leer desde memoria secundaria

Tres comentarios adicionales sobre **LeerArchivo**:

- **LeerMemoriaSecundaria**(I, *pos*) retorna $\emptyset$ si no hay un bloque de datos en la posición *pos* de I
- **Append**(L_1, L_2) reemplaza L_1 por la concatenación de L_1 con L_2
- **TransformarLista**(*bloque*) transforma un bloque de n números enteros en una lista de n números enteros

Un procedimiento para escribir en memoria secundaria

Desde la posición *pos* en el archivo apuntado por O, el siguiente procedimiento escribe por bloques los enteros almacenados en la lista L

EscribirArchivo(O, *pos*, L)
    $p := pos$
    $i := 1$
    while $i \leq$ **Length**(L) **do**
        $j :=$ mín$\{i + B - 1,$ **Length**(L)$\}$
        bloque := **TransformarBloque**(**SubList**(L, i, j))
        EscribirMemoriaSecundaria(O, p, *bloque*)
        $p := p + 1$
        $i := i + B$

Un procedimiento para escribir en memoria secundaria

Desde la posición *pos* en el archivo apuntado por O, el siguiente procedimiento escribe por bloques los enteros almacenados en la lista L

EscribirArchivo(O, *pos*, L)
    $p := pos$
    $i := 1$
    while $i \leq$ **Length**(L) **do**
        $j := \text{mín}\{i + B - 1, \textbf{Length}(L)\}$
        bloque := **TransformarBloque**(**SubList**(L, i, j))
        EscribirMemoriaSecundaria(O, p, *bloque*)
        $p := p + 1$
        $i := i + B$

Nótese que este procedimiento funciona bajo la restricción **Length**(L) $\leq M$

- Puesto que la lista L es almacenada en memoria principal

Un procedimiento para escribir en memoria secundaria

Tres comentarios adicionales sobre **EscribirArchivo**:

- **Length**(L) retorna el largo de la lista L
  - Recuerde que el primer elemento de L está en la posición 1
- **SubList**(L, i, j) retorna la sub-lista de L entre las posiciones i y j (suponiendo que $i \leq j$)
- **TransformarBloque**(L) transforma una lista de n números enteros en un bloque de n números enteros
  - Suponemos que $n \leq B$