Diseño y Análisis de Algoritmos - Apuntes

Jérémy Barbay

10 de abril de 2012

1. Un modelo mas fino quel anterior
   a) Cuantos paginas quedan en memoria local?
      - no tan importante para busqueda
      - muy importante para applicaciones de computacion con mucho datos.
   b) Prerequisitos
      - Notaciones
        - B = Tamano pagina
        - N = cantidad de elementos en total
        - n = cantidad de paginas con elementos = N/B
        - M = cantidad de memoria local
        - m = cantidad de paginas locales = M/B
        - mnemotechnique:
          - N, M, B en cantidad de "palabras maquinas" (=bytes?)
          - n, m en cantidad de paginas
          - $n << N$, $m << M$, por eso son "pequeñas" letras
      - Formulas utiles:
        - $\log(x!) \approx x \log x$
        - $\log \binom{M}{B} \approx B \lg \frac{M}{B}$

1. En estas notaciones, usando resultos previos:
   - `Insertion Sort` (en un B-Arbol)
     - usa dictionarios en memoria externa
     - $N \lg N / \lg B = N \log_B N$
   - `Heap Sort`
     - usa colas de prioridades en memoria externa
     - $N \lg N / \lg B = N \log_B N$
   - Eso es optimo o no?
2. Cotas Inferiores en Memoria Secundaria
   - para buscar en un diccionario?
     - en modelo RAM? (de comparaciones)
       - $\lg N$
     - en modele Memoria Externa? (de comparaciones)
       - al maximo $\lg N / \lg B = \log_B N$
   - para fusionar dos arreglos ordenados?

- en modelo RAM?
  - N
- en modelo Memoria Externa con paginas de tamano B?
  - al maximo $N/B = n$

- para fusionar k arreglos ordenados?
  - en modelo RAM?
    - N
  - en modelo de Memoria Externa con M paginas de tamano B?
    - al maximo $N/B = n$ **si $M >kB$**
    - que hacemos si $M < kB$?
      - el caso extremo cuando $k = N$ se llama ordenar
      - veamos como ordenar, generalisar a la union de k arreglos ordenados es un ejericio despues.

- para Ordenar
  - (corrige y adapte la prueba de http://www.daimi.au.dk/l̃arge/ioS06/Alower.pdf)
  - en modelo RAM de comparaciones
    - $N \lg N$
  - en modelo Memoria Externa con n/B paginas de tamano B
    - $\Omega(N/B\frac{\lg(N/B)}{\lg(M/B)})$
    - que se puede notar mas simplamente $\Omega(n \lg_m n)$
  - Prueba:
    a) en vez de considerar el problema de ordenamiento, supponga que el arreglo sea una permutacion y considera el problema (equivalente en ese caso) de identificar cual permutacion sea.
    b) inicialemente, pueden ser $N!$ permutaciones.
      - supponga que cada bloque de B elementos sea ya ordenado (impliqua un costo de al maximo $n = N/B$ accessos a la memoria externa).
      - queda $N!/((B!)^n)$ permutaciones posibles.
    c) para cada accesso a una pagina de memoria externa, cuantos permutaciones potenciales podemos eliminar?
      - con M entradas en memoria primaria
      - B nuevas entradas se pueden quedar de $\binom{M}{B} = \frac{M!}{B!(M-B)!}$ maneras distintas
      - calcular la union de los $M + B$ elementos reduce la cuantidad de permtuaciones por un factor de $1/\binom{M}{B}$
      - después de t accessos (distintos) a la memoria externa, se reduci la cuantidad de permutaciones a $N!/((B!)^n\binom{M}{B}^t)$
    d) cuanto accessos a la memoria sean necesarios para que queda al maximo una permutacion?
      - $N!/((B!)^n\binom{M}{B}^t)$ debe ser al maximo uno.
      - usamos las formulas siguientes:
        - $\log(x!) \approx x \log x$
        - $\log\binom{M}{B} \approx B \lg \frac{M}{B}$
      - Inicialemente, tenemos la inequalidad siguiente:

$$N! \leq (B!)^n \binom{M}{B}^t$$

- aplicando el log de ambos lado (y las formulas previas) queda con

$$N \lg N \leq nB \lg B + tB \lg \frac{M}{B}$$

- Una secuencia de reduccion simples da:

$$t \geq \frac{N \lg N - nB \lg B}{B \lg(M/B)}$$

- que se puede reescribir como $\frac{N \lg(N/B)}{B \lg(M/B)}$
- Pero $n = N/B$ y $m = M/B$, asi se puede reescribir $\frac{n \lg n}{\lg m}$
- en final, deducimos $t \geq n \log_m n$.

- BONUS: Para ordenar strings, un caso particular (donde la comparacion de dos elementos tiene un costo variable):
  - http://www.brics.dk/~large/Papers/stringsstoc97.pdf
  - $\Omega(N_1/B \log_{M/B}(N_1/B) + K_2 \lg_{M/B} K_2 + N/B)$
  - donde
    - N_1 es la suma de los tamanos de las caldenas mas cortas que B
    - K_2 es la cuantidad de caldenas mas largas que B

3. Ordenar en Memoria Externa N elementos (en $n = N/B$ paginas)
   - http://en.wikipedia.org/wiki/External_sorting
   - Usando dictionarios o colas de prioridades en memoria externa
     - $N \lg N / \lg B = N \log_B N$
     - No es "ajustado" con la cota inferior
     - impliqua
       - o que hay un mejor algoritmo
       - o que hay una mejor cota inferior
   - Queda un algoritmo de ordenamiento: MergeSort
     - usa la fusion de $m - 1$ arreglos ordenados en memoria externa:
       a) carga en memoria principal $m - 1$ paginas, cada una la primera de su arreglo.
       b) calcula la union de estas paginas en la pagina m de memoria principal,
         - botando la pagina cuando llena
         - cargando una nueva pagina (del mismo arreglo) cuando vacilla
       c) La complejidad es n accessos.
     - Algoritmo:
       a) ordena cada de las n paginas $\rightarrow$ n accessos
       b) Cada nodo calcula la union de m arreglos y escribe su resultado, pagina por pagina, en la memoria externa.
     - Analisis:
       - Cada nivel de recurencia costa n accessos
       - Cada nivel reduce por $m - 1$ la cantidad de arreglos
       - la complejidad total es de orden $n \log_m n$ accessos. (ajustado)

4. **BONUS** cota inferior para una cola de prioridad?
   - una cola de prioridad se puede usar para ordenar (con N accessos)
   - hay una cota inferior para ordernar de $n \log_m n$
   - entonces????