CC4102/CC40A/CC53A - Diseño y Análisis de Algoritmos

Jérémy Barbay

26 March 2011

Contents

1 Algoritmos y Estructuras de Datos para Memoria Secundaria (3 semanas = 6 charlas)

1.1 PREREQUISITOS DE UNIDAD

- Apuntes de CC3001:
  - Arboles 2-3
    - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Diccionario/#4
  - Arboles B
    - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Diccionario/#5

- Mergesort y Ordenamiento Externo
  - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Ordenacion/#5
- Colas de Prioridades
  - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/TDA/#4

1.2 Modelo de computacion en memoria secundaria. Accesos secuenciales y aleatorios

1.2.1 MATERIAL A LEER

- Memory hierarchy
  - http://en.wikipedia.org/wiki/Memory_hierarchy
  - En castellano, mas corto, http://es.wikipedia.org/wiki/Jerarqu%C3%ADa_de_memoria

1.2.2 APUNTES

Arquitectura de un computador: la memoria

1. Muchos niveles de memoria
   - Procesador
   - registros
   - Cache L1
   - Cache L2
   - Memory
   - Cache
   - Disco Duro magnetico / Memory cell
   - Akamai cache
   - Discos Duros en la red
   - CD y DVDs tambien son "memoria"
2. Diferencias
   - velocidad
   - precio de construccion
   - relacion fisica entre volumen y velocidad
   - volatil o no
   - accesso arbitrario en tiempo constante o no.
   - latencia vs debito

3. Modelos formales

   - RAM
   - Jerarquia con dos niveles, paginas de tamano B
   - Jerarquia con k niveles, de paginas de tamanos $B_1, ..., B_k$
   - "Cache oblivious"
   - Otros... mas practicas, mas dificil a analizar.

1.3 Diccionarios en Memoria Externa

1.3.1 MATERIAL A LEER

- B-Arboles:
  - Arboles B en apuntes de CC3001
    - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Diccionario/#5
  - Árbol-B
    - http://es.wikipedia.org/wiki/%C3%81rbol-B
  - Árbol-B+ (corto)
    - http://es.wikipedia.org/wiki/%C3%81rbol-B%2B
  - Árbol-B* (corto)
    - http://es.wikipedia.org/wiki/%C3%81rbol-B*
    - (en ingles http://en.wikipedia.org/wiki/B*-tree)
- Descripcion de van Emde Boas arboles
  - http://en.wikipedia.org/wiki/Van_Emde_Boas_tree

1.3.2 APUNTES

1. B-arbol

   (a) $(2, 3)$ Arbol: un arbol de busqueda donde
   - cada nodo tiene 1 o 2 llaves, que corresponde a 2 o 3 hijos, con la excepcion de la raiz;
   - todas las hojas son al mismo nivel (arbol completamente balanceado)
   - Propiedades:
     - altura de un $(2, 3)$ arbol?
     - tiempo de busqueda?
     - insercion en un $(2, 3)$ arbol?
     - delecion en un $(2, 3)$ arbol?

   (b) $(d, 2d)$ Arbol un arbol donde

- cada nodo tiene de d a $2d$ hijos, con la excepcion de la raiz (significa $d-1$ a $2d-1$ llaves).
- todas las hojas son al mismo nivel (arbol completamente balanceado)
- Propiedades:
  - altura de un $(d, 2d)$ arbol?
  - tiempo de busqueda?
  - insercion en un $(d, 2d)$ arbol?
  - delecion en un $(d, 2d)$ arbol?

(c) B-Arbol, y variantes

- B-Arbol
  - http://www.youtube.com/watch?v=coRJrcIYbF4
  - http://en.wikipedia.org/wiki/B-tree
- B^* arbol
  - otros nodos que la raiz son llenos al menos hasta 2/3 (en vez de 1/2)
  - http://en.wikipedia.org/wiki/B*-tree
- B^+ arbol
  - the leaf nodes of the tree are chained together in the form of a linked list.

2. Van Emde Boas arbol (vEB)

(a) Historia:

- Originalemente (1977) un estructura de datos normal, que suporta todas las operaciones en $O(\lg \lg n)$, inventada por el equipo de Peter van Emde Boas.
- No considerado utiles en practica para "pequenos" arboles.
- Applicacion a "Cache-Oblivious" algoritmos y estructuras de datos
  - optimiza el cache sin conocer el tamano B de sus paginas
  - => optimiza todos los niveles sin conocer $B_1, ..., B_k$
- otras applicaciones despues en calculo parallelo (?)

(b) Definicion

- Cada nodo contiene un arbol van EmdeBoas sobre $\sqrt{n}$ elementos
- $\lg \lg n$ niveles de arboles
- operadores:
  - Findnext
  - Insert
  - Delete

(c) Analisis

- Busqueda en "tiempo" $O(\lg n / \lg B)$ a cualquier nivel i, donde el tiempo es la cuantidad de accessos al cache del nivel considerado
- Insercion
- Delecion

1.4 Colas de prioridad en memoria secundaria. Cotas inferiores.

1.4.1 MATERIAL a LEER

- Apuntes CC3001
  - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/TDA/#4
- Colas de Prioridad
  - http://en.wikipedia.org/wiki/Priority_queue]]
- Heaps
  - http://en.wikipedia.org/wiki/Heap_data_structure]]
- van Emde Boas Queues
  - http://www.itl.nist.gov/div897/sqg/dads/HTML/vanemdeboas.html

1.4.2 APUNTES

1. Colas de Prioridades tradicional:
   - que se necesita?
     - Operadores
       - *insert(key,item)*
       - *findMind()*
       - *extractMin()*
     - Operadores opcionales
       - *heapify*
       - *increaseKey, decreaseKey*
       - *find*
       - *delete*
       - *successor/predecessor*
       - *merge*
       - ...
   - diccionarios: demasiado espacio para que se pide
     - menos operadores que diccionarios
       - => mas flexibilidad en la representacion
       - => mejor tiempo y/o espacio
   - **binary heap**: una estructura a dentro de muchas otras:
     - sequence-heaps
     - binomial queues

  - Fibonacci heaps
  - leftist heaps
  - min-max heaps
  - pairing heaps
  - skew heaps
  - *van Emde Boas queues*
- **van Emde Boas queues**
  - Definicion:
    "An efficient implementation of priority queues where insert, delete, get minimum, get maximum, etc. take O(log log N) time, where N is the total possible number of keys. Depending on the circumstance, the implementation is null (if the queue is empty), an integer (if the queue has one integer), a bit vector of size N (if N is small), or a special data structure: an array of priority queues, called the bottom queues, and one more priority queue of array indexes of the bottom queues."
    * rendimiento en memoria secundaria de "binary heap": muy malo?

2. Colas de Prioridades en Memoria Secundaria: diseno

   - El equivalente de B-Arbol
   - Muchas alternativas en practica
     - Buffer trees
     - M/B-ary heaps
     - array heaps
     - R-Heaps
     - Array Heaps
     - sequence heaps

3. Colas de Prioridades en Memoria Secundaria: cota inferior?

   - Cota inferior para dictionaries es una cota inferior por colas de prioridades o no?
     - No. La reduccion es en la otra direccion.
   - Cual es la cota inferior mas simple que se puede imaginar?
     - $\Omega(n/B)$

1.4.3 REFERENCIAS ADICIONALES

- http://www.dcc.uchile.cl/gnavarro/algoritmos/tesisRapa.pdf
  - paginas 9 hasta 16
- Otras referencias en http://www.leekillough.com/heaps/
- "An experimental Study of Priority Queues in External Memories" by Brengel, Crauser, Ferragina and Meyer
  - http://portal.acm.org/citation.cfm?id=351827.384259

1.5 Ordenamiento en memoria secundaria: Mergesort. Cota inferior.

1.5.1 MATERIAL A LEER

- Algoritmos de Ordenamiento en Apuntes de CC3001
  - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Ordenacion/
  - ☐ Quicksort
  - ☐ Heapsort
  - ☐ Bucketsort
  - ☐ Mergesort
  - ☐ Ordenamiento Externo
- Ordenamiento en Memoria externa en Wikipedia:
  - http://en.wikipedia.org/wiki/External_sorting
- Cota Inferior Ordenamiento en Memoria externa
  - http://www.daimi.au.dk/~large/ioS06/Alower.pdf

1.5.2 APUNTES

1. Un modelo mas fino
   (a) Cuantos paginas quedan en memoria local?
   - no tan importante para busqueda
   - muy importante para applicaciones de computacion con mucho datos.

   (b) Nuevas 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
     - $n << N$, $m << M$

   (c) 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)
    - $\lg N / \lg B = \log_B N$ (ajustado)
- para fusionar dos arreglos ordenados?
  - en modelo RAM?
    - N
  - en modelo Memoria Externa con paginas de tamano B?
    - $N/B = n$ (ajustado)
- para fusionar k arreglos ordenados?
  - en modelo RAM?
    - N
  - en modelo de Memoria Externa con M paginas de tamano B?
    - $N/B = n$ (si $M > kB$)
- para Ordenar
  - (corrige y adapte la prueba de http://www.daimi.au.dk/~large/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:
    - en vez de considerar el problema de ordenamiento, supponga que el arregla sea una permutacion y considera el problema (equivalente en ese caso) de identificar cual permutacion sea.
    - 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.
    - para cada accesso a una pagina de memoria externo, 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 $1/\binom{M}{B}$
- despues de t accessos (distintos) a la memoria externa, se reduci la cuantidad de permutaciones a $N!/((B!)^n \binom{M}{B}^t)$

* 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:
  - $\log(x!) \approx x \log x$
  - $\log \binom{M}{B} \approx B \lg \frac{M}{B}$

$N!$	$\leq$	$(B!)^n \binom{M}{B}^t$
$N \lg N$	$\leq$	$nB \lg B + tB \lg \frac{M}{B}$
t	$\geq$	$\frac{N \lg N - nB \lg B}{B \lg(M/B)}$
	$\geq$	$\frac{N \lg(N/B)}{B \lg(M/B)}$
	$\geq$	$\frac{n \lg n}{\lg m}$
	$\geq$	$n \log_m n$

- 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
    1. N_1 es la suma de los tamanos de las caldenas mas cortas que B
    2. 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,
(c) botando la pagina cuando llena
(d) cargando una nueva pagina (del mismo arreglo) cuando vacilla
(e) 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????

1.6 Resultados de Aprendisajes de la Unidad Dos

- Comprender el modelo de costo de memoria secundaria
- Conocer algoritmos y estructuras de datos basicos que son eficientes en memoria secundaria,
- y el analisis de su desempeno.