Unidad 2: Memoria Secundaria

Jeremy Barbay

13 April 2011

Índice

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

1.1. DESCRIPCION de la Unidad

1. Modelos de Memoria
2. Diccionarios en Memoria Secundaria
3. Colas de Prioridades en Memoria Secundaria
4. Ordenamiento en Memoria Secundaria
5. Cotas Inferiores en Memoria Secundaria

1.2. PREREQUISITOS DE UNIDAD

* Apuntes de CC3001:

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

* Mergesort y Ordenamiento Externo

- http://www.dcc.uchile.cl/ bebustos/apuntes/cc3001/Ordenacion/5http://www.dcc.uchile.cl/~bebustos/apuntes/cc3

* Colas de Prioridades

- http://www.dcc.uchile.cl/ bebustos/apuntes/cc3001/TDA/4http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/T

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

1.3.1. MATERIAL A LEER

- Memory hierarchy
- http://en.wikipedia.org/wiki/Memory$_h ierarchyhttp : //en.wikipedia.org/wiki/Memory_hierarchyEncastellano,$

1.3.2. 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” http://en.wikipedia.org/wiki/Cache-oblivious$_algorithmhttp://en.wikipedia.org/wiki/Cache-$

1.3.3. PREGUNTAS [6/7] :PREGUNTAS:

Cuantas segundas vale 1ns? Cuantas segundas vale un nano segunda?

1. ☐ 10^{-12} segundas
2. ☐ 10^{-9} segundas
3. ☐ 10^{-6} segundas
4. ☐ 10^{-3} segundas
5. ☐ otra respuesta

Camino de accesso a valores Cuando un programa hace un acceso a dos elementos de un arreglo, cual es el camino de accesso a estas valores el mas {frecuente / probable }?

1. ☐ Registros
2. ☐ Caches (1,2 o 3)
3. ☐ RAM (principal)
4. ☐ Disco Duro
5. ☐ otra respuesta
6. ☐ Cache + RAM + Disco Duro
7. ☐ Cache + Disco Duro
8. ☐ RAM + Disco Duro

Cual es la significacion de GHz? :CANC: Que significa que un procesador funciona a 4 GHz?

1. ☐ 4 instrucciones per segunda
2. ☐ $4 * 10^3$ instrucciones per segunda
3. ☐ $4 * 10^6$ instrucciones per segunda
4. ☐ $4 * 10^9$ instrucciones per segunda
5. ☐ otra respuesta

Nivel a 25ns Cual de los niveles siguentes parece el mas cerca de un tiempo de acceso de 25 ns, por un computador funcionando a 4 GHz?

1. ☐ Registro
2. ☐ Cache (L1, L2 o L3)
3. ☐ RAM
4. ☐ Disco duro
5. ☐ otra respuesta

Nivel a 1ns Cual de los niveles siguentes parece el mas cerca de un tiempo de acceso de 1 ns, por un computador funcionando a 4 GHz?

1. ☐ Registro
2. ☐ Cache (L1, L2 o L3)
3. ☐ RAM
4. ☐ Disco duro
5. ☐ otra respuesta

Cuanto se demora una instruccion? Un CPU funciona a 4 GHz: cuanto se demora una instruccion? (elija el valor mas cercano).

1. ☐ 1 nano segunda
2. ☐ 1 micro segunda
3. ☐ 1 mili segunda
4. ☐ 1 centi segunda
5. ☐ otra respuesta

TODO Cuesto (en plata) de la memoria

1.4. Diccionarios en Memoria Externa

1.4.1. MATERIAL A LEER

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

1.4.2. APUNTES

1. Cota inferior en el modelo de comparacion
   - Una cota inferior trivial de la cantidad de accesos (en el modelo de comparacion) es $\Omega(\log_B n)$:
     - la busqueda (en el modelo de comparacion) tiene una cota inferior de $\Omega(\log n)$ sobre la cantidad de comparaciones para un "find"
     - una pagina de B elementos tiene una cota **superior** de $\lg B$ sobre la cantidad de informacion que puede dar sobre la posicion relativa de un elemento x.
     - eso resulta en una cota inferior de $\Omega(\log_B n)$ sobre la cantidad de accesos a la memoria secundaria.
2. Cota superior en el modelo de comparacion

   Nota que la cota inferior es valida a todos niveles de memoria, por cualquier valor de B. La cota es asintoticamente estricta. Se puede lograr con un B-arbol, conociendo B, o con un vEB-arbol, sin conocer B.

   Nota que en la seccion [*Dominiosuna estructura de datos con mejor rendimiento, afuera del modelo de comparacion.

   a) B-arbol

   1) $(2, 3)$ 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?
   2) $(d, 2d)$ 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?

3) B-Arbol, y variantes
- B-Arbol
  - http://www.youtube.com/watch?v=coRJrcIYbF4http://www.youtube.com/watch?v=coRJrcIYbF
  - http://en.wikipedia.org/wiki/B-treehttp://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*-treehttp://en.wikipedia.org/wiki/B*-tree
- B^+ arbol
  - una caldena conecta todas las hojas del arbol: permite de describir intervales de soluciones.

b) Van Emde Boas arbol (vEB)

1) 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 (?)

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

3) 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

1.4.3. PREGUNTAS [3/3] :PREGUNTAS:

Cola de Prioridad: operaciones Cuales (no) son operaciones de un diccionario?

1. ☐ insert(key,item)
2. ☐ search(key)
3. ☐ delete(key)
4. ☐ findNext(key)
5. ☐ findPrevious(key)
6. ☐ findMind()
7. ☐ extractMin()

$(2,3)$ **arboles** Cual es el orden de {la altura, el tiempo de busqueda, el tiempo de insercion, el tiempo de delecion} de un $(2,3)$ arbol con n valores?

1. ☐ menos que $\log_3 n + O(1)$
2. ☐ $\log_3 n + O(1)$
3. ☐ entre $\log_3 n$ y $\log_2 n$
4. ☐ $\log_2 n + O(1)$
5. ☐ otra respuesta

$(2,3)$ **arboles con** $n \in \{8, 9, 256\}$ **elementos :CANC:** Cual es {la altura, el tiempo de busqueda, el tiempo de insercion, el tiempo de delecion} de un $(2,3)$ arbol con $n \in \{8, 9, 256\}$ valores?

1. ☐ menos que $\log_3 n + O(1)$
2. ☐ $\log_3 n + O(1)$
3. ☐ entre $\log_3 n$ y $\log_2 n$
4. ☐ $\log_2 n + O(1)$
5. ☐ otra respuesta

B **arboles vs** $(2,3)$ **arboles** Cual es {la altura, el tiempo de busqueda, el tiempo de insercion, el tiempo de delecion} de un B arbol con $n = \{8, 9, 256\}$ valores?

1. ☐ menos que $\log_3 n + O(1)$
2. ☐ $\log_3 n + O(1)$
3. ☐ entre $\log_3 n$ y $\log_2 n$
4. ☐ $\log_2 n + O(1)$
5. ☐ otra respuesta

Altura de B **arboles** Cual es {la altura, el tiempo de busqueda, el tiempo de insercion, el tiempo de delecion} de un B arbol sobre $n = \{8, 9, 256\}$ valores, si cada nodo contiene B valores?

1. ☐ n/B
2. ☐ $\lg n / \lg B$
3. ☐ $\log_B n$
4. ☐ $\log_n B$
5. ☐ otra respuesta

Relacion hijos/llaves en la raiz de un B **arboles** Si un nodo de un B arbol tiene d llaves, cuantos hijos tienes?

1. ☐ $d - 1$
2. ☐ d
3. ☐ $d + 1$
4. ☐ $2d + 1$
5. ☐ otra respuesta

Cantidad de llaves en un nodo de B arbol (Part 1) :CONTEXT: Una pagina de la memoria secundaria puede tener B valores juntas con $B+1$ punteros. :END: Cuantos hijos (d) puede tener un B arbol sobre $n >> B$ elementos?

1. □ $d \in [0, B/2]$
2. □ $d \in [1, B/2]$
3. □ $d \in [0, B]$
4. □ $d \in [1, B]$
5. □ $d \in [B/2, B]$
6. □ otra respuesta

Cantidad de llaves en un nodo de B arbol (Part 2) :CONTEXT: Una pagina de la memoria secundaria puede tener B valores juntas con $B+1$ punteros. :END: Una pagina de la memoria secundaria puede tener B valores juntas con $B+1$ punteros. La **raiz** de un B arbol sobre $n >> B$ elementos tiene d hijos. Cual es el domanio de valores posibles por d?

1. □ $d \in [0, B/2]$
2. □ $d \in [1, B/2]$
3. □ $d \in [0, B]$
4. □ $d \in [1, B]$
5. □ $d \in [B/2, B]$
6. □ otra respuesta

Cantidad de llaves en un nodo de B arbol (Part 3) :CONTEXT: Una pagina de la memoria secundaria puede tener B valores juntas con $B+1$ punteros. :END: Un nodo (**otro que la raiz**) en un B arbol sobre $n >> B$ elementos tiene d hijos. Cual es el domanio de valores posibles por d?

1. □ $d \in [0, B/2]$
2. □ $d \in [1, B/2]$
3. □ $d \in [0, B]$
4. □ $d \in [1, B]$
5. □ $d \in [B/2, B]$
6. □ otra respuesta

B^* arboles :CONTEXT: Un B^* arbol llena sus nodos hasta 2/3, en vez de 1/2 por los B arboles. :END: Cual es el objetivo de un B^* arbol (en comparacion con un B arbol)?

1. □ Reducir el tiempo de busqueda?
2. □ Reducir la cantidad de accesos al cache en busqueda?
3. □ Reducir la complejidad espacial?
4. □ Reducir la frecuencia de "Split/Merge"?
5. □ Practical [e.g. Optimizacion para data-set (de injeniero)]
6. □ otra respuesta

B^+ arboles :CONTEXT: En B^+ arboles, las hojas tienen punteros adicionales formando una caldena de todas las hojas. :END: Cual es el objetivo de un B^+ arbol (en comparacion con un B arbol)?

1. □ Optimizar la Busqueda Secuencial (adaptiva)?
2. □ Suportar otro tipos de consultas/busquedas?
3. □ Suportar la exportacion de las valors en tiempo razonable?
4. □ Practical (e.g. facilitar el back-up de base de datos)?
5. □ otra respuesta

vEB arboles vs AVL arboles, $(2,3)$ arboles y AVL Arboles :CANC: :CONTEXT: Fija k, $m = 2^k$, $M = 2^m$. Un vEB arbol de n elementos sobre $[0..M-1]$ tiene una raiz con

- $\sqrt{M}$ punteros a sus ninos $C[0..\sqrt{M}-1]$,
- dos valores min y max
- un otro vEB aux sobre $[0..\sqrt{M}-1]$

:END: Que **no** distingue los vEB arboles de las otras estructuras de arboles que conocen (e.g. B-arboles) para el ADT diccionario?

1. □ usa el dominio de las valores **para buscar**
2. □ el nodo contiene los elementos extremos (no medios como en un AVL)
3. □ supporta *FindNext* y *FindPrev*
4. □ sirven para colas de prioridades tambien
5. □ petmitten de optimizar mejor la memoria
6. □ otra respuesta

Altura de un vEB arbol :CONTEXT: Fija k, $m = 2^k$, $M = 2^m$. Un vEB arbol de n elementos sobre $[0..M-1]$ tiene una raiz con

- $\sqrt{M}$ punteros a sus ninos $C[0..\sqrt{M}-1]$,
- dos valores min y max
- un otro vEB aux sobre $[0..\sqrt{M}-1]$

:END: En cual clase asimptotica es { el tiempo de busqueda, de insercion, de delecion, la altura } de un vEB con n valores codificadas en m bits?

1. □ $O(\lg n)$
2. □ $O(\lg m)$
3. □ $O(\lg \lg n)$
4. □ $O(\lg \lg m)$
5. □ otra respuesta

vEB children :CONTEXT: Fija k, $m = 2^k$, $M = 2^m$. Un vEB arbol de n elementos sobre $[0..M-1]$ tiene una raiz con

- $\sqrt{M}$ punteros a sus ninos $C[0..\sqrt{M}-1]$,
- dos valores min y max
- un otro vEB aux sobre $[0..\sqrt{M}-1]$

:END: El valor $x \in]min, max[$ se encuentra en el nino $C[i]$ donde $i =$

1. □ $\frac{2^{m/2}(max-x)}{(max-min)}$
2. □ $\frac{2^{m-2}(max-x)}{(max-min)}$
3. □ $\frac{x}{2^{m/2}}$
4. □ $\frac{x}{2^{m-2}}$
5. □ otra respuesta

vEB aux :CONTEXT: Fija k, $m = 2^k$, $M = 2^m$. Un vEB arbol de n elementos sobre $[0..M-1]$ tiene una raiz con

- $\sqrt{M}$ punteros a sus ninos $C[0..\sqrt{M}-1]$,
- dos valores min y max
- un otro vEB aux sobre $[0..\sqrt{M}-1]$

:END: El role de aux es de memorisar cuales ninos son vacios. $j \in aux$ si y solamente si $T.C[j]$ es non vacio. Cual es el principal objetivo?

1. □ Optimizar Find
2. □ Optimizar Insert
3. □ Optimizar LookUp
4. □ Optimizar FindNext
5. □ otra respuesta

vEB Find Previous :CONTEXT: Fija k, $m = 2^k$, $M = 2^m$. Un vEB arbol de n elementos sobre $[0..M-1]$ tiene una raiz con

- $\sqrt{M}$ punteros a sus ninos $C[0..\sqrt{M}-1]$,
- dos valores min y max
- un otro vEB aux sobre $[0..\sqrt{M}-1]$

:END: La complejidade de Find Previous es en

1. □ $O(k)$
2. □ $O(2^k = m)$
3. □ $O(2^{2^{k-1}}) = \sqrt{M}$
4. □ $O(2^{2^k}) = M$
5. □ $O((\lg \lg M)^2))$
6. □ otra respuesta

vEB Insercion :CONTEXT: Fija k, $m = 2^k$, $M = 2^m$. Un vEB arbol de n elementos sobre $[0..M-1]$ tiene una raiz con

- $\sqrt{M}$ punteros a sus ninos $C[0..\sqrt{M}-1]$,
- dos valores min y max
- un otro vEB aux sobre $[0..\sqrt{M}-1]$

:END: Si un nino $C[i]$ es lleno antes de agregar un elemento x a dentro.

1. □ Split $C[i]$ en dos
2. □ Mudar algunos elementos de $C[i]$ a sus vecinos, y si no se puede a su padre, recursivamente
3. □ Crea un nuevo sobre arbol con una hoja.
4. □ Genera un error
5. □ otra respuesta

1.5. Colas de prioridad en memoria secundaria. Cotas inferiores.

1.5.1. MATERIAL a LEER

- Apuntes CC3001
  - http://www.dcc.uchile.cl/ bebustos/apuntes/cc3001/TDA/4http://www.dcc.uchile.cl/~bebustos/apuntes/cc30 (last accessed on *2011-04-13 Wed*)
- Colas de Prioridad
  - http://en.wikipedia.org/wiki/Priority_queuehttp://en.wikipedia.org/wiki/Priority_queue (last accessed on *2011-04-13 Wed*)
- Heaps
  - http://en.wikipedia.org/wiki/Heap_data_structurehttp://en.wikipedia.org/wiki/Heap_data_structure (last accessed on *2011-04-13 Wed*)
- B-Heap
  - http://en.wikipedia.org/wiki/B-heaphttp://en.wikipedia.org/wiki/B-heap (last accessed on *2011-04-13 Wed*)
- van Emde Boas Queues
- http://en.wikipedia.org/wiki/Van$_E mde_B oas_p riority_q ueuehttp$: $//en.wikipedia.org/wiki/Van\_Emde\_Boas\_priori$

1.5.2. APUNTES

1. Colas de Prioridades tradicional:
   - que se necesita? * Operadores
     - *insert(key,item)*
     - *findMind()*
     - *extractMin()*
   - *heapify*

- *increaseKey, decreaseKey*
- *find*
- *delete*
- *successor/predecessor*
- *merge*
- ...
- diccionarios: demasiado espacio para que se pide
  - menos operadores que diccionarios
    - =¿mas flexibilidad en la representación
    - =¿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**
  - Definición:
    "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 diccionarios es una cota inferior por colas de prioridades o no?
     - No. La reducción es en la otra dirección: una cota inferior por colas de prioridad impliqua una cota inferior por diccionarios.
   - Cual es la cota inferior mas simple que se puede imaginar?
     - $\Omega(n/B)$
   - La cota superior de $O(\log_B n)$ que da una estructura de diccionario, on un B-heap es optima o no?
     - en el modelo de comparacion, se puede mostrar una cota inferior de $\Omega(\log_B n)$

1.5.3. PREGUNTAS [3/10] :PREGUNTAS:

Cola de Prioridad: operaciones Cuales (no) son operaciones de una (min) cola de prioridad?

1. □ insert(key,item)
2. □ search(key)
3. □ delete(key)
4. □ findNext(key)
5. □ findPrevious(key)
6. □ findMin()
7. □ extractMin()

Colas de Prioridades contra Diccionarios Dado estructuras de datos C y D, respectivamente implementando los ADT "cola de prioridad" y "diccionario". Cual(es) de estas proposiciones tiene(n) problemas?

1. □ C implementa el ADT "diccionario" también.
2. □ D implementa el ADT "cola de prioridad" también.
3. □ C toma menos espacio que D
4. □ D es asintóticamente mas rápido que C (en los operadores que tienen en común)
5. □ ninguna

Colas de Prioridades contra Diccionarios Considera las estructuras de datos Heap C y AVL-árbol D, respectivamente implementando los ADT "cola de prioridad" y "diccionario". Cual(es) de estas proposiciones tiene(n) problemas?

1. □ C implementa el ADT "diccionario" también, pero en malo **tiempo**.
2. □ D implementa el ADT "cola de prioridad" también, pero en malo **tiempo**.
3. □ C implementa el ADT "diccionario" también, pero en malo **espacio**.
4. □ D implementa el ADT "cola de prioridad" también, pero en malo **espacio**.
5. □ C implementa el ADT "diccionario" también, pero en malo **tiempo y espacio**.
6. □ D implementa el ADT "cola de prioridad" también, pero en malo **tiempo y espacio**.
7. □ otra respuesta

Cola de Prioridad: Heapify El operador "Heapify"

1. □ es parte del ADT "colas de Prioridad"
2. □ es parte de la estructura de datos "Heap"
3. □ tiene complejidad $O(\lg n)$
4. □ tiene complejidad $O(n)$
5. □ tiene complejidad $O(n \lg n)$
6. □ otra respuesta

Estructuras de datos Çola de Prioridad

Cuales estructuras de datos "Cola de Prioridad" conocen?

1. ☐ **binary heap**
2. ☐ sequence-heaps
3. ☐ *binomial queues*
4. ☐ *Fibonacci heaps*
5. ☐ leftist heaps
6. ☐ min-max heaps
7. ☐ pairing heaps
8. ☐ skew heaps
9. ☐ *van Emde Boas queues*

Heaps en Memoria Segundaría: Find Min :CONTEXT: Considera un modelo de memoria segundaría con paginas de tamaño para contener B elementos, y un "heap" de n elementos ($n >> B$). :END: A cuantos accesos a la memoria secundaría corresponde un llamado a "FindMin" en un "min heap"?

1. 1 acceso
2. ☐ $\log_B n$ accesos
3. ☐ $\log n / \log B$ accesos
4. ☐ n/B accesos
5. ☐ n accesos

Heaps en Memoria Segundaría: Delete Min :CONTEXT: Considera un modelo de memoria segundaría con paginas de tamaño para contener B elementos, y un "binary min heap" de n elementos ($n >> B$). :END: A cuantos accesos a la memoria segundaría corresponde un llamado a "DeleteMin" en un "min heap"?

1. ☐ $\log_B n$ accesos
2. ☐ $(n - B)/B + 1$ accesos
3. ☐ n/B accesos
4. ☐ $n - B$ accesos
5. ☐ n accesos
6. ☐ otra respuesta

vEB queues: Delete Min :CONTEXT: Considera un modelo de memoria segundaría con paginas de tamaño para contener B elementos, y un "heap" de n elementos ($n >> B$). :END: A cuantos accesos a la memoria segundaría corresponde un llamado a "DeleteMin" en un vEB Queue?

1. ☐ $\log_B n$ accesos
2. ☐ $(n - B)/B + 1$ accesos
3. ☐ n/B accesos
4. ☐ $n - B$ accesos
5. ☐ otra respuesta

vEB queues: cantidad de hijos :CONTEXT: Considera un modelo de memoria segundaría con paginas de tamaño para contener B elementos, y un "vEB queue" de n elementos ($n >> B$). :END:
Cuanto hijos tiene la raíz de un vEB?

1. □ 2
2. □ B
3. □ $B + 1$
4. □ $\sqrt{B}$
5. □ $\sqrt{n}$
6. □ otra respuesta

vEB queues: altura :CONTEXT: Considera un modelo de memoria segundaría con paginas de tamaño para contener B elementos, y un "vEB queue" de n elementos ($n >> B$). :END: Cual es la altura de un vEB queue?

1. □ $\log_B \log_B n$
2. □ $\log_2 \log_2 n$
3. □ $\log_B n$
4. □ $\log_2 n$
5. □ otra respuesta

vEB queues: tiempo de búsqueda :CONTEXT: Considera un modelo de memoria segundaría con paginas de tamaño para contener B elementos, y un "vEB queue" de n elementos ($n >> B$). :END:
Cual es el tiempo de búsqueda ("findKey(k)") un vEB queue?

1. □ $\log_B \log_B n$
2. □ $\log_2 \log_2 n$
3. □ $\log_B n$
4. □ $\log_2 n$
5. □ otra respuesta

vEB queues: tiempo de "deleteMin
:CONTEXT: Considera un modelo de memoria segundaría con paginas de tamaño para contener B elementos, y un "vEB queue" de n elementos ($n >> B$). :END:
Cual es el tiempo de "deleteMin" en un vEB queue?

1. □ $\log_B \log_B n$
2. □ $\log_2 \log_2 n$
3. □ $\log_B n$
4. □ $\log_2 n$
5. □ otra respuesta

vEB queues: espacio :CONTEXT: Considera un modelo de memoria segundaría con paginas de tamaño para contener B elementos, y un "vEB queue" de n elementos ($n >> B$). :END:

Cuanto bytes toma un "vEB queue"?

1. □ n
2. □ $k = \lg m$
3. □ m
4. □ $M = 2^m$
5. □ otra respuesta

Cotas Inferiores en Memoria Secundaria :CONTEXT: :END:

Cual(es) de estas afirmaciones esta(n) correctas (en el modelo de comparacion)?

1. □ $\Omega(\log_B N)$ por colas de prioridades implicaria $\Omega(\log_B N)$ por diccionarios
2. □ $\Omega(\log_B N)$ por diccionarios implicaria $\Omega(\log_B N)$ por colas de prioridades
3. □ $\Omega(\log_B N)$ por colas de prioridades implicaria $\Omega(N \log_B N)$ por ordenamiento
4. □ $\Omega(N \log_B N)$ por ordenamiento implicaria $\Omega(\log_B N)$ por colas de prioridades
5. □ ninguna

1.5.4. REFERENCIAS ADICIONALES

- http://www.dcc.uchile.cl/ gnavarro/algoritmos/tesisRapa.pdfhttp://www.dcc.uchile.cl/g̃navarro/algoritmos/tesisR
  - paginas 9 hasta 16: para cotas inferiores y resultados experimentales.
- Otras referencias en http://www.leekillough.com/heaps/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=351827.384259http://portal.acm.org/citation.cfm?id=351827.384259

1.6. Ordenamiento en memoria secundaria: Mergesort. Cota inferior.

1.6.1. MATERIAL A LEER

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

1.6.2. APUNTES

- Un modelo mas fino quel anterior
  - Cuantos paginas quedan en memoria local?
    - no tan importante para busqueda
    - muy importante para applicaciones de computacion con mucho datos.
  - 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
      - $n << N$, $m << M$, por eso son "pequeñas" letras
  - En estas notaciones, usando resultos previos:
    * `Insertion Sort` (en un B-Arbol)
    - usa dictionarios en memoria externa
    - $N \lg N / \lg B = N \log_B N$
  - usa colas de prioridades en memoria externa
  - $N \lg N / \lg B = N \log_B N$
    * Eso es optimo o no?
- 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 $\lg N / \lg B = \log_B N$
- en modelo RAM?
  - N
- en modelo Memoria Externa con paginas de tamano B?
  - al maximo $N/B = n$
- 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.

- (corrige y adapte la prueba de http://www.daimi.au.dk/ large/ioS06/Alower.pdfhttp://www.daimi.au.dk/l̃arge/ioS0
)
- 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 arreglo 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.
- 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 $N!/((B!)^n\binom{M}{B}^t)$

  * 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.pdfhttp://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)

- $N \lg N / \lg B = N \log_B N$
- No es "ajustado" con la cota inferior
- impliqua

  - o que hay un mejor algoritmo
  - o que hay una mejor cota inferior

- usa la fusion de $m-1$ arreglos ordenados en memoria externa:

  1. carga en memoria principal $m-1$ paginas, cada una la primera de su arreglo.
  2. 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
  3. La complejidad es n accessos.

- Algoritmo:

1. ordena cada de las n paginas $\rightarrow n accessos Cada nodo calcula la union de m arreglos y escribe su resultado, pagina por pagi$

2. Analisis:

   1. Cada nivel de recurencia costa n accessos
   2. Cada nivel reduce por $m-1$ la cantidad de arreglos
   3. 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????

1.6.3. PREGUNTAS [6/9] :PREGUNTAS:

Effecto de M sobre *Find* La complejidad de *Find* en un B-arbol o vEB arbol es de $\lg_B N$ acesos a la memoria segundaria, con $M = 1$ paginas en memoria principal. Con M mas largo, este complejidad

1. ☐ se queda igual
2. ☐ baja a $\lg_B N/M$
3. ☐ baja a $\lg_{B/M}$ N $
4. ☐ baja a $\lg_B(N/M)$
5. ☐ Otra respuesta

Effecto de M sobre *FindMin* La complejidad de *Find* en un B-arbol o vEB arbol es de $\lg_B N$ acesos a la memoria segundaria, con $M = 1$ paginas en memoria principal. Con M mas largo, este complejidad

1. ☐ se queda igual
2. ☐ baja a $\lg_B N/M$
3. ☐ baja a $\lg_{B/M} N$
4. ☐ baja a $\lg_B(N/M)$
5. ☐ Otra respuesta

Complejidad de Insertion Sort en Memoria Segundaria :CONTEXT: Considera que

- 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

:END:
El algoritmo de `Insertion Sort`, con un B-arbol, permite de ordenar N elementos en

1. ☐ al menos $N \lg N / \lg B = N \log_B N$ accesos
2. ☐ exactamente $N \lg N / \lg B = N \log_B N$ accesos
3. ☐ al maximo $N \lg N / \lg B = N \log_B N$ accesos
4. ☐ menos que $N \lg N / \lg B = N \log_B N$ accesos
5. ☐ otra respuesta

Complejidad de Heap Sort en Memoria Segundaria :CONTEXT: Considera que

- 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

:END:
El algoritmo de `Heap Sort`, con un vEB-cola de prioridad, permite de ordenar N elementos en

1. ☐ al menos $N \lg N / \lg B = N \log_B N$ accesos
2. ☐ exactamente $N \lg N / \lg B = N \log_B N$ accesos
3. ☐ al maximo $N \lg N / \lg B = N \log_B N$ accesos
4. ☐ menos que $N \lg N / \lg B = N \log_B N$ accesos
5. ☐ otra respuesta

Cota inferior de ordenamiento en memoria segundaria :CONTEXT: Considera que

- 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

:END:

Cual de estas cotas inferiores para el problema de ordenar en memoria segundaria parece la mas razonable?

1. □ $\Omega(N/B\frac{\lg(N/B)}{\lg(M/B)})$
2. □ $\Omega(n \lg_m n)$
3. □ $\Omega(N \lg N / \lg B)$
4. □ $\Omega(N \log_B N)$
5. □ otra respuesta

Ordenar (a dentro de las) paginas :CONTEXT: Considera que

- 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
- un arreglo A contiene unas de las $N!$ permutaciones possibles sobre N elementos.

:END:

Cual es el costo asintótico (en cantidad de accesos a la memoria secundaria) de ordenar cada bloque (pagina) de B elementos?

1. □ $n = N/B$
2. □ $N = n \times B$
3. □ $n \times B \lg B$
4. □ $N \times B \lg B$
5. □ otra respuesta

La permutacion escrita Alguien elijo una permutacion muy grande sobre $[1..N]$, una de las $N!$ posibles. El entrega la primera cifra. Cuantas permutaciones posibles quedan?

1. □ $N!/(N-1)!$
2. □ $N!/(N-1)$
3. □ $N!/N$
4. □ $N!$
5. □ otra respuesta

Contando permutaciones (Part 1) :CONTEXT: Considera que

- 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
- un arreglo A contiene unas de las $N!$ permutaciones possibles sobre N elementos;

:END:

Cuantas posibilidades de permutaciones quedan en A despues de ordenar la primera pagina?

1. □ $n!/B!$
2. □ $N!/B!$
3. □ $N!/B \lg B$
4. □ $N!$
5. □ $(N-B)!
6. □ $N! - B!$
7. □ otra respuesta

Contando permutaciones (Part 2) :CONTEXT: Considera que

- 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
- un arreglo A contiene unas de las $N!$ permutaciones possibles sobre N elementos;
- alguien ya ordeno cada bloque de B elementos.

:END:

Cuantas posibilidades de permutaciones quedan en A despues de ordenar a dentro de las paginas?

1. □ $N!/(B!)^n$
2. □ $N!/n!$
3. □ $N!/B!$
4. □ $N!$
5. □ otra respuesta

Insertando B elementos en un arreglo ordenado de M elementos (Part 1) De cuantas maneras se pueden mezclar B valores en un arreglo de M valores?

1. □ $\binom{M}{B}$
2. □ $\frac{M!}{B!(M-B)!}$
3. □ $M \times (M-1) \times \ldots \times (M-B+1)$
4. □ $M \times (M-1) \times \ldots \times (M-B)$
5. □ otra respuesta

Insertando B elementos en un arreglo ordenado de M elementos (Part 2) Si tenemos X permutaciones posibles, que descubrimos las posiciones relatives de B nuevas valores en relacion con M valores en memoria primaria, cuantas permutaciones quedan?

1. □ $X/\binom{M}{B}$
2. □ $X/\frac{M!}{B!(M-B)!}$
3. □ $X/M \times (M-1) \times \ldots \times (M-B+1)$
4. □ $X/M \times (M-1) \times \ldots \times (M-B)$
5. □ otra respuesta

Insertando t veces B elementos a dentro de M elementos :CONTEXT: Considera que

- 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
- un arreglo A contiene unas de las $N!$ permutaciones possibles sobre N elementos;
- alguien ya ordeno cada bloque de B elementos,
- deseamos "descubrir" cual es la permutacion.

:END:

Después de t accessos (distintos) a la memoria externa, la cuantidad de permutaciones se reduci a

1. □ $N!/(B!)^t$
2. □ $N!/(B!)^n \binom{M}{B}^t$
3. □ \$N! / {M $\binom{}{B\}^t\$}$□ $N!/(N - B \times t)!$
4. □ otra respuesta

Cuantos acesos para reducir a una sola permutacion? Que agumento se usa para cada etapa del razonamiento siguente?

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

1. ☐ $n = N/B$ y $m = M/B$
2. ☐ $\lg x$ es cresciente
3. ☐ $\lg(x/y) = \lg x - \lg y$
4. ☐ $\lg(x!) \approx x \lg x$
5. ☐ $\lg \binom{M}{B} \approx B \lg \frac{M}{B}$
6. ☐ otra tecnica

Cota inferior ordenamiento en memoria segundaria Que significa que $t \geq n \log_m n$?

1. ☐ No se puede ordenar en menos que $n \log_m n$ comparaciones.
2. ☐ No se puede ordenar en menos que $n \log_m n$ acesos a la memoria segundaria.
3. ☐ Se puede ordenar en menos que $n \log_m n$ comparaciones.
4. ☐ Se puede ordenar en menos que $n \log_m n$ acesos a la memoria segundaria.
5. ☐ otra respuesta

Cota superior ordenamiento en memoria segundaria :CONTEXT: :END:

Existe un algoritmo que ordena N elementos (repartidos en n paginas de al maximo B elementos cada una) en $O(n \log_m n)$ acesos a la memoria segundaria?

1. ☐ No
2. ☐ Si, es una varianta de Merge Sort
3. ☐ Si, es una varianta de Insertion Sort
4. ☐ Si, es una varianta de Heap Sort
5. ☐ Otra Respuesta

Cantidad de memoria Local :CONTEXT: :END:

Dado un tamaño de pagina fijo B, en cual problema la cantidad M de memoria local (y la cantidad m de paginas que se pueden guardar en memoria local) affecta mas la complejidad asintótica?

1. ☐ *Find* en ADT Diccionario
2. ☐ *FindNext* en ADT Diccionario (e.g. B-arbol o van Emde Boas)
3. ☐ *FindMin* en ADT Cola de prioridad
4. ☐ *MergeSort*
5. ☐ todas iguales: mas memoria siempre ayuda.

1.7. RESUMEN de la Unidad 2

1.7.1. Objetivos

- 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.

1.7.2. Temas:

1. □ Memoria Secundaria
2. □ vEB diseno original ($\lg\lg m$ busqueda)
3. □ vEB diseno "cache-oblivious" ($\log_B n$ busqueda sin conocer B)
4. □ Diccionarios en Memoria Secundaria
5. □ Colas de Prioridades en Memoria Secundaria
6. □ Ordenamiento en Memoria Secundaria
7. □ Cotas Inferiores en Memoria Secundaria

1.7.3. Summary of vEB trees variants

	B-Arbol	recursive vEB	value based vEB	(otras)
Diccionario	$(2,3)$ generalized	un vEB a dentro de un vEB	un arreglo indexado por k/2 bits	$(\dots)$
Cola de prioridad	Heap Generalized	idem	idem	$(\dots)$
Propriedades	simple cuando B conocido	cache-oblivious (B desconocido)	tiempo $\lg\lg m$ (cuando $n \approx m$)	$(\dots)$

1.7.4. PREGUNTAS [0/10] :PREGUNTAS:

NEXT Peor Caso de Ïnsert.$^{\text{en}}$ Memoria Secundaria :CONTEXT: Considera un nivel de memoria tal que

- 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

:END:

Para cada de las estructuras de datos siguentes,

1. "min binary heap"
2. avl arbol
3. (2,3)-arbol
4. B-arbol para diccionario
5. $2B$-arbol para diccionario

6. $B/2$-arbol para diccionario
7. vEB-arbol original para colas de prioridades
8. vEB-arbol recursivo para colas de prioridades
9. vEB-arbol original para diccionario
10. vEB-arbol recursivo para diccionario

Cual es el rendimiento (asintótico), en terminos de accesos a la memoria secundaria en el peor caso, por un llamado a "Insert" (se acordan que la estructura de datos contiene N elementos)?

1. □ $\log_B \log_B N$
2. □ $\log_2 \log_2 N$
3. □ $\log_B N$
4. □ $\log_2 N$
5. □ otra respuesta

NEXT Mejor Caso de Ïnsert.[en] Memoria Secundaria :CONTEXT: Considera un nivel de memoria tal que

- 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

:END:
Para cada de las estructuras de datos siguentes,

1. "min binary heap"
2. avl arbol
3. (2,3)-arbol
4. B-arbol para diccionario
5. $2B$-arbol para diccionario
6. $B/2$-arbol para diccionario
7. vEB-arbol original para colas de prioridades
8. vEB-arbol recursivo para colas de prioridades
9. vEB-arbol original para diccionario
10. vEB-arbol recursivo para diccionario

Cual es el rendimiento, en terminos de accesos a la memoria secundaria en el **mejor** caso, por un llamado a "Insert" (se acordan que la estructura de datos contiene N elementos)?

1. □ $\log_B \log_B N$

2. ☐ $\log_2 \log_2 N$
3. ☐ $\log_B N$
4. ☐ $\log_2 N$
5. ☐ otra respuesta

NEXT Ordenamiento en Memoria Secundaria: Cota superior (Part 0) :CONTEXT: Considera un nivel de memoria tal que

- 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

:END:

Cual(es) de los algoritmos siguentes, en su varianta adaptada a la memoria secundaria, permite(n) de ordenar N elementos en $O(N \lg N)$ accesos a la memoria secundaria en el peor caso?

1. ☐ Insertion Sort
2. ☐ Merge Sort
3. ☐ Heap Sort
4. ☐ Bubble Sort
5. ☐ otra respuesta

NEXT Ordenamiento en Memoria Secundaria: Cota superior (Part 1) :CONTEXT: Considera un nivel de memoria tal que

- 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

:END:

Cual(es) de los algoritmos siguentes, en su varianta adaptada a la memoria secundaria, permite(n) de ordenar N elementos en $O(N \log_B N)$ accesos a la memoria secundaria en el peor caso?

1. ☐ Insertion Sort
2. ☐ Merge Sort
3. ☐ Heap Sort
4. ☐ Bubble Sort
5. ☐ otra respuesta

NEXT Ordenamiento en Memoria Secundaria: Cota superior (Part 2) :CONTEXT: Considera un nivel de memoria tal que

- 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

:END:

Cual(es) de los algoritmos siguentes, en su varianta adaptada a la memoria secundaria, permite(n) de ordenar N elementos en $O(n \log_m n)$ accesos a la memoria secundaria en el peor caso?

1. □ Insertion Sort
2. □ Merge Sort
3. □ Heap Sort
4. □ Bubble Sort
5. □ otra respuesta

NEXT Torneo Vencedor: Cota inferior en el peor caso :CONTEXT: El torneo internacional de Karate se tiene en una isla con capacidad por $M = 20$ participantes. Una sola nave puede traer los $N = 200$ participantes, que pudede transportar $B = 5$ participantes al mismo tiempo. Se supone que los niveles de los participantes corresponden a un orden total, de manera a ce que se pueden ordenar completamente.

- $M = 20$
- $N = 200$
- $B = 5$

:END:

Cuantos viajes de la nave se necessitan en total para identificar el gañador del torneo, en el peor caso?

1. □ $\log_B N$
2. □ N/B
3. □ $N \log_B N$
4. □ $N/B + N \log_B N$
5. □ otra respuesta

NEXT Torneo Vencedor: Cota superior en mejor caso :CONTEXT: El torneo internacional de Karate se tiene en una isla con capacidad por $M = 20$ participantes. Una sola nave puede traer los $N = 200$ participantes, que pudede transportar $B = 5$ participantes al mismo tiempo. Se supone que los niveles de los participantes corresponden a un orden total, de manera a ce que se pueden ordenar completamente.

- $M = 20$
- $N = 200$
- $B = 5$

:END:

Cuantos viajes de la nave se necessitan en total para identificar el gañador del torneo, en el **mejor** caso?

1. □ $\log_B N$
2. □ N/B
3. □ $N \log_B N$
4. □ $N/B + N \log_B N$
5. □ otra respuesta

NEXT Torneo Orden: Cota inferior (Part 1) :CONTEXT: El torneo internacional de Karate se tiene en una isla con capacidad por $M = 20$ participantes. Una sola nave puede traer los $N = 200$ participantes, que pudede transportar $B = 5$ participantes al mismo tiempo. Se supone que los niveles de los participantes corresponden a un orden total, de manera a ce que se pueden ordenar completamente.

- $M = 20$
- $N = 200$
- $B = 5$

:END:

Cuantos viajes de la nave se necessitan en total para identificar el orden total del torneo, en el peor caso?

1. □ $\log_B N$
2. □ N/B
3. □ $N \log_B N$
4. □ $N/B + N \log_B N$
5. □ otra respuesta

NEXT Torneo Orden: Cota inferior (Part 2) :CONTEXT: El torneo internacional de Karate se tiene en una isla con capacidad por $M = 20$ participantes. Una sola nave puede traer los $N = 200$ participantes, que pudede transportar $B = 5$ participantes al mismo tiempo. Se supone que los niveles de los participantes corresponden a un orden total, de manera a ce que se pueden ordenar completamente.

- $M = 20$
- $N = 200$
- $B = 5$

:END:

Cuantos viajes de la nave se necessitan en total para identificar el orden total sobre los $M = 20$ mejores participantes del torneo, en el peor caso?

1. □ $\log_B N$
2. □ N/B
3. □ $N \log_B N$
4. □ $N/B + N \log_B N$
5. □ otra respuesta

NEXT Torneo Orden: Cota inferior (Part 3) :CONTEXT: El torneo internacional de Karate se tiene en una isla con capacidad por $M = 20$ participantes. Una sola nave puede traer los $N = 200$ participantes, que pudede transportar $B = 5$ participantes al mismo tiempo. Se supone que los niveles de los participantes corresponden a un orden total, de manera a ce que se pueden ordenar completamente.

- $M = 20$
- $N = 200$
- $B = 5$

:END:

Cuantos viajes de la nave se necessitan en total para identificar el orden total sobre los $2M = 40$ mejores participantes del torneo, en el peor caso?

1. □ $\log_B N$
2. □ N/B
3. □ $N \log_B N$
4. □ $N/B + N \log_B N$
5. □ otra respuesta