Diseño y Análisis de Algoritmos - Preguntas

Jérémy Barbay

15 de marzo de 2012

Índice

1. How to Include Concept Questions inside your lecture notes

2. Introduciendo "Concept questions": "Hanoi Tower" y "Disk Pile"

2.1. Una corta historia sobre "Concept Questions" :TALK:

- Charla de Eric Mazur:
  - http://www.youtube.com/watch?v=WwslBPj8GgI
- Grupo de Investigacion sobre "Peer Instruction"
  - http://mazur-www.harvard.edu/research/detailspage.php?rowid=8

2.2. Torre de Hanoi de altura 4 :PREGUNTAS:

What is the minimum number of moves required to move a Hanoi Tower of height 4?

1. ☐ 4
2. ☐ $4 * \lg 4 = 4 * 2 = 8$
3. ☐ $4! = 4 * 3 * 2 * 1 = 24$
4. ☐ $2^4 = 32$
5. ☐ none of the above

2.3. Torre de Hanoi de altura 8 :PREGUNTAS:

What is the minimum number of moves required to move a Hanoi Tower of height 8?

1. □ 8
2. □ $8 * \lg 8 = 8 * 3 = 24$
3. □ $2^8 = 254$
4. □ $8! = 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 40320$
5. □ none of the above

2.4. Disk Pile of height 8 :PREGUNTAS:

What is the minimum number of moves required to move a disk pile of height 8?

1. □ 8
2. □ $8 * \lg 8 = 8 * 3 = 24$
3. □ 2^8
4. □ $8! = 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 =?$
5. □ none of the above

2.5. Disk Pile of height 8 with 2 disk sizes :PREGUNTAS:

What is the minimum number of moves required to move a disk pile of height 8 in the worst case over the instances with exactly two distinct sizes of disc?

1. □ 8
2. □ $8 * \lg 8 = 8 * 3 = 24$
3. □ 2^8
4. □ $8! = 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 =?$
5. □ none of the above

3. Revisiones de CC3001

3.1. Asintóticas :CP:

$f(n)$	$g(n)$	$f(n) \in O(g(n)$	$f(n) \in \Omega(g(n)$	$f(n) \in Theta(g(n)$
$3n+6$	$100n-50$			
$n^{\frac{1}{2}}$	$n^{\frac{2}{3}}$			

3.2. ¿Cuántos árboles binarios distintos se pueden construir con 3 nodos internos?

1. □ 1
2. □ 3
3. □ 4
4. □ 6
5. □ otra

3.3. Arboles Binarios, nodos internos externos

Si se define i = número de nodos internos, e = número de nodos externos, entonces se tiene que:

1. ☐ i = e
2. ☐ e = i+1
3. ☐ i = e+1
4. ☐ e = 2^i
5. ☐ sin relación

3.4. Sea n = número de nodos internos. Se define:

- In = suma del largo de los caminos desde la raíz a cada nodo interno (largo de caminos internos).
- En = suma del largo de los caminos desde la raíz a cada nodo externo (largo de caminos externos). Se tiene que:
    1. ☐ En = In
    2. ☐ En = In+1
    3. ☐ En = In+n
    4. ☐ En = In+2n
    5. ☐ sin relación

3.5. Heap

La característica que permite que un heap se pueda almacenar sin punteros es que, si se utiliza la numeración por niveles indicada, entonces la(s) relación(es) entre padres e hijos es (son):

1. ☐ Hijos del nodo $j = \{2 * j, 2 * j + 1\}$
2. ☐ Padre del nodo $k = \lfloor k/2 \rfloor$
3. ☐ Hijos del nodo $j = \{2 * j - 1, 2 * j\}$
4. ☐ Padre del nodo $k = \lfloor k/2 \rfloor + 1$
5. ☐ ningunos

3.6. AVL

La altura de un AVL con n elementos es

1. ☐ $\log_\phi(n + 1) + \Theta(1)$
2. ☐ en $O(\lg n)$
3. ☐ en $\Omega(\lg n)$
4. ☐ en $\Theta(\lg n)$
5. ☐ ningunos o mas que dos

3.7. AVL $h- > n$

para una altura h dada, cuantos nodos tiene un árbol AVL con **mínimo** número de nodos que alcanza esa altura?

1. □ h
2. □ $2h$
3. □ 2^h
4. □ $2^h - 1$
5. □ otra respuesta

4. Cotas Superiores/Inferiores

4.1. Cota superior de (la complejidad de) Max ord

Dado un arreglo ordenado de n enteros, en cuanto accessos al arreglo pueden calcular su valor maximal?

1. □ 0
2. □ 1
3. □ $n - 1$
4. □ n
5. □ otra

4.2. Definicion de la mediana

Dado un arreglo de n enteros, cual es la definicion correcta de la mediana?

1. □ El promedio de las valores minima y maxima del arreglo.
2. □ La valor en el centro del arreglo.
3. □ La valor en el centro del arreglo ordenado.
4. □ La valor superior a $\lceil (n-1)/2 \rceil$ valores y inferior a $\lfloor (n-1)/2 \rfloor$ valores.
5. □ otra respuesta.

4.3. Dificultad de problemas en arreglos

Dado un arreglo de n enteros, cual problema requiere mas accessos al arreglo? Mas computacion?

1. □ Calcular la valor minima
2. □ Calcular la valor maxima
3. □ Calcular la valor mediana
4. □ Calcular la valor promedia
5. □ Son todos iguales

4.4. Cota Inferior para Max

Dado un arreglo de n enteros, cuanto comparaciones entre los elementos del arreglo se necessitan para calcular su valor maximal?

1. ☐ 0
2. ☐ 1
3. ☐ $n - 1$
4. ☐ n
5. ☐ otra respuesta

4.5. Definicion del problema de MinMax

Dado un arreglo A de n enteros, cual es la definicion del problema de "minmax"?

1. ☐ calcular $\min_{i\in[1..n],j\in[i..n]} A[i]$
2. ☐ calcular $\min_{i\in[1..n]} \max j \in [i..n]A[i]$
3. ☐ calcular $(\min_{i\in[1..n]} A[i], \max_{i\in[1..n]} A[i])$
4. ☐ calcular $(\min_{i\in[1..n]} A[i], \max_{j\in[1..n]} A[j])$
5. ☐ otra respuesta

4.6. Cotas de (la complejidad de) problemas combinados

Dado dos problemas A y B (e.g. min y max), cada uno con un algoritmo que le resuelve optimalemente con complejidad $f_A(n)$ y $f_B(n)$, cual es la complejidad del problema AB (e.g. min max)?

1. ☐ mín{$f_A(n)$, f$_B$(n)}$
2. ☐ $f_A(n)$ + f$_B$(n)$
3. ☐ ($f_A(n)$ + f$_B$(n))/2$
4. ☐ máx{$f_A(n)$, f$_B$(n)}$
5. ☐ otra respuesta

4.7. Cota superior de (la complejidad de) Min Max

Dado un arreglo de n enteros, en cuanto comparaciones (cantidad exacta, no asimptotica) entre los elementos del arreglo pueden calcular su valor maximal y minimal?

1. ☐ $n - 1$
2. ☐ $3n/2 - 2$ si n es par, $3n/2 + 1/2$ si n es impar.
3. ☐ $(n - 1) + (n - 2)$
4. ☐ $2(n - 1)$
5. ☐ otra respuesta

4.8. Cota inferior de (la complejidad de) Min Max

Dado un arreglo de n enteros, cuanto comparaciones (cantidad exacta, no asimptotica) entre los elementos del arreglo se necessitan para calcular su valor maximal y minimal?

1. ☐ $n-1$
2. ☐ $\lceil 3n/2 \rceil - 2$
3. ☐ $(n-1)+(n-2)$
4. ☐ $2(n-1)$
5. ☐ otra respuesta

4.9. Juego de las preguntas, $n = 4$

Cuanta preguntas (e.g. "$x < 4$?", "x=2"?) se necesitan para adivinar un entero entre 1 y $4 (i.e. $x \in$ [1.,4])?

1. ☐ 1
2. ☐ 2
3. ☐ 3
4. ☐ 4
5. ☐ otra

4.10. Juego de las preguntas, $n = 1024$

Cuanta preguntas (e.g. "$x < 10$?", "x=10"?) se necesitan para adivinar un entero entre 1 y 1024?

1. ☐ 8
2. ☐ 9
3. ☐ 10
4. ☐ 11
5. ☐ otra

4.11. Codificacion de un simbolo

Dado 1 simbolo elegido a dentro de $[1..\sigma]$

1. ☐ no se puede codificar **nunca** en $o(\lg \sigma)$ bits
2. ☐ no se puede codificar **siempre** en $o(\lg \sigma)$ bits
3. ☐ no se sabe **como codificar siempre** en $o(\lg \sigma)$ bits
4. ☐ no se sabe **si nunca se puede codificar** en $o(\lg \sigma)$ bits
5. ☐ otra

4.12. Definicion de un arbol de decision

Un arbol de decision es definido como un arbol

1. □ modelisando algoritmos en el modelo de comparacion.
2. □ binario donde cada hoja identifica una instancia.
3. □ binario donde cada nodo prueba una caracteristica de la instancia.
4. □ un arbol de grado finito donde cada hoja indica una decision sobre la instancia.
5. □ otra.

4.13. Codificacion de n simbolos

Dado n simbolos elegido a dentro de un alfabeto de tamaño σ

1. □ no se puede codificar **nunca** en $o(n \lg \sigma)$ bits
2. □ no se puede codificar **siempre** en $o(n \lg \sigma)$ bits
3. □ no se sabe **como codificar siempre** en $o(n \lg \sigma)$ bits
4. □ no se sabe **si nunca se puede codificar** en $o(n \lg \sigma)$ bits
5. □ otra

4.14. Definicion de "InsertionRank"

Dado un arreglo ordenado $A[1..n]$ de n valores y una valor x, cual(es) de estas definiciones del *Posicion de Insercion* ("Insertion Rank") de x en A son incorectas? ($A[0] = -\infty$ y $A[n+1] = +\infty$)

1. □ la posicion en cual x deberia ser insertado por dejar A ordenado
2. □ el entero $p \in [1..n+1]$ tal que $A[p-1] < x \leq A[p]$
3. □ el entero $p \in [0..n]$ tal que $A[p] \leq x < A[p+1]$
4. □ el entero $p \in [1..n]$ tal que $x = A[p]$
5. □ ningunos o mas que dos

4.15. Dos tipos de busqueda ordenada

Dado el codigo siguente, cual es la mejor manera de completarlo para minimizar la complejidad (non asymptotica) en el peor caso? El el caso promedio?
insertionRank(x,A,l,r) { if($r - l < 2$) return l else { m=(l+r)/2; ... } }

1. □ if($x < A[m]$) return insertionRank(x,A,l,m) else if($x > A[m]$) return insertionRank(x,A,m,r) else if($x = A[m]$) return m endif
2. □ if($x = A[m]$) return m else if($x < A[m]$) return insertionRank(x,A,l,m) else if($x > A[m]$) return insertionRank(x,A,m,r) endif
3. □ if($x = A[m]$) return m else if($x < A[m]$) return insertionRank(x,A,l,m) else return insertionRank(x,A,m,r) endif
4. □ if($x < A[m]$) return insertionRank(x,A,l,m) else return insertionRank(x,A,m,r) endif
5. □ performan iguales todos en el peor caso.

4.16. Cota inferior por busqueda ordenada $n = 1024$.

Dado un arreglo ordenado A de 1024 enteros y un entero x, cuanto comparaciones con elementos del arreglo son necesarias para decidir si x pertenece a A (en el peor caso)?

1. ☐ 9
2. ☐ 10
3. ☐ 11
4. ☐ 1024
5. ☐ otra

4.17. Cota inferior por busqueda ordenada general n.

Dado un arreglo ordenado A de n enteros y un entero x, cuanto comparaciones con elementos del arreglo son necesarias para decidir si x pertenece a A (en el peor caso)?

1. ☐ $\lceil \lg n \rceil$
2. ☐ $1 + \lceil \lg n \rceil$
3. ☐ $n - 1$
4. ☐ n
5. ☐ otra

4.18. Definicion del modelo de comparacion

Cuales de estos algoritmos simples son en el modelo de comparacion?

1. ☐ c=0; for(int i=1; i<n; i++) { if(A[i]>A[i+1]) c++;}
2. ☐ for(int i=1; i<n; i++) { if(A[i]>A[i+1]) print i;}

; 3. [ ] for(int i=1; i<n; i++) { if(A[i]>A[i+1]) print i;} ; 4. [ ] for(int i=1; i<n; i++) { if(A[i]>A[i+1]) print i;}

1. ☐ ningunos

4.19. Relacion entre codificacion y busqueda

Cual de estas aserciones es falsa en el modelo de comparacion?

1. ☐ A cada algoritmo de busqueda corresponde una codificacion de enteros.
2. ☐ A cada codificacion de enteros corresponde un algoritmo de busqueda.
3. ☐ A algunos algoritmos de busqueda corresponde una codificacion de enteros
4. ☐ A algunas codificaciones de enteros corresponde un algoritmo de busqueda.
5. ☐ otra

4.20. Busqueda Doblada

Cual de las asercions siguentes son falsas? Dado una valor x y un arreglo ordenado A de n valores, existe un algoritmo calculando la posicion de inserción p de x en A en

1. □ $\lg(1+n)$ comparaciones
2. □ $p+1$ comparaciones
3. □ $2\lg p$ comparaciones
4. □ $2\lg(n-p)$ comparaciones
5. □ ningunas o mas que dos.

4.21. Compression de enteros

Dado un entero $x \in [1..n]$, existe un esquema de codificacion representando x con

1. □ $\lg n$ bits,
2. □ $2\lg p$ bits,
3. □ p bits,
4. □ $2\lg(n-p)$ bits,
5. □ ningunas o mas que dos.

4.22. Cota inferior ordenamiento (en el modelo de comparacion)

Decir que "Ordenar es en $\Omega(n\lg n)$ (en el modelo de comparacion) significa que

1. □ no se puede ordenar en $o(n\lg n)$ comparaciones
2. □ ninguno algoritmo conocido (del modelo de comparacion) ordena en $o(n\lg n)$ comparaciones
3. □ no se puede ordenar en tiempo $o(n\lg n)$
4. □ ninguno algoritmo conocido (del modelo de comparacion) ordena en tiempo $o(n\lg n)$
5. □ otra respuesta

4.23. Complejidad en promedio de un algoritmo

Dado un entero fijado n, un algoritmo deterministico A. Cual de estas definiciones corresponde a la complejidad en promedio de A?

1. □ $\sum_{x,|x|=n} C(A,x)/n$
2. □ $\sum_{x,|x|=n} C(A,x)/2^n$
3. □ $\sum_{x,|x|=n} C(A,x)/\#\{x,|x|=n\}$
4. □ El promedio de su complejidad sobre cada instancia.
5. □ ningunas o mas de dos.

4.24. Complejidad en promedio de un problema

Dado un problema Pb, un entero fijado n, un conjunto $X_n = (x_i)_{i\in[1.,2^n]}$ de instancias legales por Pb y una distribucion $(p_i)_{i\in[1.,2^n]}$ sobre X_n. La complejidad en promedio de Pb es

1. □ $\max_A \sum_i p_i C(A, x_i)$
2. □ $\min_A \sum_i p_i C(A, x_i)$
3. □ $\sum_i p_i \max_A C(A, x_i)$
4. □ $\sum_i p_i \min_A C(A, x_i)$
5. □ ningunas

4.25. Complejidad aleatorizada

1. □

4.26. Relacion entre Complejidad en Promedio y en el peor caso

- Nota
  - $C(A, I)$ la complejidad de un algoritmo A sobre la instancia I, y
  - $E_I(C(A, I))$ la complejidad en el peor caso sobre las instancias de tamano n, y
  - $E_I(C(A, I))$ la complejidad en promedio por la distribucion uniforme sobre las instancias de tamano n.
- Cuales de estas relaciones son verdad?
  1. □ $E_I(C(A, I)) \leq \max_I C(A, I)$
  2. □ $E_I(C(A, I)) < \max_I C(A, I)$
  3. □ La complejidad en el peor caso (de un algoritmo) es siempre peor que la complejidad en promedio
  4. □ La complejidad en promedio (de un algoritmo) nunca es peor que la complejidad en el peor caso
  5. □ ningunas

4.27. Tecnicas de cotas inferiores

Cual(es) de las tecnicas siguentes permitten de mostrar cotas inferiores para la complejidad en promedio?

1. □ lemma del ave
2. □ Estrategia de Adversario
3. □ Arbol Binario de Decision
4. □ lemma del minimax
5. □ ningunas o mas de dos.

5. Experimentacion

5.1. Cuantos segundos vale 1ns?

Cuántos segundos vale un nano segundo?

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

5.2. Camino de acceso 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

5.3. 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 segundo
3. ☐ $4 * 10^6$ instrucciones per segundo
4. ☐ $4 * 10^9$ instrucciones per segundo
5. ☐ otra respuesta

5.4. 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

5.5. 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

5.6. Cuanto se demora una instruccion?

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

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

5.7. Costo (en plata) de la memoria

6. Memoria Externa

6.1. Cola de Prioridad: operaciones

Cuáles (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()

6.2. $(2,3)$ arboles

Cuál es el orden de {la altura, el tiempo de búsqueda, el tiempo de inserción, el tiempo de eliminación} de un $(2,3)$-árbol 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

6.3. $(2,3)$ arboles con $n \in \{8, 9, 256\}$ elementos :CANC:

Cuál es {la altura, el tiempo de búsqueda, el tiempo de inserción, el tiempo de eliminación} de un $(2,3)$-árbol 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

6.4. B arboles vs $(2,3)$ arboles

Cuál es {la altura, el tiempo de búsqueda, el tiempo de inserción, el tiempo de eliminación} de un B-árbol 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

6.5. Altura de B arboles

Cuál es {la altura, el tiempo de búsqueda, el tiempo de inserción, el tiempo de eliminación} 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

6.6. Relacion hijos/llaves en la raiz de un B arboles

Si un nodo de un B arbol tiene d llaves, cuántos hijos tiene?

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

6.7. Cantidad de llaves en un nodo de B arbol (Part 1)

Cuántos 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

6.8. Cantidad de llaves en un nodo de B arbol (Part 2)

Una página de la memoria secundaria puede tener B valores juntos con $B + 1$ punteros. La **raíz** de un B-Arbol sobre $n >> B$ elementos tiene d hijos. Cuál es el dominio 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

6.9. Cantidad de llaves en un nodo de B arbol (Part 3)

Un nodo (**distinto de la raiz**) en un B-Arbol sobre $n >> B$ elementos tiene d hijos. Cual es el dominio 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

6.10. B^* arboles

Cuál es el objetivo de un B^*-Arbol (en comparación con un B-Arbol)?

1. ☐ Reducir el tiempo de búsqueda?
2. ☐ Reducir la cantidad de accesos al cache en búsqueda?
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

6.11. B^+ arboles

Cuál es el objetivo de un B^+-Arbol (en comparación con un B-Arbol)?

1. □ Optimizar la Busqueda Secuencial (adaptativa)?
2. □ Suportar otro tipos de consultas/búsquedas?
3. □ Suportar la exportación de los valores en tiempo razonable?
4. □ Practical (e.g. facilitar el back-up de base de datos)?
5. □ otra respuesta

6.12. vEB arboles vs AVL arboles, $(2,3)$ arboles y AVL Arboles :CANC:

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 los 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 también
5. □ petmiten de optimizar mejor la memoria
6. □ otra respuesta

6.13. Altura de un vEB arbol

En cual clase asintótica está { el tiempo de búsqueda, de inserción, de eliminación y 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

6.14. vEB children

El valor $x \in]min, max[$ se encuentra en el hijo $C[i]$ 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

6.15. vEB aux

El rol de *aux* (o *summary*, como fue visto en la auxiliar) es de memorizar cuales hijos están vacíos. $j \in aux$ si y sólo si $T.C[j]$ es no vacío. Cuál es el principal objetivo de esto?

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

6.16. vEB Find Previous

La complejidade de Find Previous está 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

6.17. vEB Insercion

Si un hijo $C[i]$ está lleno antes de agregar un elemento x adentro.

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 árbol con una hoja.
4. ☐ Genera un error
5. ☐ otra respuesta

6.18. 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()

6.19. 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

6.20. 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

6.21. 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

6.22. Estructuras de datos “Cola 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*

6.23. Heaps en Memoria Segundaría: Find Min

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

6.24. Heaps en Memoria Segundaría: Delete Min

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

6.25. vEB queues: Delete Min

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

6.26. vEB queues: cantidad de hijos

Cuanto hijos tiene la raíz de un vEB?

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

6.27. vEB queues: altura

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

6.28. vEB queues: tiempo de búsqueda

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

6.29. vEB queues: tiempo de “deleteMin”

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

6.30. vEB queues: espacio

Cuanto bytes toma un “vEB queue”?

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

6.31. Cotas Inferiores en Memoria Secundaria

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

6.32. 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$_{\mathrm{B/M}}$ N \$
4. ☐ baja a $\lg_B(N/M)$
5. ☐ Otra respuesta

6.33. 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$_{\mathrm{B/M}}$ N \$
4. ☐ baja a $\lg_B(N/M)$
5. ☐ Otra respuesta

6.34. Complejidad de Insertion Sort en Memoria Segundaria

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

6.35. Complejidad de Heap Sort en Memoria Segundaria

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

6.36. Cota inferior de ordenamiento en memoria segundaria

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

6.37. Ordenar (a dentro de las) paginas

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

6.38. 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

6.39. Contando permutaciones (Part 1)

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

6.40. Contando permutaciones (Part 2)

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

6.41. 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

6.42. Insertando B elementos en un arreglo ordenado de M elementos (Part 2)

Si tenemos X permutaciones posibles, que descubrimos las posiciones relativas 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

6.43. Insertando t veces B elementos a dentro de M elementos

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 $\left(_{B\}^{t\$}}\right)$☐ $N!/(N-B\times t)!$
4. ☐ otra respuesta

6.44. 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

6.45. 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

6.46. Cota superior ordenamiento en memoria segundaria

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

6.47. Cantidad de memoria Local

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.

6.48. Peor Caso de "Insert" en Memoria Secundaria

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"? (recuerde 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

6.49. Mejor Caso de “Insert” en Memoria Secundaria (Part 1)

- $\log_B \log_B N$
- $\log_2 \log_2 N$
- $\log_B N$
- $\log_2 N$
- otra respuesta (constante)
  - “min binary heap”
  - avl arbol
  - (2,3)-arbol
  - B-arbol para diccionario
  - $2B$-arbol para diccionario
  - $B/2$-arbol para diccionario
  - vEB-arbol original para colas de prioridades
  - vEB-arbol recursivo para colas de prioridades
  - vEB-arbol original para diccionario

: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” ?

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

6.50. Mejor Caso de "Insert" en Memoria Secundaria (Part 2)

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" **con un nuevo elemento**?

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

6.51. Ordenamiento en Memoria Secundaria: Cota superior (Part 0)

Cual(es) de los algoritmos siguentes, en su variante 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

6.52. Ordenamiento en Memoria Secundaria: Cota superior (Part 1)

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

6.53. Ordenamiento en Memoria Secundaria: Cota superior (Part 2)

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

6.54. Torneo Vencedor: Cota inferior en el peor caso

Cuantos viajes de la nave se necessitan en total para identificar el ganador 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

6.55. Torneo Vencedor: Cota superior en mejor caso

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

6.56. Torneo Orden: Cota inferior (Part 1)

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

6.57. Torneo Orden: Cota inferior (Part 2)

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

6.58. Torneo Orden: Cota inferior (Part 3)

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

7. Analisis Amortizada

7.1. Analisis Amortizada: arreglo dinamico (Part 1)

Queremos implementar una pila ("stack") en un arreglo. Iniciamos con un arreglo de tamaño $s = 1$, y cuando se llena, creamos un arreglo mas grande, copiamos todo en en nuevo arreglo y sigamos.

Cual es el costo amortizado de una insercion si el nuevo arreglo es de tamaño $n + 1$?

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

7.2. Analisis Amortizada: arreglo dinamico (Part 2)

Cual es el costo amortizado de una insercion si el nuevo arreglo es de tamaño $2n$?

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

7.3. Analisis Amortizada: arreglo dinamico (Part 3)

Cual es el costo amortizado de una insercion si el nuevo arreglo es de tamaño $4n$?

1. ☐ menos que 2
2. ☐ 2
3. ☐ entre 2 y 3
4. ☐ 3
5. ☐ mas que 3

7.4. Analisis Amortizada: arreglo dinamico (Part 4)

Cual es el costo amortizado de una insercion si el nuevo arreglo es de tamaño n^2 (y el primero arreglo de tamaño 2)?

1. ☐ $O(1)$
2. ☐ $O(\lg n)$
3. ☐ $O(n)$
4. ☐ $O(n^2)$
5. ☐ otra respuesta