CC4102/CC40A/CC53A - Diseño y Análisis de Algoritmos

Jérémy Barbay

23 March 2011

Contents

1 1. Conceptos basicos y complejidad (3 semanas = 6 charlas)

1.1 DESCRIPCION de la Unidad:

1.1.1 Resultados de Aprendisajes de la Unidad

- Comprender el concepto de complejidad de un problema como cota inferior
- conocer tecnicas elementales para demostrar cotas inferiores
- Conocer algunos casos de estudio relevantes
- Adquirir nociones basicas de experimentacion en algoritmos.

1.1.2 Principales casos de estudio

- Cota inferior para minimo y maximo de un arreglo
- Caso promedio del quicksort
- Cota inferior para busqueda en un arreglo con distintas probabilidades de acceso

1.2 PREREQUISITOS DE UNIDAD

1.2.1 Lista de themas en apuntes de CC3001 :READING:

http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/

1.3 1.1 Repaso del proceso de diseño y analisis de un algoritmo.

1.3.1 PREREQUISITOS

- "Bases de la programacion" en apuntes de CC3001 **:READING:**
  - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/
- Notaciones asimptoticas **:READING:**
  - $O()$ http://es.wikipedia.org/wiki/Cota_superior_asint%C3%B3tica
  - $\Omega()$ http://es.wikipedia.org/wiki/Cota_inferior_asint%C3%B3tica

  - $\Theta()$ http://es.wikipedia.org/wiki/Cota_ajustada_asint%C3%B3tica
  - $o()$ (y $O()$) http://es.wikipedia.org/wiki/Notaci%C3%B3n_de_Landau
  - $\omega()$
  - $\theta()$
- AVL en apuntes de CC3001 **:READING:**
  - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Diccionario/#3

1.3.2 Problema de la Torre de Hanoi :TALK:

- Definicion
- Cota superior
- Cota inferior

1.3.3 Variantes de la Torre de Hanoi :TALK:

- pesos distintos?
- Disk Pile Problem (n discos pero $h < n$ tamanos)
  - Cota inferior en funcion de n, en funcion de n, h
  - Cota superior en funcion de n, en funcion de n, h
- Precise Disk Pile Problem (n discos con n_i de tamaño i, $\forall i \in [1..h]$)
  - cota inferior en función de n, en función de $h, n_1, ..., n_h$
  - cota superior en función de n, en función de $h, n_1, ..., n_h$

1.3.4 CONCEPT QUESTIONS [7/7] :CQ:

- **DONE** Asymptotics **:CP:**

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

- **DONE** ¿Cuántos árboles binarios distintos se pueden construir con 3 nodos internos?
  1. ☐ 1
  2. ☐ 3
  3. ☐ 4
  4. ☐ 6
  5. ☐ otra

- **DONE** 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 relacion

- **DONE** 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 relacion

- **DONE** Heap
  La característica que permite que un heap se pueda almacenar sin punteros es que, si se utiliza la numeración por niveles indicada, entoncés 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 = floor(k/2)
  3. □ Hijos del nodo j = {2*j-1, 2*j}
  4. □ Padre del nodo k = floor(k/2)+1
  5. □ ningunos

- **DONE** 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

- **DONE** 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

1.4 1.2 Minimo Maximo de un arreglo

1.4.1 PREREQUISITOS

- Cotas Inferiores en apuntes de CC3001 **:READING:**
  - http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Ordenacion/#1

1.4.2 Minimo (resp. maximo) de un arreglo

- Cota superior
- Cota inferior

1.4.3 Calcular el Minimo y el Maximo de un Arreglo en una sola computacion

- Cota superior?
  - cota superior para max + cota superior para min
- cota inferior?
  - cota inferior para min + cota superior para max?
  - min(cota inferior para min , cota superior para max?)

1.4.4 Cota superior para el problema de minmax

- Calcular el minimo con el algoritmo previo, y el maximo con un algoritmo simetrico, da una complejidad de $2n - 2$ comparaciones, que es demasiado.
- El algoritmo siguente calcula el max y el min en $\frac{3n}{2} - 2$ comparaciones:
  1. Dividir A en $\lfloor n/2 \rfloor$ pares (y eventualemente un elemento mas, x).
  2. Comparar los dos elementos de cada par.
  3. Ponga los elementos superiores en el grupo S, y los elementos inferiores en el grupo I.
  4. Calcula el minima m del grupo I con el algorimo de la pregunta previa, que performa $\lfloor n/2 \rfloor - 1$ comparaciones
  5. Calcula el maxima M del grupo I con un algoritmo simetrico, con la misma complejidad.
  6. Si n es par,
     - m y M son respectivamente el minimo y el maximo de A.

7. Sino, si $x < m$,
   - x y M son respectivamente el minimo y el maximo de A.
8. Sino, si $x > M$,
   - m y x son respectivamente el minimo y el maximo de A.
9. Sino
   - m y M son respectivamente el minimo y el maximo de A.

- La complejidad total del algoritmo es
  - $n/2 + 2(n/2 - 1) = 3n/2 - 2 \in 3n/2 + O(1)$ si n es par
  - $(n-1)/2 + 2(n-1)/2 + 2 = 3n/2 + 1/2 \in 3n/2 + O(1)$ si n es impar.
  - en la clase $3n/2 + O(1)$ en ambos casos.

1.4.5 Cota inferior para el problema de minmax

- Sean las variables siguentes:
  - O los o elementos todavia no comparados;
  - G los g elementos que "ganaron" todas sus comparaciones hasta ahora;
  - P los p elementos que "perdieron" todas sus comparaciones hasta ahora;
  - E las e valores eliminadas (que perdieron al menos una comparacion, y ganaron al menos una comparacion);

- (o, g, p, e) describe el estado de cualquier algoritmo:
  - siempre $o + g + p + e = n$;
  - al inicio, $g = p = e = 0$ y $o = n$;
  - al final, $o = 0$, $g = p = 1$, y $e = n - 2$.

- Despues una comparacion $a?b$ en cualquier algoritmo del modelo de comparacion, (o, g, p, e) cambia en funcion del resultado de la comparacion de la manera siguente:

	$a \in O$	$a \in G$	$a \in P$	$a \in E$
$b \in O$	$o-2, g+1, p+1, e$	$o-1, p, e+1$ $o-1, g, p+1, e$	$o-1, g, p, e+1$ $o-1, g+1, p, e$	$o-1, g+1, p, e$ $o-1, g, p+1, e$
$b \in G$		$o, g-1, p, e+1$	o, g, p, e $o, g-1, p-1, e+2$	o, g, p, e $o, g-1, p, e+1$
$b \in P$			$o, g, p-1, e+1$	o, g, p, e $o, g, p-1, e+1$
$b \in E$				o, g, p, e

- En algunas configuraciones, el cambio del vector estado depende del resultado de la comparacion: un adversario puede maximizar la complejidad del algoritmo eligando el resultado de cada comparacion. El arreglo siguente contiene en graso las opciones que maximizan la complejidad del algoritmo:

	$a \in O$	$a \in G$	$a \in P$	$a \in E$
$b \in O$	$o-2, g+1, p+1, e$	$o-1, p, e+1$ $\mathbf{o-1, g, p+1, e}$	$o-1, g, p, e+1$ $\mathbf{o-1, g+1, p, e}$	$o-1, g+1, p, e$ $o-1, g, p+1, e$
$b \in G$		$o, g-1, p, e+1$	$\mathbf{o, g, p, e}$ $o, g-1, p-1, e+2$	$\mathbf{o, g, p, e}$ $o, g-1, p, e+1$
$b \in P$			$o, g, p-1, e+1$	$\mathbf{o, g, p, e}$ $o, g, p-1, e+1$
$b \in E$				o, g, p, e

- Con estas opciones, hay
  - $\lceil n/2 \rceil$ transiciones de O a $G \cup P$, y
  - $n-2$ transiciones de $G \cup P$ a E.
- Eso resulta en una complejidad en el peor caso de $\lceil 3n/2 \rceil - 2 \in 3n/2 + O(1)$ comparaciones.

1.4.6 CONCEPT QUESTIONS [6/8] :CQ:

- **DONE** 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
- **NEXT** 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.
- **DONE** 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

- **DONE** Cota Inferior para Max
  Dado un arreglo de n enteros, cuanto comparaciones entre los elementos del arreglo se necesitan para calcular su valor maximal?

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

- **DONE** 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

- **DONE** 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. ☐ $\min\{f_A(n)$, $\mathrm{f}_B(\mathrm{n})\}\$$
  2. ☐ $f_A(n)$ + $\mathrm{f}_B(\mathrm{n})\$$
  3. ☐ $(f_A(n)$ + $\mathrm{f}_B(\mathrm{n}))/2\$$
  4. ☐ $\max\{f_A(n)$, $\mathrm{f}_B(\mathrm{n})\}\$$
  5. ☐ otra respuesta

- **DONE** 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

- **NEXT** 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

1.5 1.3 Tecnicas para demostrar cotas inferiores:

adversario, teoria de la informacion, reduccion

1.5.1 PREREQUISITOS

- de las apuntes de CC3001:
  - Ordenacion / Cotas inferiores http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Ordenacion/#1
  - Ordenacion / Merge sort http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Ordenacion/#5
- de las apuntes de CC4102:
  - min max

1.5.2 Busqueda Ordenada (en el modelo de comparaciones)

1. Cota superior: $2\lg n$ vs $1+\lg n$
2. Cota inferior en el peor caso: Strategia de Adversario cota inferior en el peor caso de $1+\lg n$
3. Cota inferior en el caso promedio uniforme
   - Teoria de la Informacion
   - = Arbol de Decision
   - cota inferior de $\lg(2n+1)$, i.e. de $1+\lceil \lg(n+1/2) \rceil$
4. La complejidad del problema
   - en el peor caso es $\Theta(\lg n)$
   - en el caso promedio es $\Theta(\lg n)$
5. Pregunta: en este problema las cotas inferiores en el peor caso y en el mejor caso son del mismo orden. Siempre es verdad?

1.5.3 Busqueda desordenada

1. Complejidad en el peor caso es $\Theta(n)$
2. Complejidad en el caso promedio?
    - cota superior
        - Move To Front
        - ?BONUS? Transpose
    - cota inferior
        - algoritmo offline, lemma del ave
    - A VER EN CASA O TUTORIAL: Huffman?

1.5.4 Ordenamiento (en el modelo de comparaciones)

- cota superior $O(n \lg n)$
- cota inferior en el peor caso
    - cual tecnica?
        * lema del ave?
        * Strategia de Adversario?
        * Arbol Binario de Decision
    - Resultado:
        * $\Omega(n \lg n)$
- cota inferior en el caso promedio
    - $\Omega(n \lg n)$

1.5.5 Lista de tecnicas para mostrar cotas inferiores

1. lema del ave
2. Strategia de Adversario
3. Arbol Binario de Decision
4. Lemma del Minimax (para complejidad en promedio y complejidad de algoritmos aleatroizados)

1.5.6 BONUS: complejidad en promedio y aleatorizada

- La relacion entre
    - complejidad en promedio de un algoritmo deterministico
    - complejidad en el peor caso de un algoritmo aleatorizado (promedio sobre su aleatoria)

1.5.7 CONCEPT QUESTIONS [11/19] :CQ:

- **DONE** 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

- **DONE** 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

- **DONE** 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

- **DONE** 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.

- **DONE** 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

- **DONE** 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? ($A[1] = -\infty$ 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

- **DONE** 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.

- **DONE** 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

- **NEXT** 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

- **TODO** 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

- **NEXT** Relacion entre codificacion y busqueda
  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

- **DONE** Busqueda Doblada
  Dado una valor x y un arreglo ordenado A de n valores, existe un algoritmo calculando la posicion de insercion 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. ☒ $\lg^* p + \sum_{i=1}^{\lg^* p} \lg^{(i)}(p) + \lg p$ comparaciones
  6. ☐ ningunas o mas que dos.

- **DONE** 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. □ $\lg^* p + \sum_{i=1}^{\lg^* p} \lg^{(i)}(p) + \lg p$ bits,
6. □ ningunas o mas que dos.

- **NEXT** 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

- **NEXT** 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.

- **NEXT** 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

- **NEXT** Complejidad aleatorizada

  1. □

- **DONE** 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.

- **NEXT** 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

1.6 1.4 Metodologia de experimentacion

1.6.1 PREREQUISITOS

- de las apuntes de CC3001:
  - Nociones básicas de programación http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Programacion/ (sin estudiarlas)
- de las apuntes de CC4102:
  - 1.4 Metodologia de experimentacion
- en la red (y en ingles)
  - Research Design: Qualitative, Quantitative, and Mixed Methods Approaches John W. Creswell http://www.amazon.com/Research-Design-Qualitative-Quantitative-Approaches/

1.6.2 Al inicio, Ciencias de la computacion fue solamente experimentacion.

- Turing y el codigo Enigma
  - el importante estaba de solucionar la instancia del dia (romper la llave del dia, basado en los messages de meteo, para decryptar los messages mas importantes)
    * no mucho focus en el problema, aunque Turing si escribio la definicion de la Maquina universal.
- Experimentacion basica

  - correga hasta que fonciona (o parece foncionar)
    * correga hasta que entrega resultados correctos (o que parecen correctos)
    * mejora hasta que fonciona en tiempo razonable (en las instancias que tenemos)

- Problemas:
  - Demasiado “Ah Hoc”
    * falta de rigor, de reproducibilidad
    * desde el inicio, no “test bed” estandard, cada uno tiene sus tests.
    * mas tarde, no estandard de maquinas

- Respuestas: Knuth et al.
  - complejidad asymptotica: independancia de la maquina
  - complejidad en el peor caso y promedio: independancia del “test bed”
  - todavia es necesario de completar las estudias teoricas con programacion y experimentacion: el modelo teorico es solamente una simplificacion.

- Theoreticos desarollaron un lado "mathematico" de ciencias
  de la computacion, con resultados importantes tal que
  - NP-hardness
  - “Polynomial Hierarchy” (http://en.wikipedia.org/wiki/Polynomial_hierarchy)

- Theoria y Practica se completen, pero hay conflictos en ambos lados:
  - demasiado theorias sin implementaciones (resultado del ambiante social tambien).
  - todavia hay estudios experimentales “no reproducibles”

1.6.3 Sobre la "buena" manera de experimentar

(“A Theoretician’s Guide to the Experimental Analysis of Algorithms”, David S. Johnson, 2001)

- Fija una hipothesis **antes** de programar.
  - aunque el objetivo sea de programar un software completo, solamente es necesario de implementar de manera eficiente la partes relevantes. El resto se puede implementar de manera “brutal”. (E.g. “Intersection Problem”)

- "Incremental Programming"
  - busca en la red “Agile Programming”, “Software Engineering”.
    * una experimentacion es tambien un proyecto de software, y las tecnicas de ingeniera de software se aplican tambien.
    * Construe un simulador en etapas, donde a cada etapa fonctiona el simulador entero.

- "Modular Programming"
  - Experimentacion es Investigacion, nunca se sabe por seguro que se va a medir despues.
    * Hay que programar de manera modular por salgar tiempo en el futuro.

1.6.4 Sobre la "buena" manera de presentar sus resultados experimentales.

("Presenting Data from Experiments in Algorithmics", Peter Sanders, 2002)

- El proceso:
  - Experimentacion tiene un ciclo:
    * "Experimental Design" (inclue la eleccion de la hypothesis)
    * "Description of Measurement"
    * "Interpretation"
    * vuelve al paso 1.
    * La presentacion sigue la misma estructura, pero solamente exceptionalemente describe mas que una iteracion (la mejor, no necesaramiente la ultima) del ciclo.
- Eliges que quieres comunicar.
  - el mismo dato se presenta diferamente en funcion de la emfasis del reporte.
    * pero, siempre la descripcion debe permitir la **reproducibilidad** de la experimentacion.
- Tables vs 2d vs 3d plot
  - tables
    * son faciles, y buenas para menos de 20 valores
    * son sobre-usadas
    * Grafes 3d
      · mas modernos, impresionantes, pero
      · en impresion no son tan informativos
      · tiene un futuro con interactive media donde el usuario puede cambiar el punto de vista, leer las valores precisas, activar o no las surfacas.
    * Grafes 2d
      · en general preferables, pero de manera inteligente!
      · cosas a considerar:
      · log scale en x y/o y
      · rango de valores en x y/o y.
      · regla de "banking to 45 deg":
      · "The weighted average of the slants of the line segments in the figure should be about 45"
      · se puede aproximar on un grafo en "paysage" siguiendo el ratio de oro.
      · factor out la informacion ya conocida
    * Maximiza el Data-Ink ratio.
      "Toward an Experimental method for algorithm simulation," INFORMS Journal on Computing, Vol 8, No 1 Winter 1995.
      "Analyzing Algorithms by simulation: Variance Reduction Techniques and Simulation Speedups," ACM Computing Surveys, June 1992.

- For a good book on introductory statistics with computer science examples, I recommend
- Cohen: Empirical Methods for Artificial Intelligence
- Thomas Bartz-Beielstein et al, on Empirical Methods for the Analysis of OPtimization Algorithms
- Catherine McGeoch, A Guide to Experimental Algorithmics, (January 2011)

1.6.5 Sobre la "buena" manera de describir una investigacion en general:

http://www.amazon.com/Making-Sense-Students-Engineering-Technical/dp/019542591X Making sense; a student's guide to research and writing; engineering and the technical sciences, 2d ed. Northey, Margot and Judi Jewinski. Oxford U. Press 2007 252 pages $32.50 Paperback

1.6.6 Otras referencias:

- Research Design: Qualitative, Quantitative, and Mixed Methods Approaches John W. Creswell http://www.amazon.com/Research-Design-Qualitative-Quantitative-Approaches/

1.7 1.5 Recurrencias y Introduccion a la programacion dinamica

1.7.1 PREREQUISITOS

- de las apuntes de CC3001:
  - Torre de Hanoi y recursion en http://www.dcc.uchile.cl/~bebustos/apuntes/cc3001/Programacion/#6
- de las apuntes de CC4102:
  - 1.5 Recurrencias y Introduccion a la programacion dinamica
- en la red (y en ingles)
  - http://en.wikipedia.org/wiki/Master_theorem
  - http://www.csanimated.com/animation.php?t=Master_theorem

1.7.2 Recurencias Lineales

- $X_n = X_{n-1} + a_n$
- Torre de Hanoi

1.7.3 Recurrencias mas complejas

- Fibonacci

1.7.4 Formulas practicas

- Subsecuencia de suma maximal

1.7.5 Programacion Dinamica

- Subsecuencia commun mas larga
  - solucion ingenua
  - Solucion en tiempo polynomial (pero espacio $O(n^2)$)
  - Solucion en espacio lineal (y tiempo $O(n^2)$)
  - (BONUS) Solucion de Hirshberg en tiempo $O(nm)$ y espacio $\min(n, m)$

1.8 RESUMEN de la Unidad

1. Conceptos Basicos
   - $O(), o(), \Omega(), \omega(), \Theta(), \theta()$
   - Complejidad en el peor caso, en promedio
   - Modelos computacionales:
     - modelo de comparaciones
     - modelo de memoria externa
2. Tecnicas de Cotas Inferiores
   - lema del ave (reduccion)
   - strategia de adversario
   - teoria de la informacion (arbol de decision binario)
   - Analisis fine
3. Metodologia de experimentacion
   - Porque?
   - Como hacer la experimentacion
   - Como analizar y presentar los resultados
4. Casos de Estudios

- Torre de Hanoi
- "Disk Pile problem"
- Busqueda y Codificacion de Enteros (busqueda doblada)
- Busqueda binaria en $\Theta(1 + \lg n)$ (mejor que $2 \lg n$)
- Algoritmo en $2n/3 + O(1)$ comparaciones para min max

[1] FOOTNOTE DEFINITION NOT FOUND: 0