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

Jérémy Barbay

18 November 2010

Contents

1 Resultados de Aprendisajes de la Unidad

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

2 Principales casos de estudio

- cota inferior para mínimo y máximo de un arreglo
- caso promedio del quicksort
- cota inferior para búsqueda en un arreglo con distintas probabilidades de acceso

3 Repaso del proceso de diseño y análisis de un algoritmo.

- Problema de la Torre de Hanoi
  - definition
  - cota superior
  - cota inferior
  - A VER EN CASA
  - Variantes de la Torre de Hanoi
    - 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
    - A VER EN CASA
    - Minimo (resp. maximo) de un arreglo
      - cota superior
      - cota inferior
      - Minimo Maximo de un arreglo
      - cota superior
      - cota inferior

4 Minimo Maximo: Cota inferior

- Sean las variables siguentes:
  - O los o elementos todavia no comparados;
  - G los g elementos que "gan" todas su comparaciones hasta ahora;
  - P los p elementos que "perdieron" todas su comparaciones hasta ahora;
  - E las e valores eliminadas (que perdieron al menos una comparaci gan 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 maximisar 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 case de $\lceil 3n/2 \rceil - 2 \in 3n/2 + O(1)$ comparaciones.

5 Minimo Maximo: Cota superior

- 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:
  - Dividir A en $\lfloor n/2 \rfloor$ pares (y eventualemente un elemento mas, x).
  - Comparar los dos elementos de cada par.
  - Ponga los elementos superiores en el grupo S, y los elementos inferiores en el grupo I.
  - Calcula el minima m del grupo I con el algorimo de la pregunta previa, que performa $\lfloor n/2 \rfloor - 1$ comparaciones
  - Calcula el maxima M del grupo I con un algoritmo simetrico, con la misma complejidad.
  - Si n es par,
    - m y M son respectivamente el minimo y el maximo de A.
  - Sino, si $x < m$,
    - x y M son respectivamente el minimo y el maximo de A.
  - Sino, si $x > M$,
    - m y x son respectivamente el minimo y el maximo de A.
  - 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
  - en la clase $3n/2 + O(1)$ en ambos casos.

6 Técnicas para demostrar cotas inferiores: adversario, teoria de la informacíon, reduccion

1. Busqueda Ordenada (en el modelo de comparaciones)

   (a) Cota superior: $2 \lg n$ vs $1 + \lg n$

   (b) Cota inferior en el peor caso: Strategia de Adversario cota inferior en el peor caso de $1 + \lg n$

   (c) Cota inferior en el caso promedio uniforme

   - Teoria de la Informacion
   - = Arbol de Decision
   - cota inferior de lg(2n+1), i.e. de $1 + \lg(n + 1/2)$ *rsup*

   (d) La complejidad del problema

   - en el peor caso es $\Theta(\lg n)$
   - en el caso promedio es $\Theta(\lg n)$

   (e) Pregunta: en este problema las cotas inferiores en el peor caso y en el mejor caso son del mismo orden. Siempre es verdad?

2. Busqueda desordenada

   (a) Complejidad en el peor caso es $\Theta(n)$

   (b) 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?

3. 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
  - $\Omega(n \lg n)$

4. 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)

7 Metodologia de experimentación

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

7.1.1 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.

7.1.2 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)

7.1.3 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

7.1.4 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.

7.1.5 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$_h ierarchy$)

7.1.6 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”

7.2 Sobre la "buena" manera de experimentar

("A Theoretician's Guide to the Experimental Analysis of Algorithms", David S. Johnson, 2001)

7.2.1 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")

7.2.2 "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.

7.2.3 "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.

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

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

7.3.1 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

7.3.2 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.

7.3.3 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.

1. 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)

7.4 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

7.5 Otras referencias:

- Research Design: Qualitative, Quantitative, and Mixed Methods Approaches John W. Creswell http://www.amazon.com/Research-Design-Qualitative-Quantitative-Approaches/dp/1412965578/ref=ntt$_a t_e p_d p i_1$

8 Resumen de la Section

8.1 Controles y Tareas

- TAREA 1: Busqueda Binaria vs others
- TAREA 2: static cache-oblivious tree
- TAREA 3: Interpolation vs others
- CONTROL 1
- TAREA 4: Hashing

- TAREA 5: Paginamiento
- TAREA 6: Vertex Cover
- CONTROL 2
- EXAMEN

Las tareas se hacen en dos semanas.

8.2 Recurrencias y Introduccion a la programacion dinamica

- $X_n = X_{n-1} + a_n$
- Torre de Hanoi
- Fibonacci
- Subsecuencia de suma maximal
- 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 $\min(n, m)$

8.3 Resumen de la Section

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