cc4102

Jeremy Barbay

17 October 2011

Contents

1 DONE CC4102: (B203) Lower Bounds

`SCHEDULED:` *2011-10-17 Mon 11:30-12:45*

1. Complejidad Computacional de un algoritmo y de un problema:
   (a) Tres nociones distintas:
      - Cota superior: "en cuanto tiempo se puede hacer"?
      - Cota inferior: "cuanto tiempo se necesita al minimo"?
      - `Complejidad Computacional`: cuanto tiempo usa la mejor solucion?

   (b) Puntos importantes
      i. Diferencia entre cota superior y inferior
      ii. Definir el modelo (e.g. modelo de comparacion)
      iii. Diferencia entre Peor Caso y otros (e.g. Mejor caso, caso promedio, caso de una instancia)

2. Ejemplo: Torre de Hanoi y Platos Sucios

	Torre de Hanoi con n discos	Platos Sucios con n platos de s tamaños	Platos Sucios con n_i platos de tamaño i, $i \in [1..s]$
Cota Superior	$1+2+4+..=2^{n+1}-1$	Menos que $2^{n+1}-1$?	
Cota Inferior	$1+2+4+..=2^{n+1}-1$	Mas que $2^{s+1}-1$?	
Complejidad	$2^{n+1}-1$	$\Theta((n-s-1)2^{s-1})$	$\sum_{i=1}^{s} n_i 2^{s-i}$

- Torre de Hanoi
  - cf Años previos, Wikipedia, etc...
- Platos Sucios
  - Cota Superior
    * Mueve todos platos de mismo tamaño en tiempo lineal
    * por el resto, como por torre de hanoi
  - Cota Inferior *en el peor caso con n_i platos de tamaño i, $i \in [1..s]$
    * todas las instancias con n_i platos de tamaño i, $i \in [1..s]$ son de misma dificultad.
    * similar a la torre de hanoi sobre s discos.
    * $\sum_{i=1}^{s} n_i 2^{s-i}$
  - Cota Inferior *en el peor caso con n platos y s tamaños
    * el peor caso es cuando hay $n_1 = n - s + 1$ dicos pequeños.
    * la formula general da
      · $(n - s + 1)2^{s-1}$ para los n_1 discos de tamaño 1, y
      · $\sum_{i=2}^{s} 2^{s-i} = 2^0 + \ldots + 2^{s-2} = 2^{s-1} - 1$ para los otros discos
      · que suma a $(n - s)2^{s-1} + 2^s - 1$
      · (uno puede verificar que vale $2^n - 1$ por $s = n$)

3. Tecnicas

   (a) Strategia de Aversario
   - "batalla naval"

   (b) Reduccion
   - Lemma del Ave
   - $3SUM$

   (c) Information Theory
   - Arboles de Decisiones

2 TODO CC4102: (B203) Lower Bounds

`SCHEDULED:` *2011-10-20 Thu 11:30-12:45*

1. Lista de Tecnicas para mostrar Cotas Inferiores

   (a) Estrategia de Aversario
   - "batalla naval"
   - Max
   - minMax
   - busqueda desordenada
   - busqueda ordenada

(b) Reduccion
    - Lemma del Ave
    - $3SUM$
(c) Information Theory
    - Arboles de Decisiones
    - Huffman

2. Estrategia de Adversario
    (a) Busqueda Desordenada
    (b) Maxima
        - for loop
    (c) minMax
        i. 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:
            A. Dividir A en $\lfloor n/2 \rfloor$ pares (y eventualemente un elemento mas, x).
            B. Comparar los dos elementos de cada par.
            C. Ponga los elementos superiores en el grupo S, y los elementos inferiores en el grupo I.
            D. Calcula el minima m del grupo I con el algorimo de la pregunta previa, que performa $\lfloor n/2 \rfloor - 1$ comparaciones
            E. Calcula el maxima M del grupo I con un algoritmo simetrico, con la misma complejidad.
            F. Si n es par,
                - m y M son respectivamente el minimo y el maximo de A.
            G. Sino, si $x < m$,
                - x y M son respectivamente el minimo y el maximo de A.
            H. Sino, si $x > M$,
                - m y x son respectivamente el minimo y el maximo de A.
            I. 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.

ii. Cota Inferior

- 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.

(d) Maxima y secundo maxima
  - Cota inferior de $n - 1 + log_2 n - 1$ con campionato
  - TAREA: cota inferior de $3/2n - 2$
    - piensan en las limitas de este modelo

3. REDUCCION:
   - Prerequisitos:
     - Vearon reducciones en el curso de calculabilidad (Alejandro Hevia o Gonzalo Navarro) para NP
     - Ya conocen la cota inferior de $\Omega(n \lg n)$ en el modelo de comparacion para ordenar
   - Cobertura Convexa (i.e. Convex Hull) $\in \Omega(n \lg n)$
     - Cota inferior de $\Omega(n \log n)$ por reduccion
   - Insertar y Extract-Min en Colas de Prioridades (i.e. Priority Queues) $\in \Omega(\lg n)$
   - 3SUM,
     - se puede mostrar cotas inferiores sin conocer la complejidad
     - Definicion del problema $3SUM$
     - Encontrar 3 puntos colineares a dentro de n puntos esta **3SUM-hard**:
       * Dado una instancia S de 3SUM
       * $\forall x \in S$, crea los puntos $(x, 1)$, $(-x/2, 2)$, and $(x, 3)$.
       * $\exists a, b, c \in S$ tq $a + b + c = 0$ iff hay 3 puntos colineares $(a, 1), (-b/2, 2), (c, 3)$.

4. Information theory:
   - Arboles de decision = arboles de codificacion
   - Cada algoritmo correcto tiene que **demostrar** que su respuesta es correcta.
   - El tamaño de esta respuesta constitue una cota inferior.
   - Si existe un conjunto S de instancia que necesitan una justificacion (i.e. **certificado**) de correccion distinta, entonces hay una cota inferior de $\Omega(\lg |S|)$ en el modelo de comparacion
   - Eso permite de dar cotas inferiores para
     (a) Busqueda Binaria

- en el peor caso,
- en promedio con distribuciones de probabilidades
  * Shannon lower bound $n \sum p_i \log 1/p_i$
  * So no algorithm can do better than this
  * Can one encode in this space?
    · Huffman: $H + 1$
  * Is it useful as a search tree?
    · No: order in the leaves
  * Hu-Tucker:
    · $H+2 =>$ optimal algorithm in O-terms, $O(H+1)$ time.
  * Detour: dynamic programming algorithm for building the optimal tree in $O(n^2)$ time.

(b) Ordenamiento

- en el peor caso,
- En promedio
  * Cada permutacion requiere acciones distintas para ser ordenada
  * Un algoritmo con las mismas acciones sobre dos permutaciones distintas esta incorrecto.