Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 8 - "Algoritmos Online"

Profesor: Pablo Barceló
Auxiliar: ~~Manuel~~ Ariel Cáceres Reyes

2 de Junio del 2017

P1. Tareas y Procesadores

Tenemos m procesadores idénticos. Como entrada recibimos una lista $t_1, t_2, \ldots, t_n$ de tareas con tiempos de procesamiento $p_1, p_2, \ldots, p_n > 0$, respectivamente. Las tareas son recibidas secuencialmente, y sólo cuando una tarea llega conocemos su tiempo de procesamiento. Cada tarea debe ser asignada a una máquina inmediatamente, y la decisión no puede ser cambiada. La *carga* de un procesador es la suma de los tiempos de procesamiento de todas las tareas que le son asignadas. El costo de un algoritmo que resuelve el problema de asignación es la máxima carga entre sus procesadores.
Considere el siguiente algoritmo para el problema anterior. Cada tarea es asignada al procesador con menor carga (en caso de empate, se elige cualquiera).

a) Demuestre que este algoritmo es $\left(2 - \frac{1}{m}\right)$ −competitivo.

b) Demuestre que esta cota es óptima para el algoritmo.

P2. Online Set Covering

En el problema online de set covering, se nos entrega (U, S), donde U contiene n elementos y S contiene m subconjuntos de U. Luego se entrega de forma online una secuencia de elementos $e_1, e_2, \ldots, e_t$. El algoritmo que resuelve el problema debe construir un subconjunto R de S. Cuando llega un nuevo elemento, si no está cubierto por los subconjuntos de R, debe escogerse un conjunto de S que lo contenga y agregarlo a R (se permite agregar más conjuntos, si se desea).

a) Demuestre que no se puede obtener (usando un algoritmo determinístico) un radio competitivo mejor que $\log_2 n$.

b) Demuestre que no se puede obtener un radio competitivo mejor que $\Omega\left(\frac{\log m}{\log n}\right)$ con un algoritmo determinístico.

Soluciones

P1. a) Enumeremos las tareas en orden creciente de carga (luego de correr ALG), si lo hacemos, el costo de ALG queda expresado como la carga del último de estos procesadores, el m. Consideremos ahora la asignación de la última tarea del procesador m. Por como funciona el algoritmo, cuando se le fue asignada esta tarea, el procesador m tenía la menor carga entre los procesadores. Por lo tanto, el período de ocio de cada uno de los $m-1$ procesadores anteriores no puede ser mayor que la duración de esta última tarea (sino este procesador no hubiese sido escogido como el de carga menor). De este modo, estos períodos de ocio no pueden exceder la duración de la tarea más larga, $max\ p_i$. Con esto se tiene que:

$$\begin{aligned} m \cdot ALG &\leq \sum_{i=1}^{n} p_i + (m-1) max\ p_i \\ &\leq \frac{1}{m} \sum_{i=1}^{n} p_i + \left(1 - \frac{1}{m}\right) max\ p_i \end{aligned}$$

Finalmente, el óptimo no puede ser menor a la carga promedio (pues este es el óptimo del problema relajado), ni a la máxima duración entre las tareas (pues algún procesador debe procesarle).

$$\begin{aligned} &\leq OPT + \left(1 - \frac{1}{m}\right) OPT \\ &= \left(2 - \frac{1}{m}\right) OPT \end{aligned}$$

b) Que la cota sea óptima quiere decir que el ALG no consigue un radio competitivo menor a $\left(2 - \frac{1}{m}\right)$. Esto lo podemos hacer mostrando que hay un input para el cual se ALG alcanza exactamente $\left(2 - \frac{1}{m}\right) OPT$.

Consideremos $m(m-1)$ tareas de tiempo 1 y luego una tarea de tiempo m. En este caso ALG distribuirá equitativamente las primeras tareas dando carga $(m-1)$ a cada procesador y luego la última tarea se la dará a alguno de los procesadores quien quedará con carga $2m-1$ obteniendo este costo. Sin embargo, el OPT distribuye de manera equitativa estas tareas dándole carga m a cada uno de los procesadores.

De esta forma $ALG = 2m - 1 = \left(2 - \frac{1}{m}\right) m = \left(2 - \frac{1}{m}\right) OPT$.

P2. El objetivo de este problema es demostrar que no se pueden obtener radios competitivos mejores a cierta cota, para mostrar esto veremos que dado un algoritmo, existe un input para el cual el algoritmo alcanza exactamente esa cota (veces el OPT). Por esta razón supondremos que nos damos ALG un algoritmo cualquiera que resuelve Online Set Covering.

a) Consideremos como input $U = \{0,1\}^d, n = 2^d, S = \{S_i\}^d$, con S_i ={cadenas que tienen un 1 en la posición i}. Y la secuencia de peticiones la siguiente:

- $e_1 : 1 \dots 1$, a lo que ALG lo cubre con S_{i_1} [1]
- $e_2 : 1 \dots 1 \underbrace{0}_{i_1} 1 \dots 1$, que ALG lo cubre con S_{i_2}.
- $e_3 : 1 \dots 1 \underbrace{0}_{min(i_1,i_2)} 1 \dots 1 \underbrace{0}_{max(i_1,i_2)} 1 \dots 1$, que ALG lo cubre con S_{i_3}.
- $\dots$
- $e_d : 0 \dots 0 \underbrace{1}_{i_d} 0 \dots 0$, que el algoritmo cubre con S_{i_d}.

Notar que el costo de ALG en este caso es d, mientras que el OPT consiste en cubrir todos los e_i con S_{i_d}. Finalmente, como $d = \log n$, se tiene que $ALG = \log n = \log n \cdot 1 = \log n \cdot OPT$

b) Escojamos ahora $U = [n] = \{1, \dots, n\}$ y $S = \{Q \subseteq U : |Q| = \sqrt{n}\}$ (subconjuntos de U de tamaño $\sqrt{n}$), por lo tanto $|S| = \binom{n}{\sqrt{n}} = m$.
Y la secuencia de peticiones siguiente:

- e_1 un elemento cualquiera en U, a lo que ALG cubre con Q_1.
- $e_2 \notin A_1$, a lo que ALG cubre con Q_2.
- $e_3 \notin A_1 \cup A_2$, a lo que ALG cubre con Q_3.
- $\dots$
- $U = A_1 \cup \dots \cup A_d$. Por lo que no se puede continuar con el proceso.

Notemos que $d \geq \sqrt{n}$, pues en el mejor caso ALG escoge los conjuntos disjuntos y como estos son de tamaño $\sqrt{n}$, necesita exactamente $\sqrt{n}$ de estos para cubrir U. Pero estos $\sqrt{n}$ elementos pueden haber sido cubiertos con solo un conjunto de S, por lo que ALG es peor por un factor de $\sqrt{n}$. Finalmente, se puede mostrar matemáticamente que $\sqrt{n}$ es $\Omega\left(\frac{\log m}{\log n}\right)$.

[1] Puede que el algoritmo decida cubrir con más de un elemento, pero es simple notar que esto solo afecta el número de peticiones, pero no el costo (que corresponde al número de conjuntos en este caso).