CC4102 - Diseño y Análisis de Algoritmos
Auxiliar 8

Prof. Gonzalo Navarro; Aux. Mauricio Quezada

23 de octubre de 2012

1 El problema de los k servidores

Considere el escenario donde tiene k puntos (*servidores*) en un *espacio métrico* (donde está definida una función de distancia d: simétrica, no-negativa y que cumple la desigualdad triangular) y una secuencia de puntos (*peticiones*) que debe atender. Cada vez que llega una petición, un servidor debe moverse hacia esa posición.

El problema *online* consiste en minimizar la distancia recorrida por todos los servidores luego de n peticiones, sin saber la secuencia de puntos a atender.

Recuerde que un algoritmo *online* ALG es r-competitivo si existe una constante a tal que para cualquier instancia I y el algoritmo óptimo OPT,

$$\text{cost}_{ALG}(I) \leq r \cdot \text{cost}_{OPT}(I) + a$$

Donde r es el *radio competitivo*.

1. Sea ALG un algoritmo online para el problema de los k servidores bajo un espacio métrico arbitrario con al menos $k+1$ puntos. Pruebe que el radio competitivo de ALG es al menos k.

2. Considere el problema en un *espacio métrico uniforme*, es decir, donde la distancia entre cada par de puntos distintos es 1. El algoritmo LRU (*least recently used*) es de la siguiente forma: si la petición ya está cubierta no hace nada, si no, moverá el servidor menos usado hacia la petición. Pruebe que LRU es k-competitivo.

2 Offline Least Common Ancestor

Considere el problema del *Least Commmon Ancestor* o LCA, en el cual dado un árbol T y dos nodos u y v, $lca(u, v)$ es el nodo ancestro de u y de v de mayor profundidad en T. El problema *offline* considera que todas las consultas de LCA son dadas inicialmente como un conjunto de pares de nodos P.

Diseñe un algoritmo eficiente para el problema offline, donde haciendo un procesamiento del árbol se retornen los $lca(u, v)$ para cada $(u, v) \in P$.

3 Propuestos

3.1 k servidores en una línea

Considere el problema de los k servidores. Para el siguiente análisis competitivo, necesitamos usar una conocida herramienta, la función potencial. Una función Φ es una función de potencial que demuestra un radio competitivo r de un algoritmo ALG si satisface las siguientes condiciones:

1. Φ es nonegativa
2. Cada respuesta a una petición a OPT incrementa Φ no más de r veces el costo cargado a OPT por esa respuesta.
3. Cada respuesta a una petición a ALG disminuye el potencial por al menos el costo cargado a ALG por esa respuesta.

Por lo que, un algoritmo es r -competitivo, si existe una función de potencial Φ para $r > 0$ que cumple las propiedades anteriores.

Considere el problema de k servidores en una línea (un espacio de dimensión 1) y el siguiente algoritmo:

- Si todos los servidores están a un lado de la petición, entonces envía el servidor más cercano a ella.
- Si una petición se encuentra entre dos servidores, envía los dos servidores a velocidad constante, y se detienen cuando uno de ellos llega a su objetivo.

Muestre que la siguiente función potencial demuestra un radio competitivo de k para este algoritmo:

$$\Phi = k \sum_i |a_i - s_i| + \sum_{i<j} |s_i - s_j|$$

Donde $a_1, \ldots, a_k$ son los servidores del adversario y $s_1, \ldots, s_k$ los del algoritmo.

3.2 P2 Examen 2008

El profesor Locovich está empeñado en determinar cuál es el último piso de un edificio desde el que puede tirarse un huevo de avestruz sin romperse. (No es necesario insistir sobre la tremenda trascendencia del resultado para la Ciencia.) El edificio tiene n pisos, pero dado el alto costo de los huevos de avestruz, el Profesor sólo dispone de k huevos. Debe usted diseñar un algoritmo para resolver el problema. El costo de su algoritmo es la cantidad de huevos que necesita tirar para responder.

1. Demuestre que si $k = 1$, la complejidad del problema es n.
2. Encuentre un algoritmo $O(\sqrt{n})$ para $k = 2$.
3. Generalice para obtener $O(kn^{1/k})$.
4. Obtenga la mejor complejidad posible si no hay limitaciones en k. ¿Cuántos huevos necesita para obtener esta complejidad?

3.3 P4 Control 1 2006/1

Se tiene una secuencia de bits y se desea realizar la operación *rank* eficientemente, pero también queremos permitir la operación de invertir un bit de la secuencia. Diseñe una estructura de datos para realizar $rank(i)$ e $invertir(i)$ (que invierte el i-ésimo bit de la secuencia) en tiempo $O(\log n)$, utilizando $O(n)$ bits de espacio. Hint: comience con un árbol balanceado que contenga un nodo por bit, luego reduzca el espacio.

3.4 P3 Control 1 2009/1

Un árbol α-balanceado, para $1/2 < \alpha < 1$, es un árbol binario de búsqueda donde todo subárbol $T = (root, T_l, T_r)$ cumple $|T_l| \leq \alpha \cdot |T|$ y $|T_r| \leq \alpha \cdot |T|$. Las operaciones para buscar y mantener un árbol α-balanceado son las mismas que para un árbol binario de búsqueda, excepto que luego de insertar o borrar un nodo, se busca el nodo más alto en el camino del punto de inserción/borrado hacia la raíz, que no esté α-balanceado, y se lo reconstruye como árbol perfectamente balanceado (el costo es proporcional al tamaño del subárbol que se reconstruye).

1. Muestre que la búsqueda un árbol α-balanceado cuesta $O(\log n)$, y que lo mismo ocurre con las inserciones y borrados, si no consideramos las reconstrucciones. ¿Qué constante obtiene multiplicando el $\log n$?

2. Muestre que el costo amortizado de las inserciones y borrados, ahora considerando reconstrucciones, es también $O(\log n)$. Para ello, considere la función potencial

$$\Phi(T) = \frac{1}{2\alpha - 1} \sum_{T' \in T} abs(|T'_l| - |T'_r| - 1)$$

   donde $abs(\cdot)$ es el valor absoluto, y T' inT significa que T' es un subárbol de T. (Bonus) Encuentre la constante que multiplica a $\log n$, y recomiende un α óptimo.

3.5 P3 Control 1 2010/1

Un *árbol cartesiano* para un arreglo de enteros $A[1, n]$ se define de la siguiente manera; la raíz del árbol es el mínimo $A[\mu]$ del arreglo A (si hay varios se elige el más izquierdo). El hijo izquierdo es el árbol cartesiano de $A[1, \mu - 1]$ y el hijo derecho es el árbol cartesiano de $A[\mu + 1, n]$.

1. Dibuje el árbol cartesiano de $A[1, 10] = \langle 2, 9, 4, 6, 7, 5, 3, 1, 8, 10 \rangle$.

2. El ancestro común más bajo de dos nodos i y j, $lca(i, j)$, es el nodo de mayor profundidad que es ancestro de i y de j. Demuestre que el mínimo valor en cualquier rango $A[i, j]$ se halla en $A[lca(i, j)]$, donde lca se refiere al árbol cartesiano.

3. Muestre que el árbol cartesiano se puede construir en $O(n)$. Considere usar inducción, procesando los elementos de A de izquierda a derecha, y análisis amortizado para obtener la cota.