Mejoras al Rebalse Separado

Hugo Ryckeboer

Resumen:

La técnica de pseudoaleatorización de datos con rebalse separado une dos ideas: la de distribuir por pseudoazar los datos y una técnica de administración independiente para cada una de las listas resultantes. Sin embargo la técnica de lista vinculada que habitualmente se usa puede ser sustituida por otras más eficientes. Se analiza en detalle el uso de árboles y se tabula resultados numéricos para estos y para listas ordenadas secuenciales.

Palabras claves: *Estructuras de almacenamiento, Estructuras de datos, Pseudoaleatorización de datos ("hashing").*

Respetando un vocabulario antropomórfico se puede decir que las estructuras de datos tienen por función facilitar la evocación, principalmente asociativa, de la información. Si esta está representada por nuplas, llamaremos **X** al conjunto de asociantes e **Y** al conjunto de asociados. La incorporación de nueva información será una memorización y su eliminación un olvido (voluntario). Otros llaman a estos tres procesos consulta, alta y baja respectivamente.

En las referencias a la cardinalidad del conjunto almacenado se usará, como es habitual, la letra **N**.

Los métodos más utilizados en la evocación asociativa de nuplas, cuando la eficiencia se puede evaluar por esfuerzo medio, recurren a una división basada sobre pseudoazar.

Con ella el conjunto de asociantes **X**, queda particionado en múltiples subconjuntos. Las nuplas se ubican donde van sus partes **X**.

Para ubicar estos conjuntos en memoria se recurre ya sea a listas separadas, ya sea a un único espacio en el cual compiten por celdas libres. Como idealmente se hubiera deseado fraccionar **X** en conjuntos unitarios se habla de colisiones para indicar el hecho de que una lista tenga más de un elemento y acorde con ello las técnicas para instalar las varias listas se llaman políticas de resolución de colisiones [STA, 4.3].

No cambia demasiado el planteo si la memoria se divide primero en macroceldas (llamadas a veces baldes) que pueden contener varias nuplas (la ubicación de una se llama ranura).

Las políticas utilizadas se dividen usualmente en encadenada y abiertas. La primera maneja cada lista en forma de una lista vinculada, las segundas instalan todas las listas en un único espacio. En este trabajo se propone abordar las listas llamadas vinculadas o encadenadas. Esta política también se llama rebalse separado por cuanto mantiene los conjuntos en listas independientes.

En este trabajo se formulan dos observaciones referidas al uso de listas vinculadas. Para ello se replantea el problema.

El uso de una función de pseudoazar **h**: **X** -> **M** = {1,2,...,**M**} ha dividido el conjunto de asociantes en **M** subconjuntos $\mathbf{X}_i$. Todas las nuplas cuyo asociante pertenezca a un mismo $\mathbf{X}_i$ serán instaladas en una sola lista.

Si nos encontramos en una situación dinámica en la cual las memorizaciones se intercalan con las evocaciones resulta ineficiente pensar en instalar las listas con alojamiento secuencial. Si bien existen técnicas para ubicar y administrar varias listas de alojamiento secuencial en un único espacio [KNU 2.2.2], cuando hay muchas listas se vuelven poco prácticas. Las altas generan corrimientos y si una lista toca a su vecina arrastra a esta en sus movimientos. Cuando la ocupación se acerca al 100% del espacio disponible, la secuencia de listas que se tocan puede ser grande y se vuelve sumamente ineficiente la inserción de nuevos elementos.

Sin embargo si la situación es estática, o sea si la memorización se puede completar antes de comenzar las evocaciones, como sucede por ejemplo con las palabras claves de un lenguaje, nada impide organizar la situación con múltiples listas de alojamiento secuencial y si además se las ordena se puede buscar luego en ellas, si sus largos lo justifican, con búsquedas binarias.

Instalar las nuplas en listas de alojamiento secuencial se puede lograr con solo leerlas dos veces. La técnica propuesta no pretende ser original sino que puede considerarse una adaptación de una práctica ya establecida para construir representaciones ralas de matrices.

Supongamos que los elementos a estructurar estén en un archivo secuencial. Por otra parte supongamos que se los quiere ubicar en **M** listas. El siguiente pseudo-código ilustra la maniobra:

```
for i := 1 to M
do
     contador[i] := 0
od
open  archivo
while  not  fin(archivo)
do
     obtener x
     contador[ h(x) ] +:= 1
od
co   con esto ya se conoce la cardinalidad de cada conjunto  oc
origen[1] := 1
for  i := 1 to M
do
     origen[i+1] := origen[i] + contador[i]
od
co  ya se conoce el origen del almacenamiento de cada conjunto  oc
for  i := 1 to M
do
     proximo_a_llenar[i] := origen[i]
od
open  archivo
while  not  fin(archivo)
do
     obtener x, y
     celda[ próximo_a_llenar[ h(x) ] ] := (x,y)
     próximo_a_llenar[ h(x) ] +:= 1
od
co  cada conjunto ya está instalado (desordenadamente) en su lugar   oc
```

Cada conjunto ocupa las posiciones *origen*[i] a *origen*[i+1]-1, por eso que conviene tener **M**+1 orígenes para que el último tenga demarcación de fín.

Si se los quiere tener ordenados hay que arrancar **M** procesos de ordenamiento.

El espacio de trabajo se puede reducir a solo el vector de orígenes si se observa que el vector *próximo_a_llenar* tiene, cuando termina la carga de las nuplas, en su lugar i el origen de la lista i+1, con lo cual no tiene porque ser un vector independiente, basta con tener en cuenta que la carga corre los índices que delimitan a las listas en un lugar. Del mismo modo se puede instalar el vector de orígenes casi superpuesto con el de los contadores.

El siguiente programa unifica en un vector *P* los 3 vectores de trabajo.

```
for i := 1 to M
do
     P[i+2] := 0
od
```

```
open archivo
while not fin(archivo)
do
     obtener x
     P[ h(x)+2 ] +:= 1
od
co con esto ya se conoce la cardinalidad de cada conjunto en P[ 3..M+2 ] oc
P[2] := 1
for i := 1 to M
do
     P[i+2] := P[i+1] + P[i+2]
od
co ya se conoce el origen del almacenamiento de cada conjunto en P[ 2..M+1 ],
obsérvese que en la suma de este ciclo el primer P[i+2] es el nuevo origen que
se calcula y el P[i+2] de la derecha es un valor que todavía representa un
contador oc
open archivo
while not fin(archivo)
do
     obtener x, y
     celda[ P[ h(x)+1 ] ] := (x,y)
     P[ h(x)+1 ] +:= 1
od
P[1] := 1
co ahora P[1.. M] contiene los orígenes de las listas y p[M+1] la dirección de
finalización de la última. oc
```

Cuando la situación es dinámica, que es lo habitual, se recurre a listas vinculadas. Sin embargo, esta no es la única estructura definida por extensión, y que por lo tanto pueda acomodarse a espacios arbitrarios. Los árboles tienen la misma cualidad, pero menor esfuerzo de búsqueda. Debiera introducir por lo tanto un beneficio en tiempo organizar las listas en forma de árboles.

Inclusive, si en lugar de utilizar árboles comunes se los utilizara balanceados nos encontraríamos con una estructura de búsqueda con esfuerzo medio bajo (tan cercano a uno como se quiera bajando el factor de carga) y esfuerzo máximo de orden logarítmico en **N**. En su planteo habitual el rebalse separado tiene un esfuerzo medio bajo (aunque luego veremos que no tanto como la nueva estructura) pero un esfuerzo máximo apriori de orden lineal en **N**.

Análisis del esfuerzo para el rebalse manejado con árboles.

Los esfuerzos medios se pueden estudiar aposteriori, o sea después de haber memorizado, o apriori, o sea antes de comenzar la memorización.

Los métodos de cálculo del esfuerzo medio estudian primero los resultados aposteriori y luego promedian todas las formas

posibles en que pueden darse las estructuras aposteriori para conocer el valor apriori.

En los métodos bajo discusión: las listas encadenadas, los árboles o los alojamientos secuenciales ordenados, justamente por mantener las listas separadas el orden relativo en el cuál se suceden las memorizaciones destinadas a distintas listas no tiene incidencia. Cada una de las **M** listas puede ser estudiada por separada.

La forma más intuitiva de evaluar el esfuerzo medio consiste en tomar las **N** nuplas y evocar cada una de ellas una vez. Esto respeta una hipótesis, habitual en estos análisis, de igualdad de probabilidad de consulta de cada nupla. Se obtienen así **N** esfuerzos individuales de evocación que deben ser promediados a los efectos de estimar el esfuerzo medio.

Como el ordenamiento de los sumandos no altera la suma, se puede agruparlos por número de lista en la que han caído. Si las listas tienen n_1, n_2, ..., n_M elementos quedaría:

$$(e_1+e_2+\ldots+e_{n_1})+(e_{n_1+1}+e_{n_1+2}+\ldots+e_{n_1+n_2})+\ \ldots\ +(\ldots+e_N)$$

donde el primer paréntesis representa la contribución de los elementos de la primera lista, el segundo de la segunda, etc.

Por definición el primer paréntesis vale n_1 * $E(n_1)$ o sea el esfuerzo de n_1 búsquedas en una estructura que tiene n_1 elementos. Estrictamente este esfuerzo puede variar de una situación a otra, o sea dos conjuntos con la misma cantidad n pueden engendrar distinto esfuerzo medio de búsqueda según lo bien o mal que se hubieran acomodado sus elementos. Si se trata de listas vinculadas u ordenadas de alojamiento secuencial este esfuerzo no depende del orden de llegada de los elementos, pero si se trata de árboles el orden de llegada puede ser decisivo oscilando entre un óptimo de naturaleza logarítmica y un máximo lineal. El producto n * E(n) se denominará $E_T(n)$ y con un subíndice se resaltará el hecho de que n no es suficiente para establecer su valor aposteriori. Sin embargo cuando se promedien las situaciones también quedarán promediados los distintos órdenes de llegada y se podrá hablar de un valor $E_T(n)$ único.

De acuerdo a lo indicado el valor medio aposteriori será:

$$\frac{1}{N}\sum_{i=1}^{M} E_{T_i}(n_i)$$

y para pasar a esfuerzos apriori hay que hacer intervenir la distribución multinomial

$$\binom{N}{n_1\ n_2\ \cdots\ n_M}\ p_1^{n_1}\ p_2^{n_2}\ \cdots\ p_M^{n_M}$$

la que deberá ser sumada sobre todos los valores de n_1, n_2, ..., n_M que sean no negativos y sumen **N**. Permutando las sumatorias y observando que interviene un solo n_i por vez se puede transformar en binomial y si además se supone un problema simétrico (igualdad de probabilidades p_i o sea: $p_1 = p_2 = \ldots = p_M = 1/\mathbf{M}$) habrá **M** sumas similares lo que introduce el factor **M** en lugar de sumar **M** binomiales.

$$\frac{M}{N}\sum_{n=0}^{N} E_T(n)\binom{N}{n}\ p^n\ q^{N-n}$$

Son resultados conocidos [KNU, STA] que si se trata de listas vinculadas $E_T(n) = n*(n+1)/2$, que si se trata de árboles $E_T(n) = 2*(n+1)*\mathbf{H}_n - 3*n$ (promedio sobre todas las formas de llegada posibles) y si se trata de listas de alojamiento secuencial ordenadas exploradas con sucesivas biseccciones, midiendo el costo con el número de celdas consultadas, resulta ser

$$E_T(n) = n * \log(n+1) + n+1 - 2\hat{}\log(n+1).$$

$\mathbf{H}_n$ describe a la serie armónica hasta su n-ésimo elemento.

Para mayor brevedad de los desarrollos se define:

$$Bin\left(X_n\ ,\ N\right) = \sum_{n=0}^{N} X_n \binom{N}{n}\ p^n\ q^{N-n}$$

Facilita además el desarrollo ver primero los siguientes resultados intermedios:

$$Bin\left(nX_n\ ,\ N\right) = \sum_{n=0}^{N} nX_n\ \frac{N!}{n!\ (N-n)!}\ p^n\ q^{N-n}$$

$$= \sum_{n=1}^{N} nX_n\ \frac{N!}{n!\ (N-n)!}\ p^n\ q^{N-n}$$

$$= pN \sum_{n=1}^{N} X_n \frac{(N-1)!}{(n-1)!\,(N-n)!}\, p^{n-1}\, q^{N-n}$$

y con la sustitución i = n-1 se tiene:

$$= pN \sum_{i=0}^{N-1} X_{i+1} \frac{(N-1)!}{i!\,(N-1-i)!}\, p^{i}\, q^{N-1-i}$$

$$= pN \sum_{i=0}^{N-1} X_{i+1} \binom{N-1}{i} p^{i}\, q^{N-1-i}$$

$$= pN\ Bin\left(X_{n+1}\ ,\ N-1\right) \qquad [\ 1\ ] \quad \blacksquare$$

Usando este resultado se tiene una deducción muy breve del esfuerzo medio para listas vinculadas (el tratamiento habitual del rebalse separado):

$$\frac{M}{N}\ Bin\left(\frac{n(n+1)}{2}\ ,\ N\right) \ =$$

y aplicando el resultado [1] respecto de la primera n

$$= \frac{M}{N}\ pN\ Bin\left(\frac{n+2}{2}\ ,\ N-1\right)$$

pero la probabilidad de un balde p es 1/M. El operador Bin es lineal respecto de su primer operando

$$= Bin\left(1\ ,\ N-1\right)\ +\ Bin\left(n\,\frac{1}{2}\ ,\ N-1\right)$$

El operador Bin con primer argumento constante da esa constante y sobre el segundo Bin aplicamos nuevamente el resultado [1]:

$$= 1\ +\ p(N-1)\ Bin\left(\frac{1}{2}\ ,\ N-2\right)$$

$$= 1 + p(N-1)\frac{1}{2} = 1 + \frac{pN}{2} - \frac{p}{2}$$

$$= 1 + \frac{\rho}{2} - \frac{1}{2M} \cong 1 + \frac{\rho}{2} \quad \blacksquare$$

Segundo resultado intermedio:

$$Bin\left(\frac{1}{n+1}, N\right) = \sum_{n=0}^{N} \frac{1}{n+1}\binom{N}{n} p^n q^{N-n}$$

$$= \frac{1}{p(N+1)} \sum_{n=0}^{N} \frac{N!(N+1)}{(n+1)n!(N-n)!} p^{n+1} q^{N-n}$$

$$= \frac{1}{p(N+1)} \sum_{i=1}^{N+1}\binom{N+1}{i} p^i q^{N+1-i}$$

$$= \frac{1}{p(N+1)} \left[\sum_{i=0}^{N+1}\binom{N+1}{i} p^i q^{N+1-i} - q^{N+1}\right]$$

$$= \frac{1-q^{N+1}}{p(N+1)} \qquad [2] \quad \blacksquare$$

y sustituyendo N por N-1 se obtiene:

$$Bin\left(\frac{1}{n+1}, N-1\right) = \frac{1-q^N}{pN} \qquad [2'] \quad \blacksquare$$

y sustituyendo q por 1-p y desarrollando el binomio se tiene

$$Bin\left(\frac{1}{n+1}, N-1\right) = \frac{1}{N}\sum_{j=1}^{N}\binom{N}{j} p^{j-1}(-1)^{j+1} \qquad [2''] \quad \blacksquare$$

Tercer resultado intermedio:

$$Bin\left(H_n\ ,\ N\right) = \sum_{n=0}^{N} H_n \binom{N}{n} p^n\ q^{N-n}$$

$$= \sum_{n=0}^{N} H_n \binom{N-1}{n-1} p^n\ q^{N-n} + \sum_{n=0}^{N} H_n \binom{N-1}{n} p^n\ q^{N-n}$$

$$= p\sum_{n=1}^{N} H_n \binom{N-1}{n-1} p^{n-1}\ q^{N-n} + q\sum_{n=0}^{N-1} H_n \binom{N-1}{n} p^n\ q^{N-1-n}$$

$$= p\sum_{i=0}^{N-1} H_{i+1} \binom{N-1}{i} p^i\ q^{N-1-i} + q\ Bin\left(H_n\ ,\ N-1\right)$$

$$= p\ Bin\left(\frac{1}{n+1}\ ,\ N-1\right) + p\ Bin\left(H_n\ ,\ N-1\right) +$$

$$+ q\ Bin\left(H_n\ ,\ N-1\right)$$

$$= \frac{1-q^N}{N} + Bin\left(H_n\ ,\ N-1\right)$$

y aplicando reiteradamente la sustitución hasta llegar a 0

$$= \sum_{i=1}^{N} \frac{1-q^i}{i}$$

y sustituyendo q por 1-p y desarrollando los binomios se tiene

$$= \sum_{i=1}^{N} \frac{1-(1-p)^i}{i}$$

$$= \sum_{i=1}^{N} \frac{1}{i} \sum_{j=1}^{i} \binom{i}{j} p^j\ (-1)^{j+1}$$

$$= \sum_{j=1}^{N} p^j \, (-1)^{j+1} \sum_{i=j}^{N} \binom{i}{j} \frac{1}{i}$$

$$= \sum_{j=1}^{N} p^j \, (-1)^{j+1} \frac{1}{j} \binom{N}{j} \qquad [3] \quad \blacksquare$$

sustituyendo N por N-1 se obtiene:

$$Bin\,(H_n\ ,\ N-1) = \sum_{j=1}^{N-1} p^j \, (-1)^{j+1} \frac{1}{j} \binom{N-1}{j}$$

$$= \frac{1}{N} \sum_{j=1}^{N-1} p^j \, (-1)^{j+1} \frac{j+1}{j} \binom{N}{j+1}$$

y por cambio de variable de sumación:

$$= \frac{1}{N} \sum_{j=2}^{N} p^{j-1} \, (-1)^{j} \frac{j}{j-1} \binom{N}{j} \qquad [3'] \quad \blacksquare$$

El esfuerzo medio para rebalse separado manejado con árboles da:

$$= \frac{M}{N} Bin\,(2n\ H_n\ ,\ N) + \frac{M}{N} Bin\,(2\ H_n\ ,\ N) - \frac{M}{N} Bin\,(3n\ ,\ N)$$

$$= 2\,\frac{M}{N}\,pN\ Bin\,(H_{n+1}\ ,\ N-1) + 2\,\frac{M}{N}\,Bin\,(H_n\ ,\ N) - 3\,\frac{M}{N}\,pN$$

En los pasos siguientes se usará frecuentemente el hecho de ser pM = 1.

$$= 2\left[Bin\,(H_n\ ,\ N-1) + Bin\left(\frac{1}{n+1}\ ,\ N-1\right) + \right.$$

$$\left. + \frac{M}{N} Bin\,(H_n\ ,\ N)\right] - 3$$

Y sustituyendo por [3'], [2''] y [3] se obtiene:

$$= 2\left[\frac{1}{N}\sum_{j=2}^{N} p^{j-1}(-1)^{j}\frac{j}{j-1}\binom{N}{j} + \frac{1}{N}\sum_{j=1}^{N} p^{j-1}(-1)^{j+1}\binom{N}{j} + \right.$$

$$\left. + \frac{1}{N}\sum_{j=1}^{N} p^{j-1}(-1)^{j+1}\frac{1}{j}\binom{N}{j}\right] - 3$$

$$= \frac{2}{N}\left[\sum_{j=2}^{N} p^{j-1}(-1)^{j}\binom{N}{j}\left(\frac{j}{j-1} - 1 - \frac{1}{j}\right) + N + N\right] - 3$$

$$= 1 + \frac{2}{N}\sum_{j=2}^{N} p^{j-1}(-1)^{j}\binom{N}{j}\frac{1}{j(j-1)}$$

y haciendo el desarrollo recordando que pN = N/M = ρ (factor de carga):

$$= 1 + 2\left[\frac{1}{1\times 2\times 2!}\left(1-\frac{1}{N}\right)\rho - \frac{1}{2\times 3\times 3!}\left(1-\frac{1}{N}\right)\left(1-\frac{2}{N}\right)\rho^{2} + \right.$$

$$\left. + \frac{1}{3\times 4\times 4!}\left(1-\frac{1}{N}\right)\left(1-\frac{2}{N}\right)\left(1-\frac{3}{N}\right)\rho^{3} - \ldots\right]$$

y como normalmente N es grande:

$$\cong 1 + \frac{2}{1\times 2\times 2!}\rho - \frac{2}{2\times 3\times 3!}\rho^{2} + \frac{2}{3\times 4\times 4!}\rho^{3} - \ldots$$

$$\cong 1 + \frac{\rho}{2} - \frac{\rho^{2}}{18} + \frac{\rho^{3}}{144} - \frac{\rho^{4}}{1200} + \ldots \qquad \blacksquare$$

La búsqueda por bisección al contener función techo de logaritmos no admite un desarrollo similar, por lo cual sólo ha sido tabulada.

En el siguiente cuadro se han tabulado resultados obtenidos por cálculo explícito de las sumatorias con M=100 y M=1000. Este doble cálculo pone en evidencia la baja incidencia de M en el resultado, lo que justifica para la mayoría de los cálculos el uso de las fórmulas aproximadas de caracter asintótico en **N** y **M**.

La primera columna corresponde al factor de carga luego para dos valores de **M** se tabula el esfuerzo medio si se organiza como árbol, como lista vinculada y como lista de alojamiento secuencial con búsqueda binaria.

ρ	**M** = 100			**M** = 1000		
	árbol	lista encad.	búsqueda binaria	árbol	lista encad.	búsqueda binaria
1	1,447	1,495	1,472	1,450	1,500	1,475
2	1,820	1,995	1,873	1,822	2,000	1,876
3	2,135	2,495	2,206	2,137	2,500	2,208
4	2,407	2,995	2,484	2,409	3,000	2,486
5	2,646	3,495	2,721	2,648	3,500	2,722
6	2,859	3,995	2,927	2,861	4,000	2,928
7	3,052	4,495	3,109	3,053	4,500	3,110
8	3,226	4,995	3,272	3,227	5,000	3,273
9	3,387	5,495	3,419	3,388	5,500	3,420
10	3,535	5,995	3,552	3,535	6,000	3,553
15	4,141	8,495	4,075	4,142	8,500	4,076
20	4,603	10,995	4,471	4,603	11,000	4,471

Bibliografía:

[KNU] Knuth, Donald E.
The Art of Computer Programming
Addison-Wesley 1969

[STA] Standish, Thomas A.
Data Structure Techniques
ISBN-0-201-07256-4 1980