Análisis amortizado

M. Andrea Rodríguez-Tastets
Ayudante: Diego Gatica

Universidad de Concepción,Chile
www.inf.udec.cl\ ∼andrea
andrea@udec.cl

I Semestre - 2018

Análisis amortizado : introducción

- En un análisis amortizado, se promedia el tiempo requerido para realizar un secuencia de operaciones de estructura de datos sobre todas las operaciones realizadas.
- Con análisis amortizado se logra mostrar que el costo promedio de un operación puede ser bajo, si es que se promedia sobre todo el conjunto de operaciones, incluso cuando la operación por si sola en la secuencia sea costosa.
- En análisis amortizado la probabilidad no se usa. En este análisis se garantiza el rendimiento promedio de cada operación en el peor caso.

Análisis agregado

- En el análisis agregado se determina el costo en el peor caso $T(n)$ para una secuencia de n operaciones. Entonces, el costo promedio o amortizado por operación es $T(n)/n$.
- Este costo $T(n)/n$ aplica a cualquier operación, incluso cuando las operaciones sean de distinto tipo.

Análisis agregado: MultiPop

Consideremos un operación *MultiPop*(S, k), la cual remueve k elementos de la pila S, sacando todos los elementos de la pila si es que la pila no está vacía.

Multipop(S, k)
  while not *StackEmpty*(S) and $k > 0$ do
   Pop(S)
   $k \leftarrow k - 1$
  endwhile

El tiempo de ejecución de **Multipop**(S, k) es lineal en el número de operaciones *Pop*.

Entonces el costo total de **Multipop**(S, k) es $min(s, k)$, donde s es el número de elementos en la pila.

Análisis agregado: operación pila (cont.)

- Para una secuencia de n operaciones *Push*, *Pop* y *MultiPop* sobre una pila vacía, el peor caso en la secuencia es *MultiPop*, debido a que la pila tiene a lo más n elementos. Ya que en la secuencia cualquier operación puede en el peor caso tener un costo de $O(n)$, entonces la secuencia de n operaciones tiene un costo en el peor caso de $O(n^2)$. Este análisis considera el peor caso en forma individual de cada operación.
- Utilizando análisis amortizado, se puede llegar a una cota más ajustada.
- Cualquier secuencia de n operaciones *Push*, *Pop* y *MultiPop* en una pila inicialmente vacía cuesta a lo más $O(n)$. Esto viene del siguiente análisis. Cada elemento se puede sacar de la pila a lo más una vez por cada vez que es colocado en la pila. Entonces, el número de *Pop* que se pueden llamar es el número de *Push* en una pila que está inicialmente vacía. Entonces una secuencia de n operaciones *Push*, *Pop* y *MultiPop* es $O(n)$ y en promedio por operación es $O(n)/n = O(1)$. Este análisis no considera probabilidades.

Análisis agregado: MultiPush

Consideremos ahora la operación *MultiPush*(S, A), que coloca los $|A|$ elementos en la pila S. ?Qué sucede en este caso?

Multipush(S, A)
  $i \leftarrow 0$
  while $i < |A|$ do
    Push$(S, A[i])$
    $i \leftarrow i + 1$
  endwhile

Análisis agregado: incremento de un contador binario

Consideremos el problema de un contador k-bit binario que se incrementa desde 0. Sea $A[0 \ldots k-1]$ un arreglo de bits, donde k es el largo de A. Un número binario tiene su bit menos significativo en $A[0]$ y el bit más significativos en $A[k-1]$, tal que $x = \sum_{i=0}^{k-1} A[i] \times 2^i$. Inicialmente $x = 0$ y $A[i] = 0$ para $0 \leq i \leq k-1$. Para agregar 1 al valor en el contador, se usa el siguiente procedimiento

Increment(A)
  $i \leftarrow 0$
  while $i < length[A]$ and $A[i] = 1$ do
   $A[i] \leftarrow 0$
   $i \leftarrow i + 1$
  endwhile
  if $i < length[A]$ then
   $A[i] \leftarrow 1$
  endif

El costo de cada llamada a *Increment* es lineal en el número de bits cambiados.

Análisis agregado: incremento de un contador binario (cont.)

- Como en el caso de las n secuencias operaciones sobre una pila vacía, un análisis simple considera que una operación simple de *Increment* toma $\Theta(k)$ en el peor caso, en el que el arreglo A tiene puros 1. Así una secuencia de n operaciones toma $O(nk)$ en el peor caso.
- Un análisis amortizado considera que no todos los bits son cambiados cada vez que se llama a *Increment*: $A[0]$ cambia cada vez que se llama a *Increment*, $A[1]$ cambia cada dos veces ($\lfloor n/2 \rfloor$). Similarmente, bit $A[2]$ cambia cada 4 veces ($\lfloor n/4 \rfloor$) y generalizando, el bit $A[i]$ cambia $\lfloor n/2^i \rfloor$ para $i = 0, 1, \ldots, \lfloor logn \rfloor$. Para $i > \lfloor logn \rfloor$, el bit $A[i]$ no cambia. Entonces, el número de bits que cambian en la secuencia es:

$$\sum_{i=0}^{\lfloor logn \rfloor} \lfloor \frac{n}{2^i} \rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$$

De lo anterior, el costo de secuencia de n operaciones *Increment* es $O(n)$ y el costo amortizado por operación individual es $O(n)/n = O(1)$.

Análisis agregado: ejercicio

Considere una secuencia de n operaciones sobre una estructura de datos. La $i-$ésima operación cuesta i si i es potencia de 2 y 1 en otro caso. Use el análisis agregado para determinar el costo amortizado por operación.

Análisis agregado: Construcción de un heap

Asuma que se quiere crear un max-heap con la secuencia de n inserciones. Para ello considera las n inserciones como una arreglo tal como si este arreglo definiera un árbol binario, es decir, posicin 0 es la raíz, 1 es el hijo izquierdo de la raíz, 2 es el hijo derecho de la raíz, y así sucesivamente.
Para un subárbol de altura h (tomando la altura de abajo hacia arriba), si el subárbol ha sido convertido a heap, entonces el subárbol de altura $h+1$ puede ser convertido en heap haciendo que la raíz baje en el árbol. Entonces, si e número de nodos en altura h está dado por a lo más $\lceil n/2^{h+1} \rceil$, el costo de transformar un subárbol de altura $h+1$ en heap es de orden $O(h)$. Luego el costo total queda dado por

$$\begin{aligned}
\sum_{h=0}^{\lceil logn \rceil} \frac{n}{2^{h+1}} O(h) &= O\left(n \sum_{h=0}^{\lceil logn \rceil} \frac{h}{2^h} \right) \\
&\leq O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h} \right) \leq O(n)
\end{aligned}$$

Método de contabilidad

- En el método de contabilidad del análisis amortizado, se asignan diferentes costos a diferentes operaciones, con algunas operaciones costando más o menos de lo que ellas cuestan.
- La cantidad de costo que asignamos a una operación es llamado su *costo amortizado*.
- Cuando el costo amortizado de una operación particular excede su costo real, la diferencia es asignada a un objeto específico en la estructura de datos como *crédito*.
- El crédito se puede usar luego para compensar que el costo asignado es menor al real. Luego uno puede ver el costo amortizado de una operación como el costo real más el crédito depositado o usado.
- En ese sentido este método es muy diferente al análisis agregado donde todas las operaciones tienen el mismo costo amortizado.

Método de contabilidad (cont)

- Se debe escoger los costos amortizados de las operaciones con cuidado.
- Si se quiere un análisis con costos amortizados para mostrar que, en el peor caso, el costo promedio por operación es pequeño, el total del costo amortizado de una secuencia de operaciones debe ser una cota superior en el costo real total de la secuencia de operaciones.
- Si se denota el costo real de una operación i por c_i y el costo amortizado de una operación por $\hat{c}_i$, se requiere entonces para toda secuencia de n operaciones que

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

- El crédito entonces en la diferencia

$$\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i \geq 0$$

Método de contabilidad: Operaciones de Pila

- Para el problema de la pila se tenía que el costo en el peor caso de *Push* era 1, de *Pop* era 1 y de *MultiPop* era $min(k, s)$, donde k es el argumento y s el tamaño de la pila.
- Asumamos entonces que se tiene los siguientes costos amortizados: *Push* igual a 2, de *Pop* igual a 0 y de *MultiPop* igual a 0. Notar que el costo amortizado de *Multipop* es ahora constante cuando el costo real es variable. Aquí los costos son todos constantes e iguales, aunque en el caso real pueden diferir asintóticamente.
- Se debe mostrar entonces que uno puede pagar por cualquier secuencia de operaciones de la pila usando los costos amortizados. Suponga que se usa un peso por cada unidad de costo. Se comienza con una pila vacía. Cuando se hace un *Push* se usa \$2 para pagar el costo real y se deja \$1 como crédito. Por cada elemento insertado se tiene entonces \$1 de crédito. Cuando se realiza un *Pop* se usa el crédito para pagar la operación, sin cargar adicionalmente el costo total. Como la operación de *Multipop* se basa en *Pop* y no existen más elementos a sacar que los *Push*, entonces la operación *Multipop* no usa ningún costo.
- Entonces para una secuencia de n operaciones *Push*, *Pop* y *Multipop*, el costo amortizado total es una cota superior del costo real con valor $O(n)$.

Método de contabilidad: Incremento de contador

- Como se observó en este ejemplo anterior, el costo de operación es propocional al número de bit que se cambian, lo que se usa como costo en este caso.
- Asumamos un peso de nuevo para representar la unidad de costo.
- Defina que se carga con $2 cuando se cambia a 1 un bit. Cuando se cambia a 0 se usa el otro $1 de crédito dejado por la operación de cambiar a 1. Entonces en cada momento, un bit 1 en el contador tiene un crédito de $1.
- El costo de cambiar bit en el **while** del algoritmo son pagados por el $1 de crédito, entonces sin cargo. En cada llamada del *Increment* a lo más un bit es cambiado a 1 con un costo de $2. El número de bits en el contador no es nunca negativo, y la cantidad de crédito no es nunca negativa.
- Por lo tanto, el costo total amortizado es de $O(n)$, lo cual acota superiormente el costo total real.

Método Potencial

- En vez de representar pre pagos como crédito almacenado, el método potencial del análisis amortizado representa el pre pago como *energía potencial* o solo *potencial*, la cual puede ser liberada para pagar operaciones futuras.
- El potencial es asociado con la estructura de datos completa en vez que con objetos específicos en la estructura, tal como lo hace el método de contabilidad.
- Se parte con una estructura inicial D_o en la cual n operaciones son realizadas. Por cada $i = 1, 2, \ldots, n$, c_i será el costo de la $i-$ésima operación y D_i será la estructura de datos que resulta después de aplicar la $i-$ésima operación a la estructura D_{i-1}.
- Una *función potencial* Φ mapea cada estructura D_i a un número real $\Phi(D_i)$, el cual es el potencial asociado a la estructura D_i.
- El costo amortizado $\hat{c}_i$ de la operación i con respecto a la función potencial Φ es definido por

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$

Método Potencial (cont.)

- El costo amortizado de cada operación es el costo real más el aumento en potencia debido a la operación. El costo total amortizado es entonces:

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)$$

- Si se define una función potencial tal que $\Phi(D_i) \geq \Phi(D_0)$, para cualquier i, entonces el costo amortizado total $\sum_{i=1}^{n} \hat{c}_i$ es una cota superior al costo total real $\sum_{i=1}^{n} c_i$ y garantizamos, al igual que el método de contabilidad, que se paga con anterioridad.
- Es conveniente definir frecuentemente $\Phi(D_0) = 0$ y mostar que $\Phi(D_i) \geq \Phi(D_0)$ para cualquier i.
- Si $\Phi(D_i) \geq \Phi(D_{i-1})$, entonces el costo amortizado $\hat{c}_i$ representa un sobrecargo a la operación i. Por el contrario si $\Phi(D_i) < \Phi(D_{i-1})$ entonces el costo amortizado en un subcargo a la operación i.

Método Potencial: operaciones de una pila

- Para este problema definimos una función potencial Φ para la pila como el número de objetos en la pila. Para la pila vacía D_0, $\Phi(D_0) = 0$.
- Ya que el número de objetos en la pila no es nunca negativo, entonces $\Phi(D_i) \geq \Phi(D_0)$ para cualquier i. Entonces, el costo total amortizado de n operaciones con respecto a Φ representa una cota superior en el costo real.
- Si la operación i sobre la pila con s elementos es un *Push*, entonces la diferencia potencial es

$$\Phi(D_i) - \Phi(D_{i-1}) = (s+1) - s = 1$$

Luego, el costo amortizado es:

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 1 = 2$$

Método Potencial: operaciones de una pila (cont.)

- Si la operación i sobre la pila con s elementos es un *MultiPop* con parámetro k, entonces la diferencia potencial es, con $k' = min(k, s)$

$$\Phi(D_i) - \Phi(D_{i-1}) = -k'$$

  Luego, el costo amortizado es:

$$\hat{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = k' - k' = 0$$

- En forma similar el costo amortizado de una operación *Pop* es 0.
- En consecuencia, el costo amortizado total de las n operaciones es $O(n)$, lo cual constituye el peor caso de n operaciones.

Método Potencial: Incremento del contador binario

- Definamos el potencial del contador después del $i-$ésimo incremento ser b_i, el número de 1s en el contador después de la $i-$ésima operación.
- Suponga que la $i-$ésima operación de incremento cambia t_i bits a 0. El costo actual de la operación es a lo más $t_i + 1$, ya que además de cambiar los t_i bits, cambia a lo más un bit a 1. Si $b_i = 0$, entonces la $i-$ésima operación cambia todos los k bit a cero, y así $b_{i-1} = t_k = k$. Si $b_i > 0$ entonces $b_i = b_{i-1} - t_i + 1$. En ambos casos, $b_i \leq b_{i-1} - t_i + 1$, y la diferencia potencial es:

$$\Phi(D_i) - \Phi(D_{i-1}) \leq (b_{i-1} - t_i + 1) - b_{i-1}$$

  El costo amortizado es entonces:

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \leq (t_i + 1) + (1 - t_i) = 2$$

- Si el contador comienza en cero, entonces $\Phi(D_0) = 0$. Ya que $\Phi(D_i) \geq 0$ para todo i, entonces el total de costo amortizado de una secuencia de n *Increment* es $O(n)$.

Método Potencial: Incremento del contador binario (cont.)

- El método potencial nos da una forma de analizar el contador aunque no comience en 0.
- Hay inicialmente b_0 1-bits, y después de n operaciones de *Increment* hay b_n 1-bits, donde $0 \leq b_0, b_n \leq k$. Entonces podemos reescribir el costo amortizado como

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \hat{c}_i - \Phi(D_n) + \Phi(D_0)$$

- Sabemos que $\hat{c}_i \leq 2$ para $0 \leq i \leq n$. Ya que $\Phi(D_0) = b_0$ y $\Phi(D_n) = b_n$, entonces

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} 2 - b_n + b_0 = 2n - b_n + b_0$$

Ya que $b_0 \leq k$, cuando $k = O(n)$, el costo total real es $O(n)$. En otras palabras. Si se ejecutan al menos $n = \Theta(k)$ operaciones *Increment*, el costo total es $O(n)$, no importante el valor inicial del contador.

Tablas Dinámicas

- En algunos casos es necesario manejar tablas cuyos tamaños varían dinámicamente. Es decir, cuando se hace chica la tabla, se relocaliza en un espacio mayor y todos los elementos almacenados en la tabla original se copian a la nueva tabla. Por el contrario, si casi todos los elementos se han eliminado, se puede achicar la tabla. A esto se le denomina expansión y contracción dinámica de tablas.
- Usando análisis amortizado se muestra que el costo amortizado de inserción y borrado es solo $O(1)$, incluso cuando el costo real de la operación es mayor cuando se debe expandir o contraer la tabla.

Tablas Dinámicas (cont.)

- Se asumen las operaciones *Table* − *Insert* y *Table* − *Delete*.
- Cada elemento ocupa una celda de la tabla.
- La implementación de la tabla es irrelevantes, puede ser un arreglo, una pila, un heap, una tabla hash.
- El factor de carga $\alpha(T)$ define el número de elementos en la tabla dividido por el tamaño.
- Una tabla vacía tiene tamaño 0 y factor de carga 1. Si el factor de carga de una tabla dinámica en acotado bajo una constante, entonces el espacio sin uso de la tabla no es nunca más que una fracción constante del total del espacio.

Expansión de la Tabla

Asuma que usaremos arreglos para una tabla. Una heurística común en estos casos es doblar el tamaño de la tabla cuando ésta se llena. Si solo se hacen inserciones, el factor de carga de la tabla es siempre al menos 1/2, y así la cantidad de espacio perdido nunca excede la mitad de la tabla.
En el siguiente algoritmo, inicialmente la tabla está vacía, donde $num[T] = size[T] = 0$.

Table $-$ **Insert**(T, x)
  if $size[T] = 0$ then
    ubique $table[T]$ con 1 celda
    $size[T] \leftarrow 1$
  endif
  if $num[T] = size[T]$ then
    ubique una *nueva-tabla* con $2size[T]$ celdas
    insertar todos los elementos en T.table en la *nueva-tabla*
    libere $table[T]$
    $table[T] \leftarrow$ *nueva-tabla*
    $size[T] \leftarrow 2size[T]$
  endif
  insertar x en $table[T]$
  $num[T] \leftarrow num[T] + 1$

Expansión de la Tabla: análisis

- En este algoritmo hay dos tipos de inserciones: $Table - Insert$ y las inserciones elementales de elementos. Se asume que $Table - Insert$ tiene un costo lineal para insertar elementos.
- Se considera en este algoritmo que la iniciación de la tabla y la liberación es de orden constante.
- Se llama expansión a la parte del algoritmo que comienza en el segundo **if**.
- Asumamos que existe una secuencia de n $Table - Insert$ en una tabla inicialmente vacía. ¿Cuál es el costo c_i de la $i-$ésima operación? En el análisis simple, ya que una operación tiene un costo en el peor caso de $O(n)$, entonces es $O(n^2)$. Pero esto no es ajustado.

Expansión de la Tabla: análisis (cont.)

Se debe notar que la $i-$ésima operación causa una expansión solo cuando $i-1$ es un potencia de 2. El costo amortizado es de hecho $O(1)$ como se muestra con un análisis agregado.

$$c_i = \begin{cases} i & \text{if } i-1 \text{ es una potencia exacta de } 2 \\ 1 & \text{en otro caso} \end{cases}$$

Entones el costo total es:

$$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lfloor log n \rfloor} 2^j < n + 2n = 3n$$

Expansión de la Tabla: análisis (cont.)

El método de contabiliad puede explicar el costo de 3 por cada operación $Insert - Table$.

- Una unidad de costo se paga por la inserción elementaria en la tabla actual
- Otra unidad de costo se paga cuando se mueve un elemento en la expansión
- La última unidad se paga por otro elemento que ha sido movido una vez cuando la tabla se expande.
- Ejemplo, si el tamaño de la tabla es m después de una expansión, entonces la tabla contiene $m/2$ elemento sin ningún crédito. Se cobra 3 por cada inserción. La inserción elementaria cuesta inmediatamente 1. Otro 1 se deja como crédito. La tercera unidad de costo se deja como crédito por uno de los $m/2$ elementos ya existentes en la tabla. Para el momento en que la tabla se llena m, cada item tiene una unidad de costo para su reinserción.

Expansión de la Tabla: análisis (cont.)

Usando el método potencial.

- Se define una función potencial Φ que es 0 después de una expansión pero aumenta a medida que la tabla se llena de manera que la próxima expansión se paga por el potencial. La función es:

$$\Phi(T) = 2num[T] - size[T]$$

- Después de una expansión, tenemos $num[T] = size[T]/2$ con $\Phi(T) = 0$. Antes de una expansión, tenemos $num[T] = size[T]$ con $\Phi(T) = num[T]$. Ya que $num[T] \geq size[T]/2$, $\Phi(T) \geq 0$. Entonces la suma del costo amortizado es un ajuste superior en la suma del costo real.

Expansión de la Tabla: análisis (cont.)

Usando el método potencial.

- Si la $i-$ésima operación $Table - Insert$ no gatilla una expansión, entonces $size_i = size_{i-1}$ (siendo $size_i$, el tamaño de la tabla después de la operación i) y el costo amortizado es:

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (2num_i - size_i) - (2num_{i-1} - size_{i-1}) = 3$$

- Si la $i-$ésima operación $Table - Insert$ gatilla una expansión, entonces $size_i = 2size_{i-1}$ y $size_{i-1} = num_{i-1} = num_i - 1$, implicando que $size_i = 2(num_i - 1)$. Entonces el costo amortizado es:

$$\begin{aligned} \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} = num_i + (2num_i - size_i) - (2num_{i-1} - size_{i-1}) \\ &= num_i + 2 - (num_i - 1) = 3 \end{aligned}$$

Tabla expansión y contracción

- Cuando la tabla se contrae uno desearía: (1) que el factor de carga sea ajustado por debajo por una constante y (2) que el costo amortizado de la operación de la tabla sea ajustado por encima por una constante. Se asume que el costo se puede medir por el número de operaciones de inserción y borrado.
- Una estrategia natural es que se doble el tamaño de una tabla cuando se llena y que se achique a la mitad cuando el tamaño se amenos que la mitad. Sin embargo, ésta no es una buena estrategia.

Tabla expansión y contracción

Considere el siguiente escenario:

- Se realizan n operaciones sobre una tabla, donde n es exactamente una potencia de 2.
- Las primeras $n/2$ son operaciones de inserción, lo que por nuestro análisis previo toma $O(n)$. Al final de esta secuencia de operaciones se tiene que $num[T] = size[T] = n/2$.
- Para la siguientes $n/2$ operaciones se tienen: I,D,D,I,I, D,D, ..., donde I es inserción y D eliminación. La primera inserción causa una expansión de la tabla a un tamaño n. La dos siguientes operaciones causan una contracción. Dos inserciones luego causan una expansión, y así sucesivamente. Así el costo total es de $O(n^2)$.
- El problema aquí es que después de una expansión no se hacen suficientes borrados para contraer. Al revés, después de una contracción no se hacen suficientes inserciones para pagar una expansión.

Tabla expansión y contracción (cont)

Una estrategia que soluciona este problema es la siguiente:

- Considere que se expande al doble cuando la tabla se llena.
- Pero se achica a la mitad la tabla cuando se tiene menos de $1/4$ de la tabla llena en vez de $1/2$ como antes.
- El factor de carga en entonces ajustado por de bajo por la constante $1/4$.
- Entonces, después de una expansión el factor de carga es $1/2$. Luego la mitad de los elementos deben ser borrados antes de que una contracción ocurra. Similarmente, después de una contracción, el factor de carga es $1/2$, entonces se debe doblar el número de elementos antes de expandir.
- El algoritmo de eliminación es muy similar al de inserción.

Tabla expansión y contracción: análisos

Por el método potencial.

- Sea Φ la función potencial que es 0 inmediatamente después de una expansión o contracción y se construye cuando el factor de carga aumenta a 1 o disminuye a 1/4.
- Sea el factor de carga de una tabla no vacía T $\alpha(T) = num[T]/size[T]$. Para una tabla vacía, $num[T] = size[T] = 0$ con $\alpha(T) = 1$.
- La función potencial se define como:

$$\Phi(T) = \begin{cases} 2num[T] - size[T] & \text{if } \alpha(T) \geq 1/2 \\ size[T]/2 - num[T] & \text{if } \alpha(T) < 1/2 \end{cases}$$

El potencial de la tabla vacía es 0 y nunca negativo.

Tabla expansión y contracción: análisis (cont.)

Por el método potencial.

- Asumamos el potencial después de la operación i:

$$\Phi_i = \begin{cases} 2num_i - size_i & \text{if } \alpha_i \geq 1/2 \\ size_i/2 - num_i & \text{if } \alpha_i < 1/2 \end{cases}$$

Tabla expansión y contracción: análisis (cont.)

Por el método potencial.

- Considere que la $i-$ésima operación es $Table - Insert$. Al igual que el análisis para $\alpha_{i-1} \geq 1/2$, sea que la tabla se expanda o no, el costo es 3. Si $\alpha_{i-1} < 1/2$ la tabla no se puede expandir y el costo amortizado es:

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) = 0$$

Si $\alpha_{i-1} < 1/2$ pero $\alpha_i \geq 1/2$, entonces:

$$\begin{aligned} \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} = 1 + (2num_i - size_i) - (size_{i-1}/2 - num_{i-1}) \\ &= 1 + (2(num_{i-1} + 1) - size_{i-1}) - (size_{i-1}/2 - num_{i-1}) \\ &< 3/2size_{i-1} - 3/2size_{i-1} + 3 = 3 \end{aligned}$$

Tabla expansión y contracción: análisis (cont.)

Por el método potencial.

- Considere que la $i-$ésima operación es $Table - Delete$, con $num_i = num_{i-1} - 1$. Si $\alpha_{i-1} < 1/2$, se debe considerar si la operación causa o no una contracción. Si no la causa, entonces $size_i = size_{i-1}$ y el costo amortizado es:

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) = 2$$

Si $\alpha_{i-1} < 1/2$ y se gatilla una contracción el costo real es $num_i + 1$ ya que se borra un elemento y se mueven num_i elementos. Se tiene que $size_i/2 = size_{i-1}/4 = num_{i-1} = num_i + 1$, y el costo amortizado es:

$$\begin{aligned}\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} = (num_i + 1) + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) \\ &= (num_i + 1) + ((num_i + 1) - num_i) - ((2num_i + 2) - (num_i + 1)) = 1\end{aligned}$$

Cuando $\alpha_{i-1} \geq 1/2$, el costo amortizado es acotado por una constante (se deja como ejercicio). Finalmente el costo de cualquier secuencia de n operaciones sobre una tabla dinámica es $O(n)$.