Análisis amortizado

Alberto Verdejo

Dpto. de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

Diciembre 2008

Bibliografía

- T. H. Cormen, C. E. Leiserson, R. L. Rivest y C. Stein. *Introduction to Algorithms*. Second edition. The MIT Press/McGraw-Hill, 2001.
  Capítulo 17

Motivación

- El análisis en el caso peor de operaciones individuales puede resultar excesivamente pesimista. Ignora que los algoritmos suelen ejecutar secuencias de operaciones.
- El tiempo de ejecución de la secuencia más compleja posible puede ser menor que la cota calculada sumando las cotas de las operaciones individuales.
- Ejemplo: Operaciones con una pila
  - `apilar`, $O(1)$
  - `vaciar`, $O(n)$
  - Secuencia de n operaciones, caso peor $O(n^2)$

Análisis amortizado

- Estudia la complejidad de secuencias de operaciones, generalmente referidas a una misma estructura de datos.
- Ejemplo: Secuencia de m llamadas a un algoritmo para problemas de tamaño n
  - Complejidad amortizada $O(m \log n)$
  - Significa que para cualquier secuencia de m llamadas el tiempo total en el caso peor está acotado superiormente por $m \log n$. Comportamiento "medio" de cada operación en el caso peor: $\log n$.
  - Media sobre la secuencia de operaciones, no sobre los posibles ejemplares de un tamaño dado, como sucede en el coste promedio.
- Se permiten tiempos excesivos para una llamada solo si se han registrado tiempos breves anteriormente. No se amortiza a crédito.
- Dos casos:
  - Una operación, pero cada llamada un coste real distinto.
  - Varias operaciones, con costes distintos.

Análisis amortizado

Método de agregación Se determina una cota superior de la secuencia entera $T(n)$ para n operaciones. El coste medio por operación es $\frac{T(n)}{n}$.

Método de contabilidad Asigna un coste amortizado, posiblemente distinto, por operación. Sobrecarga el coste de algunas operaciones guardando el "prepago" en ciertos elementos de la estructura. Utiliza el crédito para pagar operaciones a las que se ha asignado un coste amortizado menor que su coste real.

Método del potencial Igual que el método de contabilidad, pero guardando el prepago como "potencial" de la estructura de datos en su conjunto, en lugar de asignarlo a elementos concretos de la misma.

Método de agregación

- Se demuestra que *para todo* n, una secuencia de n operaciones necesita un tiempo $T(n)$ en total en el caso peor.
- En el caso peor el coste medio por operación, coste amortizado, es $\dfrac{T(n)}{n}$.
- El coste amortizado es el mismo para todas las operaciones de la secuencia.

Método de agregación, operaciones sobre una pila

- Operaciones sobre una pila:
  - `apilar`, $O(1)$
  - `desapilar`, $O(1)$
  - Coste total de una secuencia de n operaciones, $O(n)$.

Método de agregación, operaciones sobre una pila

- Operaciones sobre una pila:
  - `apilar`, $O(1)$
  - `desapilar`, $O(1)$
  - Coste total de una secuencia de n operaciones, $O(n)$.
- Añadimos `multidesapilar`(p,k): desapila k elementos

proc `multidesapilar`$(p : \textit{pila}, k : \textit{nat})$
    mientras $\neg$`es-vacía?`$(p) \wedge k > 0$ **hacer**
        `desapilar`(p)
        $k := k - 1$
    fmientras
fproc

Método de agregación, operaciones sobre una pila

- Operaciones sobre una pila:
  - `apilar`, $O(1)$
  - `desapilar`, $O(1)$
  - Coste total de una secuencia de n operaciones, $O(n)$.
- Añadimos `multidesapilar`(p,k): desapila k elementos

proc `multidesapilar`$(p$: *pila*$, k$: *nat*$)$
    mientras $\neg$`es-vacía?`$(p) \wedge k > 0$ **hacer**
        `desapilar`(p)
        $k := k - 1$
    fmientras
fproc

- ¿Cuál es el tiempo de ejecución de `multidesapilar`(p,k) sobre una pila de s objetos? El coste real es del orden del $\text{mín}(k,s)$.

Método de agregación, operaciones sobre una pila

- Secuencia de n llamadas a `apilar`, `desapilar` y `multidesapilar` sobre una pila vacía.
- Sin tener en cuenta relaciones entre llamadas:
  - El coste peor de `multidesapilar` está en $O(n)$.
  - El coste en el caso peor de cualquier operación está en $O(n)$.
  - El coste de una secuencia de n operaciones está en $O(n^2)$. Correcto aunque no ajustado.

Método de agregación, operaciones sobre una pila

- Secuencia de n llamadas a `apilar`, `desapilar` y `multidesapilar` sobre una pila vacía.
- Sin tener en cuenta relaciones entre llamadas:
  - El coste peor de `multidesapilar` está en $O(n)$.
  - El coste en el caso peor de cualquier operación está en $O(n)$.
  - El coste de una secuencia de n operaciones está en $O(n^2)$. Correcto aunque no ajustado.
- Analizando las relaciones:
  - Cada elemento apilado solo puede desapilarse una vez.
  - El número de veces que se llama a `desapilar` (contando también las de dentro de `multidesapilar` es como mucho el número de veces que se apila, que como mucho es n.
  - Tiempo total $O(n)$.
  - El coste medio por operación, coste amortizado, es $O(1)$.

Método de agregación, contador binario

- Contador binario de k bits, ascendente desde 0.
- Utilizamos un vector $C[0..k-1]$. Valor $\sum_{j=0}^{k-1} 2^j \cdot C[j]$
- Operación `contar`

$$
\begin{array}{l}
\textbf{proc}\ \texttt{contar}(C[0..k-1]\ \textbf{de}\ \{0,1\}) \\
\quad j := 0 \\
\quad \textbf{mientras}\ j < k \wedge C[j] = 1\ \textbf{hacer} \\
\quad\quad C[j] := 0 \\
\quad\quad j := j+1 \\
\quad \textbf{fmientras} \\
\quad \textbf{si}\ j < k\ \textbf{entonces}\ C[j] := 1\ \textbf{fsi} \\
\textbf{fproc}
\end{array}
$$

- El coste es lineal respecto al número de bits cambiados.
- En el caso peor todos pasan a 0, $O(k)$.
- Secuencia de n llamadas a `contar`: $O(nk)$.

Método de agregación, contador binario

Podemos afinar el análisis teniendo en cuenta que no todos los bits cambian siempre.

x	$A[7]$	$A[6]$	$A[5]$	$A[4]$	$A[3]$	$A[2]$	$A[1]$	$A[0]$	coste
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

Método de agregación, contador binario

$C[0]$ cambia cada vez que se llama a `contar`

$C[1]$ cambia una de cada 2 veces $\rightsquigarrow \lfloor \frac{n}{2} \rfloor$

$C[2]$ cambia una de cada 4 veces $\rightsquigarrow \lfloor \frac{n}{4} \rfloor$

$\vdots$

$C[j]$ cambia $\left\lfloor \frac{n}{2^j} \right\rfloor$ veces, $0 \leq j \leq \lfloor \log n \rfloor$

Método de agregación, contador binario

$C[0]$ cambia cada vez que se llama a `contar`

$C[1]$ cambia una de cada 2 veces $\rightsquigarrow \lfloor \frac{n}{2} \rfloor$

$C[2]$ cambia una de cada 4 veces $\rightsquigarrow \lfloor \frac{n}{4} \rfloor$

$\vdots$

$C[j]$ cambia $\left\lfloor \frac{n}{2^j} \right\rfloor$ veces, $0 \leq j \leq \lfloor \log n \rfloor$

- Número total de cambios $\displaystyle\sum_{j=0}^{\lfloor \log n \rfloor} \left\lfloor \frac{n}{2^j} \right\rfloor < n \sum_{j=0}^{\infty} \frac{1}{2^j} = 2n$
- El coste de n llamadas a contar está en $O(n)$.
- El coste amortizado por operación es $O(1)$.

Método de contabilidad

- Asignar una carga/precio diferente a cada operación.
- A algunas operaciones se les pone un precio mayor o menor que su coste real.
- El coste amortizado será el precio asignado a cada operación.
- Cuando el coste amortizado de una operación excede el coste real, la diferencia se asigna a objetos específicos de la estructura de datos en forma de crédito, que podrá usarse después para ayudar al pago de operaciones cuyo coste amortizado sea menor que su coste real.

$$\text{coste amortizado} = \text{coste real} + \text{crédito } (+/-)$$

$$\hat{c}_i = c_i + \textit{crédito}_i$$

Método de contabilidad

- Si asignamos costes amortizados pequeños, y demostramos que el coste amortizado total de una secuencia de operaciones es cota superior del coste total real, habremos logrado ver que en el caso peor el coste medio de las operaciones es pequeño.

Método de contabilidad

- Si asignamos costes amortizados pequeños, y demostramos que el coste amortizado total de una secuencia de operaciones es cota superior del coste total real, habremos logrado ver que en el caso peor el coste medio de las operaciones es pequeño.
- Para toda secuencia de longitud n, se tiene que cumplir:

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i \Leftrightarrow \underbrace{\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i}_{\text{crédito en la estructura}} \geq 0$$

- El crédito total asociado a la estructura debe ser en todo momento no negativo.

Método de contabilidad, operaciones sobre una pila

Operación	coste real	coste amortizado
`apilar`	1	2
`desapilar`	1	0
`multidesapilar`	$\min(k, s)$	0

Método de contabilidad, operaciones sobre una pila

Operación	coste real	coste amortizado
`apilar`	1	2
`desapilar`	1	0
`multidesapilar`	$\min(k, s)$	0

- Hay que mostrar que podemos pagar cualquier secuencia de operaciones.
- Utilizamos un euro por cada unidad de coste.
- Cuando apilamos un elemento utilizamos un euro para pagar el coste real de apilar, y dejamos el otro junto al elemento apilado.
- En todo momento, cada elemento en la pila tiene un euro pegado, que será utilizado si el elemento se desapila.
- Cuando desapilamos el precio es 0, y utilizamos el euro en el elemento desapilado para pagar el coste real de desapilar.

$$\text{crédito total} = \text{número de elementos}$$

- Por tanto, para cualquier secuencia de n operaciones el coste total amortizado es cota superior del coste total real. Ambos en $O(n)$.

Método de contabilidad, contador binario

- Operaciones básicas: cambio de un bit, coste real 1
- Cambiar un bit de 0 a 1: coste amortizado 2. Uno para cambiarlo y otro para dejarlo encima del 1. Se utilizará cuando vuelva a 0.
- Cambiar un bit de 1 a 0: coste amortizado 0. Pagamos con el euro en el bit.
- En `contar` solo se pone un bit a 1 como mucho. El coste amortizado de contar es como mucho 2.
- En cualquier instante todo 1 del contador tiene un euro encima:

$$\text{crédito total} = \text{número de 1s}$$

- La cantidad de 1s en el contador nunca es negativa.
- Para n operaciones `contar` el coste total amortizado es $O(n)$, que acota el coste total real.

Método del potencial

- Trabajo pagado por adelantado

  Contabilidad: crédito asociado con objetos específicos en la estructura de datos.

  Potencial: “energía potencial” que puede usarse para pagar futuras operaciones. Asociada a la estructura en conjunto
- Empezamos con una estructura inicial D_0 a la que se aplican n operaciones

$$\text{operación } i\text{-ésima} \quad D_{i-1} \rightsquigarrow D_i$$

- Función potencial Φ, $\Phi(D_i) =$ potencial de D_i
- Coste amortizado definido como $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$
- Intuitivamente
  - $\Phi(D_i) - \Phi(D_{i-1}) > 0$, $\hat{c}_i$ representa una sobrecarga de la operación i
  - $\Phi(D_i) - \Phi(D_{i-1}) < 0$, $\hat{c}_i$ representa una infracarga

Método del potencial

- Coste amortizado total

$$
\begin{aligned}
\hat{C}_n &= \textstyle\sum_{i=1}^{n} \hat{c}_i \\
&= \textstyle\sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) \\
&= \textstyle\sum_{i=1}^{n} c_i + \sum_{i=1}^{n} \Phi(D_i) - \sum_{i=1}^{n} \Phi(D_{i-1}) \\
&= C_n + \Phi(D_n) - \Phi(D_0) \\
&\overset{?}{\geq} C_n
\end{aligned}
$$

Método del potencial

- Coste amortizado total

$$
\begin{aligned}
\hat{C}_n &= \textstyle\sum_{i=1}^{n} \hat{c}_i \\
&= \textstyle\sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) \\
&= \textstyle\sum_{i=1}^{n} c_i + \sum_{i=1}^{n} \Phi(D_i) - \sum_{i=1}^{n} \Phi(D_{i-1}) \\
&= C_n + \Phi(D_n) - \Phi(D_0) \\
&\overset{?}{\geq} C_n
\end{aligned}
$$

- Si podemos definir una función potencial Φ tal que $\Phi(D_n) - \Phi(D_0) \geq 0$, entonces el coste amortizado total será una cota superior del coste total real.
- Pedimos $\Phi(D_i) - \Phi(D_0) \geq 0$ para todo i.
- Es conveniente definir $\Phi(D_0) = 0$ y pedir $\Phi(D_i) \geq 0$.

Método del potencial, operaciones sobre una pila

- Definimos $\Phi(D_i) =$ número de elementos en la pila D_i.
- Pila inicial vacía: $\Phi(D_0) = 0$
- El número de elementos nunca es negativo: $\Phi(D_i) \geq 0$

Método del potencial, operaciones sobre una pila

- Definimos $\Phi(D_i)$ = número de elementos en la pila D_i.
- Pila inicial vacía: $\Phi(D_0) = 0$
- El número de elementos nunca es negativo: $\Phi(D_i) \geq 0$
- Calculamos el coste amortizado por operación.
- Si la i-ésima operación, sobre una pila de s elementos, es `apilar`, entonces

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

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

Método del potencial, operaciones sobre una pila

- Definimos $\Phi(D_i)$ = número de elementos en la pila D_i.
- Pila inicial vacía: $\Phi(D_0) = 0$
- El número de elementos nunca es negativo: $\Phi(D_i) \geq 0$
- Calculamos el coste amortizado por operación.
- Si la i-ésima operación, sobre una pila de s elementos, es `apilar`, entonces

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

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

- Si es `desapilar`

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

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

Método del potencial, operaciones sobre una pila

- Si es `multidesapilar`(p,k) y $k' = \text{mín}(s,k)$ (coste real)

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

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

Método del potencial, operaciones sobre una pila

- Si es `multidesapilar`(p,k) y $k' = \min(s,k)$ (coste real)

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

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

- El coste amortizado de las tres operaciones está en $O(1)$ y el coste de una secuencia de n operaciones está en $O(n)$. Cota superior del coste real.
- El coste en el caso peor de una secuencia de n operaciones está en $O(n)$.

Método del potencial, contador binario

- $\Phi(D_i) = b_i =$ número de unos en el contador.
- Supongamos que la i-ésima operación pone a 0 t_i bits
- El coste real es como mucho $t_i + 1$.
- Si $b_i = 0$ es porque la i-ésima operación pone a 0 todos los k bits, y por tanto $b_{i-1} = t_i = k$
- Si $b_i > 0$, $b_i = b_{i-1} - t_i + 1$
- En cualquier caso $b_i \leq b_{i-1} - t_i + 1$
- La diferencia de potencial es

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

- El coste amortizado es

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

- El coste del caso peor de n llamadas a `contar` está en $O(n)$.

Una cola con dos pilas

Queremos implementar una cola utilizando únicamente dos pilas, y las operaciones sobre pilas (de coste constante) `apilar` y `desapilar`. La implementación debe incluir las operaciones `añadir` y `eliminar-primero`, ambas con coste amortizado constante.

Una cola con dos pilas

Queremos implementar una cola utilizando únicamente dos pilas, y las operaciones sobre pilas (de coste constante) `apilar` y `desapilar`. La implementación debe incluir las operaciones `añadir` y `eliminar-primero`, ambas con coste amortizado constante.

tipos *cola* $=$ **reg**
    $\mathit{entran}, \mathit{salen}$: *pila*
  freg

Una cola con dos pilas

Queremos implementar una cola utilizando únicamente dos pilas, y las operaciones sobre pilas (de coste constante) `apilar` y `desapilar`. La implementación debe incluir las operaciones `añadir` y `eliminar-primero`, ambas con coste amortizado constante.

tipos *cola* $=$ **reg**
    $entran, salen$: *pila*
  freg

proc `añadir`$(c$: *cola*, e : *elemento*$)$
  `apilar`$(c.entran, e)$
fproc

Coste en el caso peor: $O(1)$

Una cola con dos pilas

```
proc eliminar-primero(c : cola, e : elemento)
{ e es de salida, y será el primer elemento de c }
  si es-pila-vacía?(c.salen) entonces
    si es-pila-vacía?(c.entran) entonces
      error(Cola vacía)
    si no
      mientras ¬es-pila-vacía?(c.entran) hacer
        desapilar(c.entran, a)
        apilar(c.salen, a)
      fmientras
      desapilar(c.salen, e)
    fsi
  si no
    desapilar(c.salen, e)
  fsi
fproc
```

Una cola con dos pilas

```
proc eliminar-primero(c : cola, e : elemento)
{ e es de salida, y será el primer elemento de c }
   si es-pila-vacía?(c.salen) entonces
      si es-pila-vacía?(c.entran) entonces
         error(Cola vacía)
      si no
         mientras ¬es-pila-vacía?(c.entran) hacer
            desapilar(c.entran, a)
            apilar(c.salen, a)
         fmientras
         desapilar(c.salen, e)
      fsi
   si no
      desapilar(c.salen, e)
   fsi
fproc
```

Coste en el caso peor: *salen* es vacía y *entran* tiene s elementos

$s + 1$ llamadas a `desapilar` y s llamadas a `apilar`

Por tanto, su tiempo de ejecución en el caso peor es $2s + 1$.

Una cola con dos pilas

Método de contabilidad

Operación	coste real	coste amortizado
`añadir`	1	3
`eliminar-primero`	$2s+1$	1

Una cola con dos pilas

Método de contabilidad

Operación	coste real	coste amortizado
`añadir`	1	3
`eliminar-primero`	$2s+1$	1

- Al añadir un elemento gastamos 1 en apilarlo, y dejamos 2 junto al elemento, que serán utilizados si tiene que ser desapilado de la pila *entran* y apilado en la pila *salen*.
- Al eliminar el primero pagamos con el coste amortizado.
- Por tanto, el crédito es el doble del número de elementos en la pila *entran*, que nunca es negativo.
- Por tanto, una secuencia de n operaciones tendrá un coste amortizado en $O(n)$.

Una cola con dos pilas

Método del potencial

- $D_i =$ estado de las pilas después de la i-ésima operación.
- $\Phi(D_i) =$ el doble del número de elementos en la pila $entran$.
- $\Phi(D_0) = 0$ y $\Phi(D_i) \geq 0$ para todo i.

Una cola con dos pilas

Método del potencial

- $D_i =$ estado de las pilas después de la i-ésima operación.
- $\Phi(D_i) =$ el doble del número de elementos en la pila *entran*.
- $\Phi(D_0) = 0$ y $\Phi(D_i) \geq 0$ para todo i.
- Si la i-ésima operación es `añadir`, entonces $\Phi(D_i) - \Phi(D_{i-1}) = 2$. Como el coste real de `añadir` es 1, el coste amortizado de `añadir` es 3.

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 2 = 3$$

Una cola con dos pilas

Método del potencial

- $D_i =$ estado de las pilas después de la i-ésima operación.
- $\Phi(D_i) =$ el doble del número de elementos en la pila *entran*.
- $\Phi(D_0) = 0$ y $\Phi(D_i) \geq 0$ para todo i.
- Si la i-ésima operación es `añadir`, entonces $\Phi(D_i) - \Phi(D_{i-1}) = 2$. Como el coste real de `añadir` es 1, el coste amortizado de `añadir` es 3.

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 2 = 3$$

- Si la i-ésima operación es `eliminar-primero` tenemos que considerar dos casos.
  - Si la pila *salen* no es vacía, entonces $\Phi(D_i) - \Phi(D_{i-1}) = 0$, y el coste real de la operación es 1.
  - Cuando la pila *salen* es vacía, si la pila *entran* tiene s elementos, entonces $\Phi(D_i) - \Phi(D_{i-1}) = -2s$, y el coste real de la operación `eliminar-primero` es $2s + 1$.

  En cualquier caso, el coste amortizado de la operación `eliminar-primero` es 1.
- Por tanto, el coste amortizado de cada operación es $O(1)$.

Tablas dinámicas

- Tablas (vectores) cuyo tamaño puede modificarse según las necesidades.
- Cuando no caben más objetos, la tabla tiene que ser recolocada en memoria con un tamaño mayor. Expansión
- Si se han borrado muchos objetos, recolocar la tabla con menor tamaño. Contracción.

Tablas dinámicas

- Tablas (vectores) cuyo tamaño puede modificarse según las necesidades.
- Cuando no caben más objetos, la tabla tiene que ser recolocada en memoria con un tamaño mayor. Expansión
- Si se han borrado muchos objetos, recolocar la tabla con menor tamaño. Contracción.
- Utilizando análisis amortizado demostraremos que el coste medio de `insertar` y `eliminar` es $O(1)$, aunque se realicen expansiones y contracciones.
- Garantizar que el espacio libre nunca sobrepasa una fracción constante del espacio total.
- Factor de carga

$$\alpha(T) = \frac{\text{n}^{\text{o}} \text{ de elementos almacenados}}{\text{tamaño total}}$$

- Para la tabla vacía: tamaño $= 0$, y $\alpha(T) = 1$.
- Si $\alpha(T)$ está acotado inferiormente por una constante, el espacio libre nunca es más que una fracción constante del total.

$$\alpha(T) \geq \frac{1}{2} \qquad \text{espacio libre} \leq 50\,\%$$

Tablas dinámicas, expansiones

- La tabla T está llena cuando $\alpha(T) = 1$.
- Podemos "expandir" la tabla pidiendo espacio para una tabla mayor y liberando el ocupado. Copiar elementos.
- Heurística común: pedir una tabla el doble de grande.
- Si solo se realizan inserciones, $\alpha(T) \geq \frac{1}{2}$.
- $T.tabla$: los datos (puntero al bloque de memoria)
  $T.num$: número de elementos en la tabla
  $T.tam$: número total de huecos
  Inicialmente: $T.num = T.tam = 0$

Tablas dinámicas, expansiones

```
proc insertar(T, x)
    si T.tam = 0 entonces
        T.tabla := reservar(1)
        T.tam := 1
    si no
        si T.num = T.tam entonces
            tabla-nueva := reservar(2 * T.tam)
            mover elementos de T.tabla a tabla-nueva
            liberar(T.tabla)
            T.tabla := tabla-nueva
            T.tam := 2 * T.tam
        fsi
    fsi
    insertar-elem(T.tabla, x)
    T.num := T.num + 1
fproc
```

Tablas dinámicas, expansiones

- Analizamos el coste en función del número de inserciones elementales. $O(T.tam)$ en el caso peor.
- Analizar secuencia de n llamadas a `insertar`
- ¿Cuál es el coste de la i-ésima operación?
- Si hay sitio, $c_i = 1$
- Si la tabla está llena, $c_i = i = (i-1)+1$
- Con n llamadas, el coste en el caso peor de una llamada es $O(n)$. Por tanto, el total está en $O(n^2)$.
- La i-ésima operación causa una expansión cuando $i-1$ es potencia de 2.

Método de agregación

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

El coste total de n operaciones es

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

El coste amortizado por operación es $O(1)$.

Tablas dinámicas, expansiones

Método de contabilidad

Coste amortizado 3: 1 para insertar el elemento en la tabla actual
1 para moverlo cuando la tabla se expande
1 para mover otro elemento que ya se había movido

Tablas dinámicas, expansiones

Método de contabilidad

Coste amortizado 3: 1 para insertar el elemento en la tabla actual
1 para moverlo cuando la tabla se expande
1 para mover otro elemento que ya se había movido

Justo después de expandir, tamaño m, crédito 0

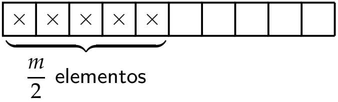
$$\underbrace{\begin{array}{|c|c|c|c|c|}\hline \times & \times & \times & \times & \times \\ \hline\end{array}}_{\frac{m}{2}\text{ elementos}}\begin{array}{|c|c|c|c|c|}\hline & & & & \\ \hline\end{array}$$

Tablas dinámicas, expansiones

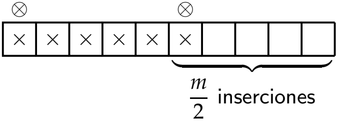
Método de contabilidad

Coste amortizado 3: 1 para insertar el elemento en la tabla actual
1 para moverlo cuando la tabla se expande
1 para mover otro elemento que ya se había movido

Justo después de expandir, tamaño m, crédito 0



Después de insertar una vez, crédito 2



Cuando la tabla vuelve a estar llena, todos los elementos tienen una moneda.
Se puede pagar la expansión.

Método del potencial

Nos interesa que el potencial valga 0 después de expandir y que crezca hasta ser el tamaño de la tabla cuando esta se llena.

$$\Phi(T) = 2 \cdot T.num - T.tam$$

Justo después de una expansión $T.num = \frac{T.tam}{2} \Rightarrow \Phi(T) = 0$

$\Phi(T_0) = 0$

Como siempre $T.num \geq \frac{T.tam}{2}$, $\Phi(T_i) \geq 0$.

Por tanto, la suma de costes amortizados es cota superior del coste total.

Tablas dinámicas, expansiones

Veamos el coste de la operación i-ésima.

$num_i =$ número de elementos después de i-ésima operación.

$tam_i =$ tamaño después de i-ésima operación.

$\Phi_i =$ potencial después de i-ésima operación.

Inicialmente, $num_0 = 0$, $tam_0 = 0$ y $\Phi_0 = 0$

Tablas dinámicas, expansiones

Veamos el coste de la operación i-ésima.

$num_i =$ número de elementos después de i-ésima operación.

$tam_i =$ tamaño después de i-ésima operación.

$\Phi_i =$ potencial después de i-ésima operación.

Inicialmente, $num_0 = 0$, $tam_0 = 0$ y $\Phi_0 = 0$

- Si la i-ésima operación no dispara una expansión

$$tam_i = tam_{i-1} \qquad num_i = num_{i-1} + 1$$

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

Tablas dinámicas, expansiones

- Si la i-ésima operación dispara una expansión

$$\frac{tam_i}{2} = tam_{i-1} = num_{i-1} = num_i - 1$$

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

Una secuencia de n operaciones tiene un coste en $O(n)$.

Tablas dinámicas, expansiones y contracciones

- “Contraer” la tabla cuando el factor de carga se haga muy pequeño, de tal forma que no se malgaste mucho espacio.
- Cuando el número de elementos es muy pequeño, se pide una tabla de espacio menor, se copian los elementos y se libera la tabla antigua.
- Preservar dos propiedades:
  - el factor de carga está acotado inferiormente por una constante.
  - el coste amortizado por operación está acotado superiormente por una constante.
- Medimos el coste en términos del número de inserciones y eliminaciones elementales.

Tablas dinámicas, expansiones y contracciones

- Posible estrategia:
  - Expansión: doblar el tamaño cuando la tabla esté llena.
  - Contracción: reducir a la mitad el tamaño cuando la tabla esté menos llena de la mitad.
- Se consigue $\alpha(T) \geq \frac{1}{2}$.
- Pero el coste amortizado es "grande".

  Ejemplo: n operaciones, n potencia de 2

  $\frac{n}{2}$ primeras operaciones son `insertar` $\rightsquigarrow \Theta(n)$

  Al final de las $\frac{n}{2}$ operaciones $T.num = T.tam = \frac{n}{2}$

  El resto de las $\frac{n}{2}$ operaciones son:

  I E E I I E ...

  Total $\Theta(n^2)$
- El problema de la estrategia es obvio: después de una expansión no realizamos suficientes eliminaciones para pagar la contracción (y viceversa).

Tablas dinámicas, expansiones y contracciones

- Nueva estrategia:
  - Expansión: doblar el tamaño cuando la tabla esté llena.
  - Contracción: reducir a la mitad el tamaño cuando una eliminación causa $\alpha(T) < \frac{1}{4}$.
- Se consigue $\alpha(T) \geq \frac{1}{4}$.
- Intuitivamente:
  - Después de una expansión, $\alpha(T) = \frac{1}{2}$, y tienen que borrarse la mitad de estos elementos para que haga falta una contracción ($\alpha(T) < \frac{1}{4}$).
  - Después de una contracción, $\alpha(T) = \frac{1}{2}$, por lo que el número de elementos debe doblarse antes de que haga falta una expansión ($\alpha(T) = 1$).

Método del potencial

Nos interesa que Φ valga 0 justo después de una expansión o contracción y que vaya aumentando cuando α crece hasta 1 o decrece hasta $\frac{1}{4}$.

$T.num = \alpha(T) \cdot T.tam$, $\quad T$ vacía o no

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.tam & \text{si } \alpha(T) \geq \frac{1}{2} \\ \dfrac{T.tam}{2} - T.num & \text{si } \alpha(T) < \frac{1}{2} \end{cases}$$

Método del potencial

Nos interesa que Φ valga 0 justo después de una expansión o contracción y que vaya aumentando cuando α crece hasta 1 o decrece hasta $\frac{1}{4}$.

$T.num = \alpha(T) \cdot T.tam$, $\quad T$ vacía o no

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.tam & \text{si } \alpha(T) \geq \frac{1}{2} \\ \dfrac{T.tam}{2} - T.num & \text{si } \alpha(T) < \frac{1}{2} \end{cases}$$

Propiedades:

- $\Phi(\text{T. vacía}) = 0 \qquad \Phi(T_i) \geq 0$
- $\alpha(T) = \frac{1}{2} \Rightarrow \Phi(T) = 0$
- $\alpha(T) = 1 \Rightarrow \Phi(T) = T.num = T.tam$. Se puede pagar la expansión.
- $\alpha(T) = \frac{1}{4} \Rightarrow T.tam = 4 \cdot T.num \Rightarrow \Phi(T) = T.num$. Se puede pagar la contracción.

Tablas dinámicas, expansiones y contracciones

Notación: $\hat{c}_i$, c_i (de la i-ésima operación)

num_i, tam_i, α_i, Φ_i (de T después de la i-ésima operación)

Inicialmente: $num_0 = 0$, $tam_0 = 0$, $\alpha_0 = 1$, $\Phi_0 = 0$

Tablas dinámicas, expansiones y contracciones

Notación: $\hat{c}_i$, c_i (de la i-ésima operación)

num_i, tam_i, α_i, Φ_i (de T después de la i-ésima operación)

Inicialmente: $num_0 = 0$, $tam_0 = 0$, $\alpha_0 = 1$, $\Phi_0 = 0$

- Si i-ésima operación es `insertar`

  $\alpha_{i-1} \geq \frac{1}{2}$ Igual que antes. $\hat{c}_i = 3$

Tablas dinámicas, expansiones y contracciones

Notación: $\hat{c}_i$, c_i (de la i-ésima operación)

num_i, tam_i, α_i, Φ_i (de T después de la i-ésima operación)

Inicialmente: $num_0 = 0$, $tam_0 = 0$, $\alpha_0 = 1$, $\Phi_0 = 0$

- Si i-ésima operación es `insertar`

  $\alpha_{i-1} \geq \frac{1}{2}$ Igual que antes. $\hat{c}_i = 3$

  $\alpha_{i-1} < \frac{1}{2}$ La tabla no se va a expandir.

  - Si $\alpha_i < \frac{1}{2}$ también:

$$
\begin{aligned}
\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + \left(\frac{tam_i}{2} - num_i\right) - \left(\frac{tam_{i-1}}{2} - num_{i-1}\right) \\
&= 1 + \left(\frac{tam_i}{2} - num_i\right) - \left(\frac{tam_i}{2} - (num_i - 1)\right) \\
&= 0
\end{aligned}
$$

Tablas dinámicas, expansiones y contracciones

$\alpha_{i-1} < \frac{1}{2}$

- Si $\alpha_i \geq \frac{1}{2}$:

$$
\begin{aligned}
\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + (2 \cdot num_i - tam_i) - \left(\frac{tam_{i-1}}{2} - num_{i-1}\right) \\
&= 1 + \Big(2 \cdot (num_{i-1} + 1) - tam_{i-1}\Big) - \left(\frac{tam_{i-1}}{2} - num_{i-1}\right) \\
&= 3 \cdot num_{i-1} - \frac{3}{2} tam_{i-1} + 3 \\
&= 3 \cdot \alpha_{i-1} \cdot tam_{i-1} - \frac{3}{2} tam_{i-1} + 3 \\
&< \frac{3}{2} \cdot tam_{i-1} - \frac{3}{2} tam_{i-1} + 3 \\
&= 3
\end{aligned}
$$

Si i-ésima operación es `eliminar`. $num_i = num_{i-1} - 1$

$\alpha_{i-1} < \frac{1}{2}$

- No causa contracción. $tam_i = tam_{i-1}$

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + \left(\frac{tam_i}{2} - num_i\right) - \left(\frac{tam_{i-1}}{2} - num_{i-1}\right) \\
&= 1 + \left(\frac{tam_i}{2} - num_i\right) - \left(\frac{tam_i}{2} - (num_i + 1)\right) \\
&= 2
\end{aligned}$$

$\alpha_{i-1} < \frac{1}{2}$

- Causa contracción.

$$c_i = num_i + 1 \qquad \frac{tam_i}{2} = \frac{tam_{i-1}}{4} = num_i + 1$$

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= num_i + 1 + \left(\frac{tam_i}{2} - num_i\right) - \left(\frac{tam_{i-1}}{2} - num_{i-1}\right) \\
&= num_i + 1 + \Big((num_i + 1) - num_i\Big) - \Big((2 \cdot num_i + 2) - (num_i + 1)\Big) \\
&= 1
\end{aligned}$$

Tablas dinámicas, expansiones y contracciones

$\alpha_{i-1} < \frac{1}{2}$

- Causa contracción.

$$c_i = num_i + 1 \qquad \frac{tam_i}{2} = \frac{tam_{i-1}}{4} = num_i + 1$$

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= num_i + 1 + \left(\frac{tam_i}{2} - num_i\right) - \left(\frac{tam_{i-1}}{2} - num_{i-1}\right) \\
&= num_i + 1 + \Big((num_i + 1) - num_i\Big) - \Big((2 \cdot num_i + 2) - (num_i + 1)\Big) \\
&= 1
\end{aligned}$$

$\alpha_{i-1} \geq \frac{1}{2}$ No hay contracción. Hacer como ejercicio.