# Análisis amortizado

de estructuras de datos

10-12-2018

Basado en las transparencias de

Javier Campos (modificaciones EM)

# Contenido

- **Introducción: ¿por qué necesitamos otro análisis de complejidad?**
- Método agregado
- Método contable
- Potencial
- Ejemplo: tablas hash

# ¿Por qué necesitamos otro análisis de complejidad?

- Pensemos en las tablas hash:
  - Memoria estática
  - Con expansión de la tabla cuando se llena

- Se consideran muy eficientes, operaciones en tiempo constante
  - ¿Es eso cierto?
  - ¿Por qué nos lo creemos?

# Introducción: conceptos básicos

- "Amortización" (*Wikipedia*):

Bla bla bla… %& $~€ €¬&@#... bla bla… %$€ ~€¬& %€… bla bla… $~€ € %$€… bla bla… ~€¬& #~%¬ /%#€¬)6& &@# … Se trata de técnicas aritméticas para repartir un importe determinado, el valor a amortizar, en varias cuotas, correspondientes a varios periodos. $~€ €¬&@#... bla bla bla… %& $~€ €¬&@#... bla bla… %$€ ~€¬& %€… bla bla… $~€ € %$€ ~€¬& ¬ &@#... %& $~€ €¬&@#... bla bla… %$€ ~€¬& %€… bla bla… $~€ € %$€ ~€¬& ¬ &@# …

# Introducción: conceptos básicos

- Análisis amortizado
  - **Cálculo del coste medio de una operación** obtenido dividiendo el coste en el caso peor de la ejecución de una **secuencia de operaciones** (no necesariamente de igual tipo) dividido por el número de operaciones
- Utilidad:
  - Es posible que el coste en el caso peor de la ejecución aislada de una operación sea muy alto y sin embargo si se considera una secuencia de operaciones el coste promedio disminuya
- Nota:
  - No es un análisis del caso promedio tal y como el que hemos visto en asignaturas anteriores (la probabilidad ahora no interviene)

# Introducción: conceptos básicos

- Análisis amortizado
  - **Cálculo del coste medio EN TIEMPO de una operación** obtenido dividiendo el coste en el caso peor de la ejecución de una **secuencia de operaciones** (no necesariamente de igual tipo) dividido por el número de operaciones
- Importante: Para que tener cotas interesantes, es útil considerar sólo secuencias de operaciones que **pueden ocurrir**, por ejemplo, no se puede borrar en una estructura vacía
- Dada una serie de n operaciones, se trata de estimar $\sum_{i=1}^{n} C(i)$

donde C(i) es el coste de la operación i

# Introducción: conceptos básicos

- En realidad el coste amortizado de una operación es un "artificio contable" que no tiene ninguna relación con el coste real de la operación.
  – El coste amortizado de una operación puede definirse como **cualquier cosa** con la única condición de que considerando una secuencia de *n* operaciones:

$$\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$$

  donde $A(i)$ y $C(i)$ son el coste amortizado y el coste exacto, respectivamente, de la operación $i$-ésima de la secuencia.

# Cómo estimar el coste amortizado

- Objetivo: probar que el coste amortizado es bajo (en muchos casos O(1))

- Para ello: definir A(i) de forma sencilla y cumpliendo $\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$

- Vamos a ver tres métodos para estimar el coste amortizado:
  - Agregado
  - Contable
  - Potencial

# Resumen introducción

- El coste amortizado de una secuencia de n operaciones es

$$\sum_{i=1}^{n} C(i) \quad /n$$

donde C(i) es el coste de la operación i

- Se trata de acotar este coste encontrando A(i) que cumpla

$$\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$$

- Muy útil: tener en cuenta sólo secuencias válidas de operaciones

# Contenido

- Introducción: ¿por qué necesitamos otro análisis de complejidad?
- **Método agregado**
- Método contable
- Potencial
- Ejemplo: tablas hash

- **Método agregado**
  - Consiste en calcular el coste en el caso peor

    $$T(n) = \sum_{i=1}^{n} C(i)$$

    de una secuencia de *n* operaciones, no necesariamente del mismo tipo, y calcular el coste medio o *coste amortizado* de una operación como $T(n)/n$**.**
  - Los otros dos métodos que veremos (contable y potencial ← este no lo veremos) calculan un coste amortizado específico para cada tipo de operación.

# Método agregado

- Ejemplo: pila con operación de *multiDesapilar*
  - Considerar una pila representada mediante una lista de registros encadenados con punteros y con las operaciones de *creaVacía*, *apilar*, *desapilar* y *esVacía*.
  - El coste de todas esas operaciones es $\Theta(1)$ y, por tanto, el coste de una secuencia de $n$ operaciones de *apilar* y *desapilar* es $\Theta(n)$.
  - Añadimos la operación ***multiDesapilar*(*p*,*k*)**, que elimina los $k$ elementos superiores de la pila $p$, si los hay, o deja la pila vacía si no hay tantos elementos.

```
algoritmo multiDesapilar(p,k)
principio
  mq not esVacía(p) and k≠0 hacer
    desapilar(p); k:=k-1
  fmq
fin
```

# Método agregado

– El coste de *multiDesapilar* es, obviamente, $\Theta(\min(h,k))$, si $h$ es la altura de la pila antes de la operación.

– ¿Cuál es el coste de una **secuencia de *n* operaciones** de *apilar*, *desapilar* o *multiDesapilar*?

- La altura máxima de la pila puede ser de orden $n$, así que el coste máximo de una operación de *multiDesapilar* en esa secuencia puede ser $O(n)$.
- Por tanto, el coste máximo de una secuencia de $n$ operaciones está acotado por $O(n^2)$.
- Esta cálculo es correcto, pero la cota $O(n^2)$, obtenida *considerando el caso peor de cada operación de la secuencia*, no es ajustada.
- El método agregado considera el caso peor de la **secuencia** de forma conjunta…

# Método agregado

– Análisis agregado de la secuencia de operaciones:
  - Cada elemento puede ser desapilado como máximo una sola vez en toda la secuencia de operaciones.
  - Por tanto, el máximo número de veces que la operación *desapilar* puede ser ejecutada en una secuencia de $n$ operaciones (incluyendo las llamadas en *multiDesapilar*) es igual al máximo número de veces que se puede ejecutar la operación *apilar*, que es $n$.
  - Es decir, el coste total de cualquier secuencia de $n$ operaciones de *apilar*, *desapilar* o *multiDesapilar* es $O(n)$.
  - Por tanto, el *coste amortizado* de cada operación es la media: $O(n)/n = O(1)$.

# Método agregado

- Otro ejemplo: incrementar un contador binario
  - Considerar un contador binario de ***k* bits**, almacenado en un vector $A[0..k-1]$ de bits.
  - La cifra menos significativa se guarda en $A[0]$, por tanto, el número almacenado en el contador es:

$$x = \sum_{i=0}^{k-1} A[i] \cdot 2^i$$

  - Inicialmente, $x = 0$, es decir, $A[i] = 0$, para todo $i$.
  - Para añadir 1 (módulo $2^k$) al contador se ejecuta el algoritmo *incrementar*…

# Método agregado

```
algoritmo incrementar(A)
principio
  i:=0;
  mq i<length(A) and A[i]=1 hacer
    A[i]:=0;
    i:=i+1
  fmq;
  si i<length(A) entonces
    A[i]:=1
  fsi
fin
```

– Es, esencialmente, el algoritmo implementado en hardware en un sumador en serie con acarreo (“ripple-carry”).

# Método agregado

– Una secuencia de 16 incrementos desde $x = 0$:

| $x$ | $A[7]$ | | | . . . | | | | $A[0]$ | coste acumulado |
|---|---|---|---|---|---|---|---|---|---|
| 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 |

en relieve: los bits que cambian para pasar al siguiente valor del contador

el coste de cada incremento es lineal en el número de bits que cambian de valor

nótese que el coste acumulado nunca es mayor del doble del número de incrementos

# Método agregado

- Primera aproximación al análisis:
  - Una ejecución de *incrementar* tiene un coste $\Theta(k)$ en el peor caso (cuando el vector contiene todo 1's).
  - Por tanto, una secuencia de $n$ incrementos empezando desde 0 tiene un coste $O(nk)$ en el caso peor.
  - Este análisis es correcto pero poco ajustado.
- Análisis agregado de la secuencia de incrementos:
  - $A[0]$ cambia de valor en cada incremento
  - $A[1]$ cambia de valor $\lfloor n/2 \rfloor$ veces en una secuencia de $n$ incrementos
  - $A[2]$ cambia $\lfloor n/4 \rfloor$ veces en la misma secuencia
  - En general, $A[i]$ cambia $\lfloor n/2^i \rfloor$ veces, $i = 0, 1, \ldots, \lfloor \log n \rfloor$

# Método agregado

- Por tanto, el número total de cambios de bit en toda la secuencia es:

$$\sum_{i=0}^{\infty} c^i = \frac{1}{1-c}$$

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

- Es decir, el coste total de toda la secuencia es en el peor caso $O(n)$ y, por tanto, el coste amortizado de cada operación es $O(n)/n = O(1)$.

# Coste agregado

- Agregado= cálculo exacto del coste de n operaciones

- $A(n)= \sum^{n}_{i=1} C(i) /n$

$$\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$$

# Resumiendo

- Método agregado: cálculo exacto

$$A(n) = \sum^{n}_{i=1} C(i) / n$$

$$\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$$

# Contenido

- Introducción: ¿por qué necesitamos otro análisis de complejidad?
- Método agregado
- **Método contable**
- Potencial
- Ejemplo: tablas hash

# Método contable

$$\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$$

| i | Queda P(i) | Pago A(i) | Gasto C(i) |
|---|---|---|---|
| 0 | 0 | | |
| 1 | | | C(1) |
| | | | |
| i | =P(i-1)+A(i)-C(i) | A(i) | C(i) |
| n | | | C(n) |

- Objetivo: Fijar el pago (A(i)) para que P(i) siempre quede $\geq 0$

# Análisis amortizado de ED: conceptos básicos

- **Método contable**
  - El coste amortizado **de cada operación** es un *precio* que se le asigna y puede ser mayor o menor que el coste real de la operación.
  - Cuando el precio de una operación excede su coste real, el *crédito* resultante se puede usar después para pagar operaciones cuyo precio sea menor que su coste real.
  - Puede definirse una *función de potencial*, $P(i)$, para cada operación de la secuencia:

    $$P(i) = A(i) - C(i) + P(i-1), \quad i = 1, 2, \ldots, n$$

    donde $A(i)$ y $C(i)$ son el coste amortizado y el coste exacto, respectivamente, de la operación $i$-ésima.
  - El potencial en cada operación es el crédito disponible para el resto de la secuencia.

# Método contable

- Sumando el potencial de todas las operaciones:

$$\sum_{i=1}^{n} P(i) = \sum_{i=1}^{n} \big( A(i) - C(i) + P(i-1) \big)$$

$$\Rightarrow \sum_{i=1}^{n} \big( P(i) - P(i-1) \big) = \sum_{i=1}^{n} \big( A(i) - C(i) \big)$$

$$\Rightarrow P(n) - P(0) = \sum_{i=1}^{n} \big( A(i) - C(i) \big) \Rightarrow \underline{P(n) - P(0) \geq 0}$$

(por definición de coste amortizado)

$$\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$$

- Debe asignarse un precio a cada operación que haga que **el crédito disponible sea siempre no negativo.**

# Método contable

- Volvamos al ejemplo de la pila con la operación de *multiDesapilar*
  - Recordar el coste real:
    - $C(apilar) = 1$
    - $C(desapilar) = 1$
    - $C(multiDesapilar) = \Theta(\min(h,k))$
  - Asignamos (arbitrariamente) el coste amortizado (*precio*) de cada operación como:
    - $A(apilar) = 2$
    - $A(desapilar) = 0$
    - $A(multiDesapilar) = 0$
  - Para ver si el coste amortizado es correcto hay que demostrar que el crédito es siempre no negativo.

# Método contable

- Es decir, hay que ver si $P(n) - P(0) \geq 0, \forall n$.
- Al apilar cada elemento, con precio 2, pagamos el coste real de una unidad por la operación de apilar y nos sobra otra unidad como crédito.
  - En cada instante de tiempo tenemos una unidad de crédito por cada elemento de la pila.
  - Es el *pre-pago* para cuando haya que desapilarlo.
- Al desapilar, el precio de la operación es cero y el coste real se paga con el crédito asociado al elemento desapilado.
- De esta forma, pagando un poco más por *apilar* (2 en lugar de 1) no hemos necesitado pagar por *desapilar* ni por *multiDesapilar*.

# Método contable

- El otro ejemplo: el contador binario
  - Definimos como 2 unidades el coste amortizado (precio) de la operación de poner un bit a 1.
  - Cuando un bit se pone a 1, se paga su coste de una unidad y la unidad restante queda como crédito para operaciones futuras.
  - Así, cada bit que vale 1 guarda asociado un crédito de una unidad, y ese crédito se usa para pagar la operación de ponerlo a 0, con lo que el coste amortizado de esta operación se puede dejar como 0.

# Método contable

– Veamos ahora el coste amortizado de *incrementar*:

```
algoritmo incrementar(A)
principio
  i:=0;
  mq i<length(A) and A[i]=1 hacer
    A[i]:=0;
    i:=i+1
  fmq;
  si i<length(A) entonces
    A[i]:=1
  fsi
fin
```

coste amortizado nulo

coste amortizado menor o igual que 2

# Resumiendo

- Método agregado: cálculo exacto

- Método contable: inventar un A(i) sencillo para que cuadren las cuentas

$$\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$$

# Contenido

- Introducción: ¿por qué necesitamos otro análisis de complejidad?
- Método agregado
- Método contable
- **Potencial**
- Ejemplo: tablas hash

# Método potencial

$$\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$$

| i | Queda P(i) | Pago A(i) | Gasto C(i) |
|---|---|---|---|
| 0 | 0 | | |
| 1 | | | C(1) |
| | | | |
| i | | = C(i)+ P(i)-P(i-1) | C(i) |
| n | | | C(n) |

•Objetivo: Fijar el queda (P(i)) siempre >=0

# Análisis amortizado de ED: conceptos básicos

- **Método potencial**
  - Consideramos, de nuevo, una función de potencial:

    $P(i) = A(i) - C(i) + P(i-1), \;\; i = 1, 2, \ldots, n \;^{(*)}$

    tal que:

    $P(n) - P(0) \geq 0, \; \forall n$

  - En este método de análisis, se parte de una función de potencial "dada" (que hay que elegir) y usando la ecuación $^{(*)}$ se calcula $A(i)$:

    $A(i) = C(i) + P(i) - P(i-1), \;\; i = 1, 2, \ldots, n$

  - El gran problema: ¿cómo elegir el potencial?

# Método potencial

- Una vez más: la pila con *multiDesapilar*
  – La función de potencial puede interpretarse a menudo como una "energía potencial" asociada a la estructura de datos del problema que puede utilizarse para pagar el coste de las operaciones futuras.

  – Ejemplo: definir el potencial de la pila como su altura.
    - Se tiene: $P(0) = 0$ (pila vacía), y $P(n) \geq 0, \forall n$.
    - Por tanto la función de potencial es válida y por eso el coste amortizado total de las $n$ operaciones es una cota superior del coste total exacto de todas ellas.

# Método potencial

– Cálculo del coste amortizado a partir del potencial:
  - Suponer que la operación *i*-ésima es *apilar* y se realiza sobre una pila de altura *h*:

    $$A(i) = C(i) + P(i) - P(i-1) = 1 + (h+1) - h = 2$$

  - Suponer que la operación *i*-ésima es *desapilar* y se realiza sobre una pila de altura *h*:

    $$A(i) = C(i) + P(i) - P(i-1) = 1 + (h-1) - h = 0$$

  - Suponer que la operación *i*-ésima es *multiDesapilar k* elementos y se realiza sobre una pila de altura *h*, entonces se desapilan $k' = \min(k,h)$ elementos, y:

    $$A(i) = C(i) + P(i) - P(i-1) = k' - k' = 0$$

– El coste amortizado en los tres casos es $O(1)$, por tanto el coste amortizado total de toda la secuencia es $O(n)$.

# Método potencial

- Y por último: el ejemplo del contador binario
  - Elección de la función de potencial:
    - $P(i) = b_i$, el número de 1's en el contador después del $i$-ésimo incremento.
    - Se tiene: $P(0) = 0$ (el contador se inicializa a 0), $P(n) \geq 0\ \forall n$.
  - Cálculo del coste amortizado:
    - Suponer que la $i$-ésima operación (incremento) pone a 0 $t_i$ bits
    - Entonces, su coste es como mucho $t_i + 1$
    - $b_i \leq b_{i-1} - t_i + 1$

```
algoritmo incrementar(A)
principio
  i:=0;
  mq i<length(A) and A[i]=1 hacer
    A[i]:=0;
    i:=i+1
  fmq;
  si i<length(A) entonces
    A[i]:=1
  fsi
fin
```

# Método potencial

– Cálculo del coste amortizado (cont.):
  - $A(i) = C(i) + P(i) – P(i – 1) \le (t_i + 1) + (b_{i\text{-}1} – t_i + 1) – b_{i\text{-}1}$
    $= 2$
  - Por tanto, el coste amortizado total de $n$ incrementos consecutivos empezando con el contador a 0 es $O(n)$.

– El método potencial permite también calcular el coste si el contador empieza en un valor no nulo:
  - Si inicialmente hay $b_0$ bits a 1 y tras $n$ incrementos hay $b_n$ 1's.
  - $A(i) \le 2$, al igual que antes.
  - $A(i) = C(i) + P(i) – P(i –1)$, $i = 1, 2, \ldots, n \Rightarrow$

$$\sum_{i=1}^{n} C(i) = \sum_{i=1}^{n} A(i) - P(n) + P(0) \le \sum_{i=1}^{n} 2 - b_n + b_0$$
$$= 2n - b_n + b_0$$

  - Como $b_0 \le k$ (número de bits del contador), si se ejecutan al menos $n = \Omega(k)$ incrementos, el coste real total es $O(n)$.

# Resumiendo

- Método agregado: cálculo exacto
- Método contable: inventar un A(i) sencillo para que cuadren las cuentas
- Método potencial: inventar un P(i) sencillo para que cuadren las cuentas

$$\sum_{i=1}^{n} A(i) \geq \sum_{i=1}^{n} C(i)$$

# Contenido

- Introducción: ¿por qué necesitamos otro análisis de complejidad?
- Método agregado
- Método contable
- Potencial
- **Ejemplo: tablas hash**

# Ejemplo: análisis de tablas dinámicas

- Considerar una tabla *hash* con operaciones de creación, búsqueda e inserción (y más adelante borrado) y con espacio limitado.
- Si la tabla está llena, la inserción obliga a *expandirla* a otra con, por ejemplo, el doble de capacidad y copiar todos los datos a la nueva.
- Se trata de analizar el coste de las inserciones, teniendo en cuenta que pueden traer consigo la expansión de la tabla.

# Ejemplo: análisis de tablas dinámicas

```
algoritmo insertar(T,x)
principio
  si T.capacidad=0 entonces
    crearTabla(T,1);
    T.capacidad:=1
  fsi;
  si T.numdatos=T.capacidad ent
    crearTabla(nuevaT,2*T.capacidad);
    insertarTabla(T,nuevaT);
    liberar(T);
    T:=nuevaT;
    T.capacidad:=2*T.capacidad;
  fsi;
  insertarDato(T,x);
  T.numdatos:=numdatos+1
fin
```

capacidad máxima

crear con capacidad 1

número de datos actualmente almacenados

expansión al doble de capacidad

volcar la tabla vieja a la nueva

Definimos la "unidad de coste" como el coste de esta operación de *inserción elemental*

# Ejemplo: análisis de tablas dinámicas

- Analicemos la secuencia de $n$ inserciones en una tabla inicialmente vacía.
  - Coste de la operación $i$-ésima: $C(i)$
    - Si hay espacio: $C(i) = 1$
    - Si hay que expandir: $C(i) = i$
  - Si se consideran $n$ inserciones el coste peor de una inserción es $O(n)$ y, por tanto, el coste peor para toda la secuencia está acotado por $O(n^2)$.
  - Este cálculo es correcto pero muy poco ajustado.

# Ejemplo: análisis de tablas dinámicas

- Análisis mediante el método agregado:
  - $C(i) = i$, si $i - 1$ es una potencia exacta de 2,
    $C(i) = 1$, en otro caso.
  - Por tanto:

$$\sum_{i=1}^{n} C(i) \leq n + \sum_{j=0}^{\lfloor \log n \rfloor} 2^j$$

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1} - 1}{c - 1}, \quad c \neq 1$$

$$< n + 2n$$

$$= 3n$$

  - Es decir, el coste amortizado de cada inserción es 3.

# Ejemplo: análisis de tablas dinámicas

- Análisis mediante el método contable:
  - Nos aporta la intuición de por qué el coste amortizado de una inserción es 3.
  - Cada elemento insertado paga por tres inserciones elementales:
    - Suponer que tenemos una tabla de capacidad *m* justo tras una expansión, y sin ningún crédito disponible.
    - Entonces el número de elementos en la tabla es *m*/2.
    - Por cada inserción posterior pagamos 3 unidades
      - Una de ellas es el coste de la inserción elemental del dato.
      - Otra queda como crédito, asociada al dato insertado.
      - La otra queda como crédito de uno de los restantes *m*/2 datos que ya estaban en la tabla y que no tenían crédito.
    - Tras *m*/2 inserciones, la tabla está llena, con *m* datos, y cada dato tiene crédito de 1 unidad.
    - Con ese crédito se hace la expansión gratis.

# Ejemplo: análisis de tablas dinámicas

- Análisis mediante el método potencial:
  - Primero: definir la función de potencial
    - $P(i)$ = potencial de la tabla $T$ tras la $i$-ésima inserción
    - Que valga 0 tras cada expansión.
    - Que haya aumentado hasta igualar la capacidad de la tabla cuando ésta esté llena (para que la siguiente expansión pueda pagarse con el potencial).
    - Por ejemplo: $P(i) = 2 \cdot numdatos(i) - capacidad(i)$
      - Tras una expansión, $numdatos(i) = capacidad(i)/2$, luego $P(i) = 0$.
      - Inmediatamente antes de una expansión, $numdatos(i) = capacidad(i)$, luego $P(i) = numdatos(i)$.
      - El valor inicial es 0, y como la tabla siempre está medio llena, $numdatos(i) \geq capacidad(i)/2$, luego $P$ es siempre no negativo.

# Ejemplo: análisis de tablas dinámicas

- Segundo: calcular el coste amortizado
  - Inicialmente: $numdatos(i) = 0$, $capacidad(i) = 0$, $P(i) = 0$.
  - Si la $i$-ésima inserción no genera expansión, $capacidad(i) = capacidad(i-1)$ y:

$$
\begin{aligned}
A(i) &= C(i) + P(i) - P(i-1) \\
&= 1 + (2\cdot numdatos(i) - capacidad(i)) - \\
&\qquad - (2\cdot numdatos(i-1) - capacidad(i-1)) \\
&= 1 + (2\cdot numdatos(i) - capacidad(i)) - \\
&\qquad - (2(numdatos(i) - 1) - capacidad(i)) \\
&= 3
\end{aligned}
$$

  - Si la $i$-ésima inserción genera expansión, $capacidad(i)/2 = capacidad(i-1) = numdatos(i-1) = numdatos(i) - 1$, y:

$$
\begin{aligned}
A(i) &= C(i) + P(i) - P(i-1) \\
&= numdatos(i) + (2\cdot numdatos(i) - (2\cdot numdatos(i) - 2)) - \\
&\qquad - (2(numdatos(i) - 1) - (numdatos(i) - 1)) \\
&= 3
\end{aligned}
$$

# Resumen análisis amortizado

- Tenemos n operaciones que pueden ocurrir seguidas con costes en tiempo

$$C(1), \ldots C(i), \ldots C(n)$$

- Buscamos coste amortizado $A(i) \geq 0$ que tiene que cumplir

$$\sum A(i) \geq \sum C(i)$$

# Resumen análisis amortizado

$$\sum A(i) \geq \sum C(i)$$

- **Método agregado:** calcula o estima $\sum C(i)$
- **Método contable:** define A(i) que consiga un balance positivo P(i)

  $P(i) = P(i-1) + A(i) - C(i)$

  C(i) es el coste real

  A(i) es lo que pago

  P(i) es lo que me queda (siempre $\geq 0$)

# Resumen análisis amortizado

$$\sum A(i) \geq \sum C(i)$$

- Método potencial: define $P(i) \geq 0$

  $$A(i) = C(i) + P(i) - P(i-1)$$

  - La diferencia $P(i)-P(i-1)$ debe compensar el coste $C(i)$
  - Ejemplo: $P(i)$= altura de la pila

# Ampliando el ejemplo

- **Tabla con operación de borrado**:
  - Si al borrar un elemento la tabla queda "muy vacía", se puede *contraer* la tabla.
  - Primera estrategia posible: "muy vacía" = la tabla tiene menos de la mitad de posiciones ocupadas.
    - Se garantiza que el factor de carga sea siempre superior a ½.
    - El coste amortizado puede ser demasiado grande. Ejemplo:
      - Hacemos $n$ operaciones (con $n$ una potencia de 2).
      - Las primeras $n/2$ son inserciones, con un coste amortizado total de $O(n)$.
      - Al acabar las inserciones, $numdatos(n) = capacidad(n) = n/2$.
      - Las siguientes $n/2$ operaciones son: I, B, B, I, I, B, B, I, I, … (con I = inserción y B = borrado).

# Ampliando el ejemplo

- La primera inserción provoca una expansión a tamaño $n$.
- Los dos borrados siguientes provocan una contracción a tamaño $n/2$.
- Las dos inserciones siguientes provocan una nueva expansión a tamaño $n$, y así sucesivamente.
- De esta forma, el coste de toda la secuencia es $\Theta(n^2)$, y por tanto el coste amortizado de cada operación es $\Theta(n)$.

- El problema de esa estrategia: después de una expansión no se realizan suficientes borrados para justificar el pago de una contracción, y viceversa, después de una contracción no se hacen suficientes inserciones para pagar una expansión.

## Ampliando el ejemplo

- Nueva estrategia (para los borrados):
  - Duplicar la capacidad de la tabla cuando hay que insertar en una tabla llena, pero **contraer la tabla a la mitad cuando un borrado hace que quede llena en menos de ¼ de su capacidad**.
    - De esta forma, tras una expansión el factor de carga es ½ y por tanto la mitad de los elementos deben ser borrados para que ocurra una contracción.
    - Tras una contracción el factor de carga es ½ y, por tanto, el número de elementos de la tabla debe duplicarse hasta provocar una expansión.

# Ampliando el ejemplo

- Análisis de *n* operaciones de inserción y/o borrado mediante el método potencial:
  - Primero: definir el potencial, función *P*, que
    - sea 0 justo tras una expansión o contracción y
    - crezca mientras el factor de carga, $\alpha(i) = \mathit{numdatos}(i)/\mathit{capacidad}(i)$, crece hacia 1 o disminuye hacia ¼.
    - Como en una tabla vacía $\mathit{numdatos} = \mathit{capacidad} = 0$ y $\alpha = 1$, siempre se tiene que $\mathit{numdatos}(i) = \alpha(i) \cdot \mathit{capacidad}(i)$.
    - Por ejemplo:

$$P(i) = \begin{cases} 2\mathit{numdatos}(i) - \mathit{capacidad}(i), & \text{si } \alpha(i) \geq 1/2 \\ \mathit{capacidad}(i)/2 - \mathit{numdatos}(i), & \text{si } \alpha(i) < 1/2 \end{cases}$$

así, el potencial nunca es negativo y el de la tabla vacía es 0.

# Ampliando el ejemplo

- Propiedades de esta función

$$P(i) = \begin{cases} 2numdatos(i) - capacidad(i), & \text{si } \alpha(i) \geq 1/2 \\ capacidad(i)/2 - numdatos(i), & \text{si } \alpha(i) < 1/2 \end{cases}$$

  – Cuando $\alpha(i) = ½$, el potencial es 0.

  – Cuando $\alpha(i) = 1$, se tiene que *numdatos*($i$) = *capacidad*($i$), y por tanto $P(i)$ = *numdatos*($i$), es decir, el potencial permite pagar por una expansión si se inserta un nuevo dato.

  – Cuando $\alpha(i) = ¼$, *capacidad*($i$) = 4·*numdatos*($i$), y por tanto $P(i)$ = *numdatos*($i$), es decir, el potencial permite pagar por una contracción si se borra un dato.

# Ampliando el ejemplo

– Segundo: cálculo del coste amortizado

- Inicialmente, *numdatos*(0)=*capacidad*(0)=$P$(0)=0 y $\alpha$(0)=1.
- Si la $i$-ésima operación es una inserción:
  - Si $\alpha(i–1) \geq ½$, el análisis es idéntico al caso anterior (con sólo operaciones de inserción), y el coste amortizado de la operación es menor o igual a 3.
  - Si $\alpha(i–1) < ½$, la tabla no precisa expandirse, y hay dos casos:
    - » Si $\alpha(i) < ½$:

$$
\begin{aligned}
A(i) &= C(i) + P(i) - P(i-1) \\
&= 1 + (\mathit{capacidad}(i)/2 - \mathit{numdatos}(i)) - \\
&\qquad - (\mathit{capacidad}(i-1)/2 - \mathit{numdatos}(i-1)) \\
&= 1 + (\mathit{capacidad}(i)/2 - \mathit{numdatos}(i)) - \\
&\qquad - (\mathit{capacidad}(i)/2 - \mathit{numdatos}(i) - 1) \\
&= 0
\end{aligned}
$$

# Ampliando el ejemplo

» Si $\alpha(i-1) < ½$ pero $\alpha(i) \geq ½$:

$$
\begin{aligned}
A(i) &= C(i) + P(i) - P(i-1) \\
&= 1 + (2\cdot numdatos(i) - capacidad(i)) - \\
&\qquad - (capacidad(i-1)/2 - numdatos(i-1)) \\
&= 1 + (2(numdatos(i-1) + 1) - capacidad(i-1)) - \\
&\qquad - (capacidad(i-1)/2 - numdatos(i-1)) \\
&= 3\cdot numdatos(i-1) - 3/2\cdot capacidad(i-1) + 3 \\
&= 3\cdot \alpha(i-1) capacidad(i-1) - 3/2\cdot capacidad(i-1) + 3 \\
&< 3/2\cdot capacidad(i-1) - 3/2\cdot capacidad(i-1) + 3 \\
&= 3
\end{aligned}
$$

- Por tanto, el coste amortizado de una inserción es como mucho 3.

# Ampliando el ejemplo

- Si la *i*-ésima operación es un borrado:
  - En este caso $numdatos(i) = numdatos(i–1) – 1$
  - Si $\alpha(i–1) < ½$ y la operación no provoca una contracción, entonces $capacidad(i) = capacidad(i–1)$ y el coste es:

$$
\begin{aligned}
A(i) &= C(i) + P(i) - P(i-1) \\
&= 1 + (capacidad(i)/2 - numdatos(i)) - \\
&\qquad - (capacidad(i-1)/2 - numdatos(i-1)) \\
&= 1 + (capacidad(i)/2 - numdatos(i)) - \\
&\qquad - (capacidad(i)/2 - (numdatos(i) + 1)) \\
&= 2
\end{aligned}
$$

# Ampliando el ejemplo

– Si $\alpha(i-1) < ½$ y la operación provoca una contracción:

$C(i) = numdatos(i) + 1$, porque se borra en dato y se mueven $numdatos(i)$

$capacidad(i)/2 = capacidad(i-1)/4 = numdatos(i) + 1$

$$
\begin{aligned}
A(i) &= C(i) + P(i) - P(i-1) \\
&= (numdatos(i) + 1) + (capacidad(i)/2 - numdatos(i)) - \\
&\qquad - (capacidad(i-1)/2 - numdatos(i-1)) \\
&= (numdatos(i) + 1) + ((numdatos(i) + 1) - numdatos(i)) - \\
&\qquad - ((2\cdot numdatos(i) + 2) - (numdatos(i) + 1)) \\
&= 1
\end{aligned}
$$

– Si $\alpha(i-1) \geq ½$ el coste amortizado también se puede acotar con una constante (**ejercicio**).

- En resumen, el coste total de las $n$ operaciones es $O(n)$.

# Resumiendo

- El análisis amortizado me permite analizar el coste de una secuencia de n operaciones y acercarme al coste real
- Hay 3 métodos de cálculo: agregado, contable y potencial
- El potencial requiere dar con la función de potencial adecuada y es el que resuelve los ejemplos más complicados

- Más ejemplos interesantes en la web de la asignatura

# Estructuras de datos (ED) avanzadas

- **Análisis amortizado de ED**
  - **conceptos básicos** **← este curso sólo hemos visto esto**
  - conjuntos disjuntos
  - listas auto-organizativas
  - árboles *splay*