CI2613: Algoritmos y Estructuras III

Blai Bonet

Universidad Simón Bolívar, Caracas, Venezuela

Enero-Marzo 2015

Análisis amortizado

Análisis amortizado

Análisis amortizado es una técnica utilizada para el análisis de algoritmos

Se utiliza para estimar el **costo amortizado** (o promedio) por operación en una secuencia de operaciones

Es una técnica de análisis de costos en el **peor caso**

Análisis amortizado

Durante el análisis de la ejecución de un algoritmo, frecuentemente identificamos secuencias de operaciones ejecutadas que son críticas para el desempeño del algoritmo

Dada una secuencia $\sigma = \langle op_1, op_2, \ldots, op_n \rangle$ de operaciones, el costo total de la secuencia es:

$$\text{costo}(\sigma) = \sum_{i=1}^{n} \text{costo}(op_i)$$

En un ańalisis de peor caso, nos interesa acotar costo(σ) para cualquier secuencia posible σ

Si C es una cota superior para costo(σ) para cualquier secuencia posible σ, el **costo amortizado** (o promedio) por operación es $C/|\sigma|$

Caso de estudio: Arreglos dinámicos

Una de las estructuras de datos más utilizadas corresponde a los **arreglos dinámicos**

Un arreglo dinámico es un arreglo que **crece de tamaño** a medida que se insertan elementos

Al implementar la ED, siempre se recomienda **duplicar el tamaño** cada vez que se necesite **crecer el arreglo**

Arreglos dinámicos: Pseudocódigo

```
template<T> struct ArregloDinamico {

    T array_[]
    int capacity_
    int size_

    ArregloDinamico()
        array_ = new T[1]
        capacity_ = 1
        size = 0

    ...
    T operator[](int index)
        return array_[index]

    ...
    void push-back(T item)
    T pop-back()
}
```


Arreglos dinámicos: Pseudocódigo

```
void push-back(T item)
    if size_ == capacity
        % arreglo está lleno, redimensionarlo
        capacity_ = 2 * capacity_
        new_array = new T[capacity_]
        for i = 0 to size_ - 1
            new_array[i] = array_[i]
        delete array_
        array_ = new_array

    array_[size_] = item
    size_ = size_ + 1


T pop-back()
    T last_item = array_[size_ - 1]
    size_ = size_ - 1
    return last_item
```


Arreglos dinámicos: Ejemplo

Secuencia de push-backs: 5, 7, 8, 1, 4, 6, 1, 3, 6, 9, 2, 1

5	7	8	1	4	6	1	3	6	9	2	1				

Uso de arreglos dinámicos

Considere la ejecución de un algoritmo que utiliza la ED de arreglos dinámicos

Suponga que el algoritmo ejecuta n operaciones sobre un arreglo dado:

$$\sigma = \langle op_1, op_2, \ldots, op_n \rangle$$

No conocemos las operaciones; sólo sabemos que son sobre el mismo arreglo dinámico

¿Podemos acotar el tiempo de ejecución costo(σ) de σ?

Arreglos dinámicos: Primer análisis

Secuencia $\sigma = \langle op_1, op_2, \ldots, op_n \rangle$

Observación:

– Si el arreglo contiene m elementos, `push-back` toma tiempo $O(m)$

– Todas las otras operaciones toman **tiempo constante** $O(1)$

Arreglos dinámicos: Primer análisis

Secuencia $\sigma = \langle op_1, op_2, \ldots, op_n \rangle$

Análisis:

1. Después de $O(n)$ operaciones, el arreglo puede contener $O(n)$ elementos
2. Por lo tanto, un `push-back` puede costar $O(n)$ unidades de tiempo
3. Si existen $O(n)$ `push-back`, incurrimos en $O(n^2)$ unidades de tiempo

Entonces, costo(σ) $= O(n + n^2) = O(n^2)$ **(muy impreciso)**

El costo amortizado por operación es $O(n)$ **(muy impreciso)**

Arreglos dinámicos: Segundo análisis

Secuencia $\sigma = \langle op_1, op_2, \ldots, op_n \rangle$

Observación:

– Como el arreglo duplica su tamaño cada vez que crece, no pueden existir $O(n)$ operaciones `push-back` que "crecen" al arreglo

– Existen a lo sumo $O(\log n)$ operaciones que crecen al arreglo

Arreglos dinámicos: Segundo análisis

Secuencia $\sigma = \langle op_1, op_2, \ldots, op_n \rangle$

Análisis:

1. Después de $O(n)$ operaciones, el arreglo puede contener $O(n)$ elementos
2. A lo sumo $O(\log n)$ operaciones `push-back` que crecen el arreglo; cada una cuesta $O(n)$ unidades de tiempo
3. Las otras operaciones toman tiempo $O(1)$

Entonces, costo$(\sigma) = O(n + n \log n) = O(n \log n)$ **(impreciso)**

El costo amortizado por operación es $O(\log n)$ **(impreciso)**

Arreglos dinámicos

Ambos análisis son correctos pero imprecisos

Utilizando las técnicas de análisis amortizado se puede mostrar:

– El costo total de la secuencia es $O(n)$ unidades de tiempo

– El costo amortizado por operación es $O(1)$

Métodos de análisis amortizado

Existen tres métodos principales de análisis amortizado:

– Método de análisis agregado

– Método de "contabilidad" o "prepago"

– Método de la función de potencial (fuera del alcance de CI2613)

Arreglos dinámicos: Análisis agregado

Secuencia $\sigma = \langle op_1, op_2, \ldots, op_n \rangle$

Estudiamos las operaciones en detalle y **agregamos su tiempo de ejecución de forma global**

– Existen $O(\log n)$ operaciones `push-back` que crecen el arreglo; llamemos a dichas operaciones "`push-back` malos"

– El arreglo comienza de tamaño 1 y va doblando su tamaño

1er `push-back` malo toma 1 unidad de tiempo

2do `push-back` malo toma 2 unidades de tiempo

3er `push-back` malo toma 4 unidades de tiempo

...

i-ésimo `push-back` malo toma 2^{i-1} unidades de tiempo

Arreglos dinámicos: Análisis agregado

Secuencia $\sigma = \langle op_1, op_2, \ldots, op_n \rangle$

– Todas las operaciones excepto `push-back` malos toman tiempo $O(n)$

– El tiempo agregado para los `push-back` malos es:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^{i-1} \;=\; \sum_{i=0}^{\lceil \log n \rceil - 1} 2^i \;=\; 2^{\lceil \log n \rceil} - 1 \;\leq\; 2^{1+\log n} \;=\; 2n$$

El costo máximo para $\sigma = O(n)$ **(preciso)**

El costo amortizado por operación $= O(1)$ **(preciso)**

Método prepago o contabilidad

Secuencia $\sigma = \langle op_1, op_2, \ldots, op_n \rangle$

La idea es “cobrar” más a las operaciones sencillas de manera que “paguen por adelantado” el costo de las operaciones complejas

Si definimos costos amortizados $\widehat{\text{costo}}(op_i)$ por operación, debemos garantizar:

$$\widehat{costo}(\sigma) \;=\; \sum_{i=1}^{n} \widehat{\text{costo}}(op_i) \;\geq\; \sum_{i=1}^{n} \text{costo}(op_i)$$

Arreglos dinámicos: Método prepago

Secuencia de push-backs: 5, 7, 8, 1, 4, 6, 1, 3, 6, 9, 2, 1

5	7	8	1	4	6	1	3	6	9	2	1				

$\widehat{\text{Costo}}$: 1+2

Crédito: 9

Costo amortizado por push-back: 3 unidades $= O(1)$

Costo amortizado de otras operaciones: 1 unidad $= O(1)$

Costo máximo para $\sigma = O(n)$ **(preciso)**

Costo amortizado por operación $= O(1)$ **(preciso)**

