Cl2613: Algoritmos y Estructuras III

Blai Bonet

Universidad Simón Bolívar, Caracas, Venezuela

Enero-Marzo 2015

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

© 2014 Blai Bonet

Análisis amortizado

CI2613

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

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

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

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

© 2014 Blai Bonet C|2613 © 2014 Blai Bonet C|2613

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

© 2014 Blai Bonet

© 2014 Blai Bonet

CI2613

CI2613

© 2014 Blai Bonet

Arreglos dinámicos: Pseudocódigo

```
void push-back(T item)
       if size == capacity
           % arreglo está lleno, redimensionarlo
3
           capacity = 2 * capacity
4
           new array = new T[capacity ]
           for i = 0 to size - 1
6
               new array[i] = array [i]
7
           delete array
8
9
           array = new array
10
       array [size ] = item
11
12
       size = size + 1
13
14
   T pop-back()
       T last item = array [size - 1]
16
       size = size - 1
17
       return last item
18
```

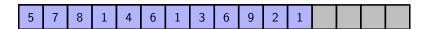
Arreglos dinámicos: Pseudocódigo

```
1 template<T> struct ArregloDinamico {
3
        T array []
       int capacity
4
       int size
6
        ArregloDinamico()
7
            array = new T[1]
            capacity = 1
9
            size = 0
10
11
12
        T operator[](int index)
13
14
            return array [index]
15
16
        void push-back(T item)
17
        T pop-back()
18
19
```

CI2613

Arreglos dinámicos: Ejemplo

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



© 2014 Blai Bonet

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:

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

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

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

© 2014 Blai Bonet

CI2613

CI2613

Arreglos dinámicos: Primer análisis

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

Análisis:

- $\textbf{1} \ \, \text{Después de } O(n) \ \, \text{operaciones, el arreglo puede contener} \ \, O(n) \\ \, \text{elementos}$
- 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, $costo(\sigma) = O(n + n^2) = O(n^2)$ (muy impreciso)

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

Arreglos dinámicos: Primer análisis

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

Observación:

- Si el arreglo contiene m elementos, push-back toma tiempo O(m)
- Todas las otras operaciones toman **tiempo constante** O(1)

© 2014 Blai Bonet CI2613

Arreglos dinámicos: Segundo análisis

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

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

Análisis:

- $\textbf{ 1} \ \, \text{Después de } O(n) \ \, \text{operaciones, el arreglo puede contener} \ \, O(n) \\ \, \text{elementos}$
- 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)

© 2014 Blai Bonet Cl2613

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 Cl2613)

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)

© 2014 Blai Bonet CI2613

Arreglos dinámicos: Análisis agregado

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

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

- Todas las operaciones excepto push-back malos toman tiempo O(n)
- El tiempo agregado para los push-back malos es:

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

El costo máximo para $\sigma = O(n)$

(preciso)

El costo amortizado por operación = O(1)

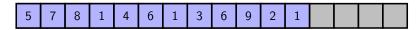
(preciso)

© 2014 Blai Bonet

CI2613

Arreglos dinámicos: Método prepago

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



Costo: 1+

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)

© 2014 Blai Bonet CI2613

Método prepago o contabilidad

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

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 $\widehat{\operatorname{costo}}(op_i)$ por operación, debemos garantizar:

$$\widehat{costo}(\sigma) = \sum_{i=1}^{n} \widehat{costo}(op_i) \ge \sum_{i=1}^{n} \operatorname{costo}(op_i)$$

© 2014 Blai Bonet CI2613