AMORTIZED ANALYSIS

Harbin Institute of Technology, Shenzhen

OUTLINE

基本思想

在摊还分析中，执行一系列数据结构操作所需要时间，是通过对执行的所有操作求平均而得出的

基本思想

对一个数据结构要执行一系列操作：

- 有的代价很高
- 有的代价一般
- 有的代价很低

将总的代价摊还到每个操作上

平摊代价

不涉及概率，
不同于平均情况分析

OUTLINE

- **7.1 摊还分析原理**
- **7.2 聚集方法**
- **7.3 会计方法**
- **7.4 势能方法**
- **7.5 动态表操作的摊还分析**

聚集分析法-原理

操作序列中的每个操作被赋予相同的代价，不管操作的类型

对数据结构共有n个操作
最坏情况下：
操作1:　t_1
操作2:　t_2
:
:
:
操作n:　t_n

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

摊还代价：
$T(n)/n$

会计方法-基本原理

- 一个操作序列中有不同类型的操作，不同类型的操作代价各不相同
  - 为每种操作分配不同的摊还代价
  - 摊还代价可能比实际代价大，也可能比实际代价小
- 操作被执行时，支付摊还代价
  - 如果摊还代价比实际代价高：摊还代价的一部分用于支付实际代价，多余部分作为存款附加在数据结构的**具体数据对象**上
  - 如果摊还代价比实际代价低：摊还代价及数据对象上的存款用来支付实际代价
- 摊还代价的总和与实际代价的总和的关系
  - 只要我们能保证：在任何操作序列上，**存款的总额非负**，则所有操作摊还代价的总和就是实际代价总和的上界（摊还代价大于等于实际代价）

会计方法–基本原理

于是：我们在各种操作上定义平摊代价使得任意操作序列上存款总量是**非负的**，将操作序列上平摊代价求和即可得到这个操作序列的复杂度上界

势能分析——基本原理

- 在会计方法中，如果操作的摊还代价比实际代价大，我们将余额与具体的**数据对象**关联
- 如果我们将这些余额都与**整个数据结构**关联，所有的这样的余额之和，构成——数据结构的**势能**
- 如果操作的摊还代价**大于**操作的实际代价-**势能增加**
- 如果操作的摊还代价**小于**操作的实际代价，要用数据结构的势能来支付实际代价-**势能减少**

势能分析——基本原理

- 分配一个数组空间，长度为m，连续m次插入元素以后，需要对数据空间进行扩张，每次扩张到2m大小
- 定义一个势函数$\boldsymbol{\phi}$，在<u>扩张操作之后值为0</u>，而<u>表满时值为表的规模</u>，如此可以用势能支付下一次扩张的代价

势能：$\boldsymbol{\phi}$(T) = 2*T.num (表中数据量) - T.size (表的规模)

- 扩张后 T.num = T.size/2，此时$\boldsymbol{\phi}$(T)=0
- 扩张前 T.num = T.size，则$\boldsymbol{\phi}$(T) = T.num

势能分析—基本原理

势能的定义：对一个初始数据结构D_0执行n个操作，对操作i:

- <u>实际代价</u>c_i将数据结构D_{i-1}变为D_i
- <u>势能函数</u>ϕ将每个数据结构D_i映射为一个实数$\phi(D_i)$
- $\phi(D_i)$ 就是关联到数据结构D_i的势能
- 摊还代价c'_i定义为：$c'_i = c_i + \phi(D_i) - \phi(D_{i-1})$
- 势能函数ϕ是非递减的

势能分析—基本原理

- $\phi(D_i)$ 就是关联到数据结构D_i的势能

  $\boldsymbol{\phi}$(T) = 2*T.num (表中数据量) - T.size (表的规模)

- 摊还代价c'_i定义为：$c'_i = c_i + \phi(D_i) - \phi(D_{i-1})$
- 针对第 i 次插入，且**不扩张**的情况，$size_{i-1} = size_i$：

$$\boldsymbol{c'_i = ci + \phi(D_i) - \phi(D_{i-1})}$$

$$\mathbf{= 1+(2*num_{\mathit{i}}-size_{\mathit{i}}) - (2*num_{\mathit{i}-1}-size_{\mathit{i}-1})}$$

$$\mathbf{= 1+(2*num_{\mathit{i}}-size_{\mathit{i}}) - (2*(num_{\mathit{i}}-1)-size_{\mathit{i}})}$$

$$\mathbf{= 3}$$

势能分析—基本原理

- $\phi(D_i)$ 就是关联到数据结构D_i的势能

  $\boldsymbol{\phi}(\mathrm{T}) = 2*\mathrm{T.num}$ (表中数据量) - T.size (表的规模)

- 摊还代价c'_i定义为：$c'_i = c_i + \phi(D_i) - \phi(D_{i-1})$
- 针对第 i 次插入，且**扩张**的情况，$size_i = 2* size_{i-1}$，

$size_{i-1} = num_{i-1} = num_i - 1$ ， $\leftrightarrow$ $size_i = 2* (num_i - 1)$ ：

$$\boldsymbol{c'_i = ci + \phi(D_i) - \phi(D_{i-1})}$$

$$\mathbf{= num_i + (2*num_i - size_i) - (2*num_{i-1} - size_{i-1})}$$

$$\mathbf{= num_i + (2*num_i - 2* (num_i - 1)) - (2*(num_i - 1) - (num_i - 1))}$$

$$\mathbf{= 3}$$

势能分析—基本原理

- n个操作的总的摊还代价为：

$$\sum_{i=1}^{n} c_i' = \sum_{i=1}^{n}\left(c_i + \phi(D_i) - \phi(D_{i-1})\right)$$

$$= \sum_{i=1}^{n} c_i + \left(\phi(D_n) - \phi(D_0)\right)$$

摊还代价依赖于所选择的势函数ϕ。不同的势函数可能会产生不同的摊还代价，但它们都是实际代价的上界

在实践中，我们定义$\phi(D_0)$为0，然后再证明对所有i有$\phi(D_i) \geq 0$

于是势函数ϕ满足$\phi(D_n) \geq \phi(D_0)$，则总的摊还代价就是总的实际代价的一个上界

动态表—表的扩张

- 向表中插入一个数组元素时，分配一个包含比原表更多的槽的新表，再将原表中的各项复制到新表中去
- 一种常用的启发式技术是分配一个比原表大一倍的新表，如果只对表执行插入操作，则表的装载因子总是至少为$\frac{1}{2}$，这样浪费掉的空间就始终不会超过表总空间的一半

动态表—表的扩张

初始为空的表上n次TABLE-INSERT操作的代价分析-**会计法分析**

插入1（注:第一次摊还代价为2，其余为3）

	1
存款	1

扩张

	1	
存款	0	

插入2

	1	2
存款	1	1

扩张

	1	2		
存款	0	0		

插入3

	1	2	3	
存款	1	0	1	

每次执行TABLE—INSERT摊还代价为3:

- **1支付第i步中的基本插入操作的实际代价**
- **1作为自身的存款**
- **1存入表中第一个没有存款的数据上**
- **当发生表的扩张时，数据的复制的代价由数据上的存款来支付**
- **初始为空的表上n次TABLE-INSERT操作的摊还代价总和为$3n$**

动态表—表的扩张

初始为空的表上n次TABLE-INSERT操作的代价分析-**会计法分析**

i	1	2	3	4
存款	1	1	1	1

插入4

i	1	2	3	4				
存款	0	0	0	0				

扩张

i	1	2	3	4	5			
存款	1	0	0	0	1			

插入5

i	1	2	3	4	5	6		
存款	1	1	0	0	1	1		

插入6

依此类推

动态表—表的扩张

初始为空的表上n次TABLE-INSERT操作的代价分析-**势能法分析**

- 势能函数怎么定义，才能使得表满发生扩张时势能能支付扩张的代价
- 如果势能函数满足：
  - 刚扩充完（**还未插入**），$\phi(T) = 0$；
  - 表满时$\phi(T) = size(T)$，是否可行**?**

势能函数可行并且：

- $\boldsymbol{num[T] \geq size[T]/2}$，$\boldsymbol{\phi(T) \geq 0}$
- **n次TABLE-INSERT操作的总的摊还代价就是总的实际代价的一个上界**

势能分析—基本原理

- 分配一个数组空间，长度为m，连续m次插入元素以后，需要对数据空间进行扩张，每次扩张到2m大小
- 定义一个势函数ϕ，在扩张操作之后值为0，而表满时值为表的规模，如此可以用**势能**支付下一次扩张的代价

  势能：ϕ(T) = 2*T.num (表中数据量) - T.size (表的规模)
- 扩张后 T.num = T.size/2，此时ϕ(T)=0
- 扩张前 T.num = T.size，则ϕ(T) = T.num

势能分析—基本原理

- $\phi(D_i)$ 就是关联到数据结构D_i的势能

  $\boldsymbol{\phi}$(T) = 2*T.num (表中数据量) - T.size (表的规模)

- 摊还代价c'_i定义为：$c'_i = c_i + \phi(D_i) - \phi(D_{i-1})$
- 针对第 i 次插入，且**不扩张**的情况，$size_{i-1} = size_i$：

$$\boldsymbol{c'_i = ci + \phi(D_i) - \phi(D_{i-1})}$$

$$\mathbf{= 1+(2*T.num_{\mathit{i}}-size_{\mathit{i}}) - (2*T.num_{\mathit{i}-1}-size_{\mathit{i}-1})}$$

$$\mathbf{= 1+(2*T.num_{\mathit{i}}-size_{\mathit{i}}) - (2*(T.num_{\mathit{i}}-1)-size_{i})}$$

$$\mathbf{= 3}$$

势能分析—基本原理

- $\phi(D_i)$ 就是关联到数据结构D_i的势能

  $\boldsymbol{\phi}$(T) = 2*T.num (表中数据量) - T.size (表的规模)

- 摊还代价c'_i定义为：$c'_i = c_i + \phi(D_i) - \phi(D_{i-1})$
- 针对第 *i* 次插入，且**扩张**的情况，$\text{size}_i = 2^*\ \text{size}_{i\text{-}1}$，

$\text{size}_{i\text{-}1} = \text{num}_{i\text{-}1} = \text{num}_i\text{-}1$ ，$\leftrightarrow$ $\text{size}_i = 2^*\ (\text{num}_i\ \text{-}1)$ ：

$$\boldsymbol{c'_i = ci + \phi(D_i) - \phi(D_{i-1})}$$

$$\mathbf{= num_{\mathit{i}} + (2^*num_{\mathit{i}}\text{-}size_{\mathit{i}}) - (2^*num_{\mathit{i}\text{-}1} - size_{\mathit{i}\text{-}1})}$$

$$\mathbf{= num_{\mathit{i}} + (2^*num_{\mathit{i}}\text{-}2^* (num_{\mathit{i}} \text{-}1)) - (2^*(num_{\mathit{i}}\text{-}1) - (num_{\mathit{i}}\text{-}1))}$$

$$\mathbf{= 3}$$