湖南大學
HUNAN UNIVERSITY

第三章 摊还分析

湖南大学信息科学与工程学院

期末试卷结构

一、判断题（24分）

二、计算题（20分）递归式计算，主要应用主定理、递归树等知识点

三、算法题1（16分）

四、算法题2（15分）

五、算法题3（15分）

小班讨论题

六、算法题4（10分），教材三章（分治、贪心、动态规划）最后思考题

湖南大学 HUNAN UNIVERSITY

目录

建立一个哈希表

一个哈希表多大才合适？

- 假设要建立一个简单哈希表，用链接法解决冲突，哈希表的大小是多少才合适？
  - 元素数量的两倍大
- 增大哈希表时，搜索时间会怎样变化呢？
  - 通常时间会减小。如果哈希表足够大，那么就会得到一个直接映射表，最坏情况下，时间为O(1)

建立一个哈希表

但哈希表又应尽可能小，那么当插入元素个数n未知时，应该怎样建立哈希表呢？

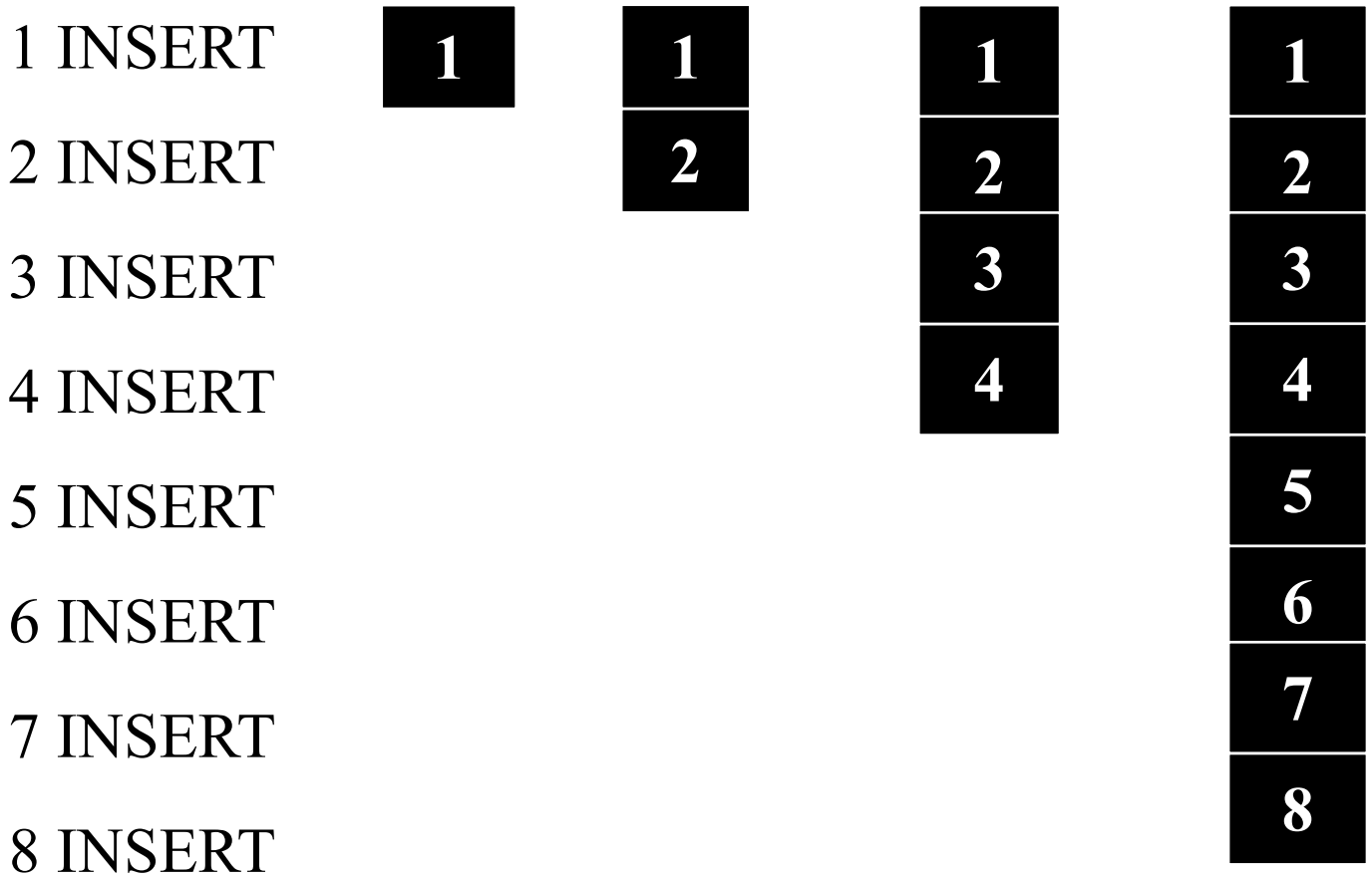
使用动态表。当表满时插入元素会产生溢出，此时：

①用malloc函数分配一个更大的表

②将旧表里的元素复制到新表

③释放旧表空间

建立一个哈希表


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

代价计算

有一个序列包括n个插入运算，在最坏情况下一次插入操作的代价为多少

O(n)

n个项的插入操作代价仍为O(n)，分析如下：

设c_i为第i个插入的代价，则

$$c_i = \begin{cases} i & \text{若} i-1\text{恰为2的幂} \\ 1 & \text{其他} \end{cases}$$

代价计算

i	1	2	3	4	5	6	7	8	9	10
$SIZE_i$	1	2	4	4	8	8	8	8	16	16
C_i	1	2	3	1	5	1	1	1	9	1

因此，n个插入运算的总代价为$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lfloor lgn \rfloor} 2^j < n + 2n = 3n$

每个插入运算的平均代价为O(n)/n，也就是O(1)。

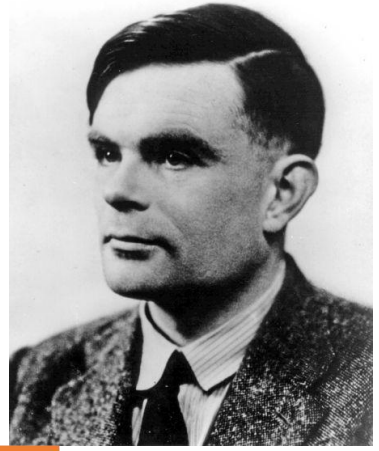
这就是摊还分析的思想，这里采用了聚合分析方法

湖南大学 HUNAN UNIVERSITY

目录

什么是摊还分析？

- 功过相抵
  - 中国：某大臣因办某件事情失败，被皇帝怪罪，看在曾经立下汗马功劳等，希望能够从轻发落。
  - 西方：英国政府曾打算对艾伦·图灵（“人工智能之父”）进行严惩，因为他被发现是一个同性恋者，但图灵破译了德国海军密码“谜”，在第二次世界大战中的胜利立下了汗马功劳，可以抵销罪过。



将一生摊开来看，一次失利并不显得怎样罪大恶极。这就是摊还！

什么是摊还分析？

定义

摊还分析里的这种平均也并不是我们平时所熟知的考虑所有情况的平均，而是在最坏情况下，一系列操作的平均成本！

摊还分析 VS 平均情况分析

例如，如果打一场篮球赛，

1、平均情况分析

结束后一个队的得分就是这个队所有队员的总得分。用这个得分除以队员数就获得每个队员的平均得分。

2、摊还分析

针对的是最坏情况下每个操作的平均成本。但如果一个球队在打球过程中每个队员都表现的是平时最差的水平。

摊还分析（特指最坏情况平均） ≠ 平均情况分析 ！

概率分析 VS 摊还分析

1、概率分析：

考虑算法平均执行时间。

考虑同一算法的所有可能输入情况。

如果使用概率，则称为期望运行时间。

2、摊还分析：

操作序列中的平均最坏情况下操作性能/代价。

针对某一数据结构的操作序列。

不使用概率。

什么是摊还分析？

摊还分析（特指最坏情况平均） ≠ 平均情况分析 ！

摊还分析（运转n次平均）≠ 最坏时间复杂性分析（理论单次）！

摊还分析 ≠ 概率分析

湖南大学 HUNAN UNIVERSITY

目录

摊还分析的三种方法

聚类法、聚合法（aggregate）简单，常用

先求出合计，然后求平均。先求出操作序列里所有n个操作的总代价上界T(n)，每个操作的摊还代价T(n)/n。

记账法、核算法、会计法（accounting）复杂

计算每个操作的摊还成本，然后累加每个个体操作的摊还成本而获得一组操作的总摊还成本。这种摊还分析由于很像会计做账时每笔账务都要清楚记录在案，最后得出一个总报表(总收益或总亏损)。

摊还分析的三种方法

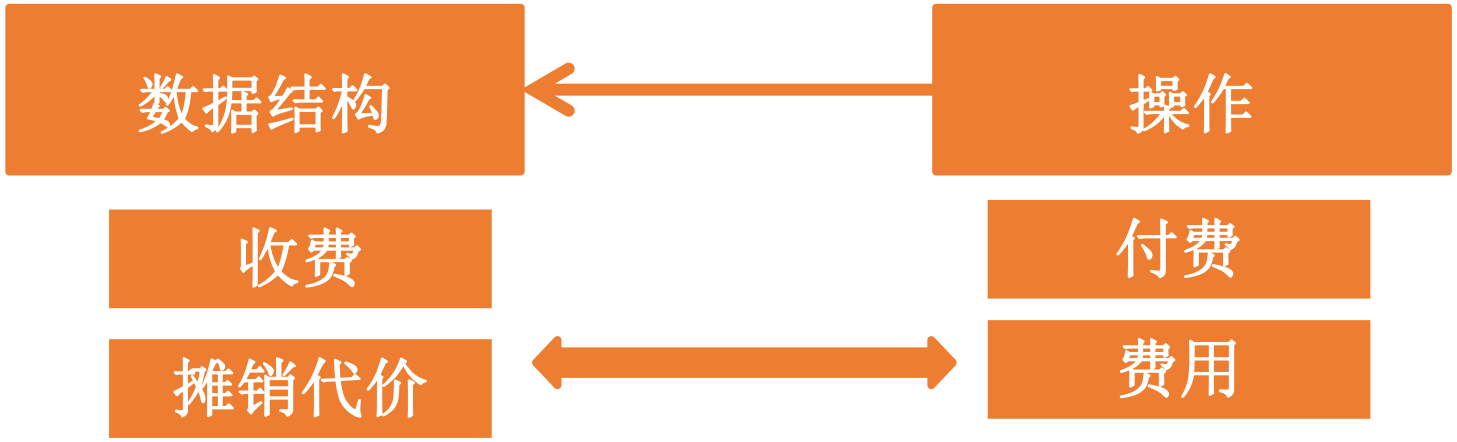
势能法（potential） **可靠、准确**

与核算法类似，也是分析每个操作的代价，但将势能作为一个整体存储，而与数据结构中的某个对象无关。

核算法

核心思想——费用分配（摊还代价）：

为不同的操作分配不同的费用，每一操作分配到的费用称为该操作的摊还代价，它可视作为数据结构对该操作预收的费用（或理解为是该操作对数据结构预付的费用）



核算法

- 超额收费(overcharge)(摊还代价>实际成本)

当一个操作的摊还代价大于其实际成本时，数据结构对该操作预收的费用过多，其超额部分作为存款存储在数据结构的某个特定对象上（称为信用）。

- 收费不足(undercharge)(摊还代价<实际成本)

当一操作的摊还代价小于其实际成本时，数据结构对该操作预收的费用不足，其差额部分可由数据结构的特定对象(是该操作操作的对象)上的存款（信用）支付。

核算法

正确选择各操作的摊还代价（摊还代价的正确性）

要说明每个操作的摊还代价是最坏情况下的平均代价,则必须保证对任意长度n的操作序列，总摊还代价是总的实际代价的一个上界：

$$\forall n, \sum_{i=1}^{n} \hat{C}_i \geq \sum_{i=1}^{n} C_i$$

核算法

回到动态表

假设收取3美元作为对第i步操作的费用

1美元作为插入操作的费用，2美元作为将表翻倍的预存费用

当表的大小增加一倍时，我们就从存款里去取出

1美元来移动新项，1美元用来移动旧项。

核算法

在大小为8的表里，插入前4项后表里已经没有美元剩下了

0	0	0	0	2	2	2	2

此时插入第5项，收取3美元，1美元作为插入的费用，2美元存入

继续插入第6项，第7项，第8项

插入第9项时需建立一个新表，此时移动旧表里的8项将存入的8美元耗尽，所以前8项变为0

0	0	0	0	0	0	0	0								

核算法

继续插入到第16项

0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2

到插入第17项时又需建立一个新表，并将旧表元素复制，所以我们始终可以得到一个永不为负的存款余额。

i	1	2	3	4	5	6	7	8	9	10
$SIZE_i$	1	2	4	4	8	8	8	8	16	16
C_i	1	2	3	1	5	1	1	1	9	1

核算法

BANKi代表存款余额。我们在第一项收取2美元，对其余每项收取3美元。注意，第一项也可以收取3美元，这样就会有多余的1美元贯穿在整个过程中，但这并不影响总代价的上限为3n

i	1	2	3	4	5	6	7	8	9	10
$\hat{C}_i$	2	3	3	3	3	3	3	3	3	3
$BANK_i$	1	2	2	4	2	4	6	8	2	4

i	1	2	3	4	5	6	7	8	9	10
$SIZE_i$	1	2	4	4	8	8	8	8	16	16
C_i	1	2	3	1	5	1	1	1	9	1

核算法

每次插入操作的摊还成本为3，即O(1)。

因此， n 次插入操作的摊还成本最坏为O(n) 。

势能法

- 与核算法的区别

存款——作为势能保存在整个数据结构上，而非其中的特定的对象上。

需要时通过释放能量来支付操作的实际代价

- 势能、势差、势函数、各操作的摊还成本
  - 数据结构：D，初态记为D_0
  - 操作：n个，n可变
  - 第i个操作的结果：$D_{i-1}\rightarrow D_i$，D_{i-1}和D_i表示操作前后D的状态($1\leq i\leq n$)

势能法

势能表示

- 势函数Φ：将每个D_i映射为一个实数$\Phi(D_i)(1\leq i\leq n)$，函数值看作势能
- 每次摊还的实际成本为C_i
- 每次摊还的摊还成本为$\hat{C}_i = C_i + \Phi(D_i) - \Phi(D_{i-1})$
- 操作的摊还代价由两部分构成：实际成本C_i和势差（操作所引起的势能变化）

$$\begin{cases} \Phi(D_i) - \Phi(D_{i-1}) < 0, & \hat{C}_i \text{超额收费，势能} \uparrow \\ \Phi(D_i) - \Phi(D_{i-1}) > 0, & \hat{C}_i \text{收费不足，势能} \downarrow \\ \Phi(D_i) - \Phi(D_{i-1}) = 0, & \hat{C}_i = C_i \end{cases}$$

势能法

- 总的摊还代价及势函数的选择(正确性)

即i个操作后，势差为：序列最终的势能和最初的势能之差。此势差≥0或$\Phi(D_i) \geq \Phi(D_0)$，则总的摊还代价是总的实际成本的一个上界，但须对任意的i成立，故有：

$\forall i \in I$，$\Phi(D_i) \geq \Phi(D_0)$，通常定义$\Phi(D_0)=0$

则有：$\forall i \in I$，$\Phi(D_i) \geq 0$

势能法

因为 $\Phi(Di)$项是交叠的，可以消去，所以得出n个操作的总摊还代价为

$$\sum_{i=1}^{n}\widehat{C}_i=\sum_{i=1}^{n})C_i+\Phi)D_i(-\Phi)D_{i-1}))=\sum_{i=1}^{n}C_i+\Phi)D_n(-\Phi)D_0($$

势能法

在动态表例子中

我们定义一个势能函数 $\Phi_i = 2 \times \mathrm{T.num} - \mathrm{T.size} = 2i - 2^{\lceil log\ i \rceil}$

假设$2^{\lceil log\ 0 \rceil} = 0$

满足$\Phi(D_0)=0$；$\Phi(D_i)\geq 0 \quad \forall i$

势能法

现在有8个槽，6个是满的，此时 i = 6,

$$\Phi_6 = 2 \times 6 - 2^{\lceil log\ 6 \rceil} = 2 \times 6 - 8 = 4$$

0	**0**	**0**	**0**	**0**	**0**		

势能法—平均代价的计算

$$\widehat{\boldsymbol{C}}_{\boldsymbol{i}} = \boldsymbol{C}_{\boldsymbol{i}} + \Phi(\boldsymbol{D}_{\boldsymbol{i}}) - \Phi(D_{\mathrm{i}\ -\ 1})$$

$$= \begin{cases} i & \text{若} i-1 \text{恰为2的幂} \\ 1 & \text{其他} \end{cases} + 2i - 2^{\lceil log\ i \rceil} - (2(i-1) - 2^{\lceil log\ i-1 \rceil})$$

$$= \begin{cases} i & \text{若} i-1 \text{恰为2的幂} \\ 1 & \text{其他} \end{cases} + 2 - 2^{\lceil log\ i \rceil} + 2^{\lceil log\ i-1 \rceil}$$

势能法—平均代价的计算

情形1：$i-1$恰为2的幂时

$$\begin{aligned}\widehat{C}_i &= i+2-2^{\lceil log\, i\rceil}+2^{\lceil log\, i-1\rceil}\\ &= i+2-2\ (i-1)\ +\ (i-1)\\ &= i+2-2i+2+i-1\\ &= 3\end{aligned}$$

势能法—平均代价的计算

情形2：$i-1$不为2的幂时

$$\widehat{C}_i = 1 + 2 - 2^{\lceil log\ i \rceil} + 2^{\lceil log\ i-1 \rceil}$$
$$= 3$$

所以我们得到，每个操作的代价为3，n个插入操作的平摊代价为3n，这也是最坏情况下实际代价的上限。所以n个操作的代价为O(n)。

湖南大學
HUNAN UNIVERSITY

谢谢观赏

实事求是　敢为人先