均摊分析

算分回课

柯宇斌

北京大学信息科学技术学院

2024 年 3 月 29 日

1. 均摊分析的概念
2. 解决方案
3. 均摊分析的应用

1 均摊分析的概念

2 解决方案

3 均摊分析的应用

均摊分析的意义和定义

- 单个操作的最坏情况分析不能反映通常情况的开销
- 解决方法
  - 赋予每个操作虚拟代价
  - 保障任意一组连续合法操作，虚拟代价和不低于实际代价，从而求平均代价
  - 虚拟代价的赋值不唯一

问题 1

栈 S 上有三种操作

- $PUSH(S, x)$ 将 x 压入 S
- $POP(S)$ 弹出栈顶
- $MULTIPOP(S, k)$ 弹出栈顶 k 个对象 (遇到栈空提前返回)

任意 n 个栈操作的最坏时间分析

- $PUSH(S, x)$ $O(1)$
- $POP(S)$ $O(1)$
- $MULTIPOP(S, k)$ $O(n)$

总时间 $O(n^2)$, 单次时间 $O(n)$

问题 2

二进制计数器 (k 个数位)

- *INCREMENT*, 计数器加 1

任意 n 个 *INCREMENT* 操作的最坏时间分析

- *INCREMENT*,$O(k)$

总时间 $O(nk)$, 单次时间 $O(k)$

1 均摊分析的概念

2 解决方案
聚集法
记账法
势能法

3 均摊分析的应用

1 均摊分析的概念

2 解决方案
聚集法
记账法
势能法

3 均摊分析的应用

聚集法

- 栈操作
  - 每个对象只会出入栈一次
  - 入栈个数 $=POP$ 个数
  - 总时间 $O(n)$, 单次时间 $O(1)$
  - 聚集法: 通过总时间求平均得到均摊时间不需要对操作序列的概率分布做假设
- 二进制计数器
  - 第 i 位, 每连续的 2^i 次操作恰翻转一次
  - 总时间 $O(n)$, 单次时间 $O(1)$

聚集法

- 包含复位功能的二进制计数器
  - 额外记录一个最高位 1 的位置
  - *INCREMENT* 可以按 $O(1)$ 处理
  - *SET* 第一个 *SET* 至多移动 k 次
  - *SET* 除了第一个 *SET* 以外，假设某次复位时最高非 0 位是第 m 位，则至少要进行 m 次 *INCREMENT*
  - 将这样的 *SET* 和它前面的 *INCREMENT* 拼在一起即可
  - 总时间 $O(n)$，单次时间 $O(1)$

1 均摊分析的概念

2 解决方案
聚集法
记账法
势能法

3 均摊分析的应用

记账法

对不同操作赋予不同的费用，平摊代价与实际代价的差叫存款，存款必须是非负的

- 栈操作

操作	实际代价	平摊代价
PUSH	1	2
POP	1	0
MULTIPOP	$min(k, s)$	0

  - 对 *PUSH* 操作多收费，多出来的 1 元钱放在入栈的对象上，弹出时刚好抵消时间代价
  - 因为栈中对象数不可能为负，所以存款不可能为负
- 二进制计数器
  - 每次 *INCREMENT* 至多把一个 0 变成 1
  - 平摊代价为 2，用于将 0 转为 1，并且放在这个新出现的 1 上
  - 当需要将 1 转为 0 时，收回这个钱
  - 因为 1 的个数不可能是负数，所以存款不可能为负

1 均摊分析的概念

2 解决方案
聚集法
记账法
势能法

3 均摊分析的应用

势能法

- 对不同的状态赋予不同的势能 $\Phi(D)$
- $Cost_{average_i} = Cost_{real_i} + \Phi(D_i) - \Phi(D_{i-1})$
- 只要 $\forall i, \Phi(D_i) \geq \Phi(D_0)$
- 即可保证实际代价小于均摊代价
- 与记账法的区别在于赋值的是数据结构还是操作

势能法

- 栈操作
  - $\Phi(D_i) = D_i$ 中栈的对象个数
  - $PUSH = 1 + 1$
  - $POP = MULTIPOP = 0$
- 二进制计算器
  - $\Phi(D_i) = D_i$ 中 1 个数
  - $INCREMENT = 2$
  - 由于 $\Phi(D)$ 有界，所以即使 $\Phi(D_n) < \Phi(D_0)$ 也没影响

势能法

考虑普通二叉最小堆上最坏运行时间为 $O(\log(n))$ 的操作 *INSERT* 和 $EXTRACT - MIN$。请给出势函数 Φ，使得 *INSERT* 操作的平摊代价为 $O(\log(n))$，$EXTRACT - MIN$ 的平摊代价为 $O(1)$。

- 我们记 $\Phi(D) = D$
- $INSERT = \log(n) + \log(n)$
- $EXTRACT - MIN = 1$
- 说明如何用两个普通的栈来实现一个队列，使得每个 *ENQUEUE* 和 *DEQUEUE* 操作的平摊代价都为 $O(1)$。

1 均摊分析的概念

2 解决方案

3 均摊分析的应用
- 动态表
- MTF 链表访问

1 均摊分析的概念

2 解决方案

3 均摊分析的应用
动态表
MTF 链表访问

扩张表示每次扩张到原来的两倍

- 聚集分析
  - 第 i 次插入的代价 $c_i = \begin{cases} k, i = 2^k \\ 1, others \end{cases}$
  - 总代价 $\sum_{i=1}^{n} c_i \leq n + 2 * n = 3 * n$
  - 均摊复杂度 $O(1)$
- 记账法
  - 插入收费 3 元，多的两块钱赋给该元素以及与该元素相差 2^{k-1} 的元素，其中当前表的长度为 2^k
  - 复制的时候刚好用完存款
- 势能函数
  - $\Phi(T) = 2num - size$
  - 不引发扩张时，$a_i = c_i + \Delta\Phi = 1 + 2 = 3$
  - 引发扩张时，$a_i = c_i + \Delta\Phi = 2^k + 1 + (2 - 2^k) = 3$

收缩的情况类似，甚至可以小于 2

1 均摊分析的概念

2 解决方案

3 均摊分析的应用
动态表
MTF 链表访问

MTF 链表访问

- 在一个单链表上
  - 访问第 i 个元素的代价是 i
  - 交换相邻元素的代价为固定常数值
- MTF 最多 4 倍坏
- 势能法
  - 设任意一个调整算法为 A
  - $\Phi(D_i) = MTF$ 与 A 相比的逆序对
  - 设访问 x 前，x 在 MTF 的 k 位置，在 A 中的 i 位置
  - 先进行 MTF 移动
    - 只会影响与 x 有关的逆序对
    - 最多增加 $min(k-1, i-1)$, 最少减少 $k-1-min(k-1, i-1)$
    - $\Delta\Phi \leq 4min(k-1, i-1) - 2(k-1)$
    - $c_i \leq 4 * i$
  - 将 A 的移动建模成 MTF 的充能行为
    - A 做 1 次元素交换，最多使得势能增加 2
    - 充能行为的均摊代价是 $2 < 4$
  - 综合以上，得证

Thanks!