17.Amortized analysis

Hsu, Lih-Hsing

Computer Theory Lab.

- The time required to perform a sequence of data structure operations in average over all the operations performed.
- Average performance of each operation in the worst case.

- For all n, a sequence of n operations takes worst time $T(n)$ in total. The amortize cost of each operation is $\frac{T(n)}{n}$.

Three common techniques

- aggregate analysis
- accounting method
- potential method

17.1 The aggregate analysis

- Stack operation
  - PUSH(S, x)
  - POP(S)
  - MULTIPOP(S, k)

MULTIPOP(S, k)

```
while not STACK-EMPTY(S) and k ≠ 0
    do POP(S)
        k ← k – 1
```


Action of MULTIPOP on a stack S

initial	MULTIPOP(S,4)	MULTIPOP(S,7)
top → 23		
17		
6		
39		
10	top → 10	
47	47	

- Analysis a sequence of n PUSH, POP, and MULTIPOP operation on an **initially empty stack**.
- $O(n^2)$
- $O(n)$ (better bound)
- The amortize cost of an operation is $\frac{O(n)}{n} = O(1)$.

Incremental of a binary counter

Counter value	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	Total cost
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

INCREMENT

INCREMENT(A)

1 $i \leftarrow 0$

2 **while** $i < length[A]$ and $A[i] = 1$

3 **do** $A[i] \leftarrow 0$

4 $i \leftarrow i + 1$

5 **if** $i < length[A]$

6 **then** $A[i] \leftarrow 1$

Analysis:

- O(n k) (k is the word length)
- Amortize Analysis:

$$\sum_{i=0}^{\lfloor \log n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i}$$

$$= 2n$$

$$\Rightarrow \text{the amortize cost is } \frac{O(n)}{n} = O(1)$$

17.2 The accounting method

- We assign different charges to different operations, with some operations charged more or less than the actually cost. The amount we charge an operation is called its **amortized cost**.

- When an operation's amortized cost exceeds its actually cost, the difference is assign to specific object in the data structure as **credit**. Credit can be used later on to help pay for operations whose amortized cost is less than their actual cost.

- If we want analysis with amortized costs to show that in the worst cast the average cost per operation is small, the total amortized cost of a sequence of operations must be an **upper bound** on the total actual cost of the sequence.
- Moreover, as in aggregate analysis, this relationship must hold for all sequences of operations.

- If we denote the actual cost of the ith operation by c_i and the amortized cost of the ith operation by $\hat{c}_i$, we require

  $$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{n=1}^{n} c_i$$

  for all sequence of n operations.
- The total credit stored in the data structure is the difference between the total actual cost, or $\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i$.

Stack operation

PUSH	1	PUSH	2
POP	1	POP	0
MULTIPOP	min{k,s}	MULTIPOP	0

- Amortize cost: O(1)

Incrementing a binary counter

$0\rightarrow 1$	1	$0\rightarrow 1$	2
$1\rightarrow 0$	1	$1\rightarrow 0$	0

- Each time, there is exactly one 0 that is changed into 1.
- The number of 1's in the counter is never negative!
- Amortized cost is at most 2 = O(1).

17.3 The potential method

- initial data structure D_0.
- D_i: the data structure of the result after applying the *i*-th operation to the data structure D_{i-1}.
- C_i: actual cost of the *i*-th operation.

- A potential function Φ maps each data structure D_i to a real number $\Phi(D_i)$, which is the potential associated with data structure D_i.
- The amortized cost $\hat{c}_i$ of the *i*-th operation with respect to potential Φ is defined by $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$.

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1}))$$

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

- If $\Phi(D_i) \geq \Phi(D_0)$ then $\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$.
- If $\Phi(D_i) \geq \Phi(D_{i-1})$ then the potential increases.

Stack operations

- $\Phi(D_i) =$ the number of objects in the stack of the ith operation.
- $\Phi(D_0) = 0$
- $\Phi(D_i) \geq 0$

PUSH

$$\Phi(D_i) - \Phi(D_{i-1}) = (s+1) - s = 1$$

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 1 = 2$$

Computer Theory Lab.

MULTIPOP

$$k' = min\{k, s\}$$

$$\Phi(D_i) - \Phi(D_{i-1}) = -k'$$

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = k' - k' = 0$$

POP

$$\hat{c}_i = 0$$

- The amortized cost of each of these operations is O(1).

Incrementing a binary counter

- $\Phi(D_i)$ = the number of 1's in the counter after the *i*th operations = b_i .
- t_i = the ith INCREMENT operation resets t_i bits.
- $\Phi(D_i) = b_i \le b_{i-1} - t_i + 1$
- $\Phi(D_i) - \Phi(D_{i-1}) \le (b_{i-1} - t_i + 1) - b_{i-1} = 1 - t_i$
- $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \le (t_i + 1) + (1 - t_i) = 2$
- Amortized cost = O(1)

Even if the counter does not start at zero:

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \hat{c}_i - \Phi(D_n) + \Phi(D_0)$$

$$\leq \sum_{i=1}^{n} 2 - b_n + b_0 = 2n - b_n + b_0$$

17.4 Dynamic tables

17.4.1 Table expansion

- TABLE-INSERT
- TABLE-DELETE (discuss later)
- load-factor $\alpha(T)$ $\quad (\alpha(T) \geq \frac{1}{2})$
- $\alpha(T) = \frac{num[T]}{size[T]}$ (load factor)

TABLE_INSERT

TABLE_INSERT(T, x)

if *size*[*T*] = 0
    then allocate *table*[*T*] with 1 slot
        size[T] $\leftarrow$ 1
if *num*[*T*] = *size*[*T*]
    then allocate *new-table* with $2 \cdot size[T]$ slots
        insert all items in *table[T]* in *new-table*
        free *table[T]*
        table[*T*] $\leftarrow$ *new-table*
        size[*T*] $\leftarrow 2 \cdot size[T]$
insert *x* into *table[T]*
num[*T*] $\leftarrow$ *num*[*T*] + 1

Aggregate method:

$$c_i = \begin{cases} i & \text{if } i\text{-1 is an exact power of 2} \\ 1 & \text{otherwise} \end{cases}$$

$$\sum_{i=1}^{n} c_i = n + \sum_{j=1}^{\lfloor \lg n \rfloor} 2^j < n + 2n = 3n$$

- amortized cost = 3

Accounting method:

- each item pays for 3 elementary insertions;

1. inserting itself in the current table,
2. moving itself when the table is expanded, and
3. moving another item that has already been moved once when the table is expanded.

Potential method:

$$\Phi(T) = 2 \cdot num[T] - size[T]$$

- $size_i = size_{i-1}$ (not expansion)

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= 1 + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1})$$
$$= 1 + (2 \cdot num_i - size_i) - (2(num_i - 1) - size_i)$$
$$= 3$$

Example:

$$size_i = size_{i-1} = 16, num_i = 13$$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= 1 + (2 \cdot 13 - 16) - (2 \cdot 12 - 16)$$
$$= 1 + (2 \cdot 13 - 16) - (2 \cdot 12 - 16)$$
$$= 3$$

- $size_i / 2 = size_{i-1} = num_i - 1$ (expansion)

$$\begin{aligned}\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= num_i + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1}) \\ &= num_i + (2 \cdot num_i - (2 \cdot num_i - 2)) \\ &\quad - (2(num_i - 1) - (num_i - 1)) \\ &= num_i + 2 - (num_i - 1) \\ &= 3\end{aligned}$$

Computer Theory Lab.

Example:

$size_i / 2 = size_{i-1} = num_i - 1 = 16$

$$\begin{aligned}\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 17 + (2 \cdot 17 - 32) - (2 \cdot 16 - 16) \\ &= 17 + (2 \cdot 17 - (2 \cdot 17 - 2)) - (2 \cdot 16 - 16) \\ &= 17 + 2 - 16 \\ &= 3\end{aligned}$$

- Amortized cost = 3

17.4.2 Table expansion and contraction

- To implement a TABLE-DELETE operation, it is desirable to contract the table when the load factor of the table becomes too small, so that the waste space is not exorbitant.

Goal:

- The load factor of the dynamic table is bounded below by a constant.
- The amortized cost of a table operation is bounded above by a constant.
- Set load factor $\geq \frac{1}{2}$

- The first $\frac{n}{2}$ operations are inserted. The second $\frac{n}{2}$ operations, we perform I, D, D, I, I, D, D, …
- Total cost of these n operations is $\Theta(n^2)$. Hence the amortized cost is $\Theta(n)$.
- Set load factor $\geq \frac{1}{4}$ (as TABLE_DELETE) (after the contraction, the load factor become $\frac{1}{2}$)

Computer Theory Lab.

$$\Phi(T) = \begin{cases} 2num[T] - size[T] & if\, \alpha(T) \geq \frac{1}{2} \\ \frac{size[T]}{2} - num[T] & if\, \alpha(T) < \frac{1}{2} \end{cases}$$

- Initial

$$num_0 = 0$$
$$size_0 = 0$$
$$\alpha_0 = 1$$
$$\Phi_0 = 0$$

TABLE-INSERT

- if $\alpha_{i-1} \geq \frac{1}{2}$, same as before.

- if $\alpha_{i-1} < \frac{1}{2}$

  if $\alpha_i < \frac{1}{2}$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= 1 + (\frac{size_i}{2} - num_i) - (\frac{size_{i-1}}{2} - num_{i-1})$$
$$= 1 + (\frac{size_i}{2} - num_i) - (\frac{size_i}{2} - (num_i - 1))$$
$$= 0$$

Example:

$$size_i = size_{i-1} = 16, num_i = 6$$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (\frac{16}{2} - 6) - (\frac{16}{2} - 5)$$

$$= 0$$

- If $\alpha_i \geq \frac{1}{2}$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$

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

$$= 1 + (2(num_{i-1} + 1) - size_{i-1}) - (\frac{size_{i-1}}{2} - (num_{i-1}))$$

$$= 3num_{i-1} - \frac{3}{2} size_{i-1} + 3$$

$$= 3\alpha_{i-1} size_{i-1} - \frac{3}{2} size_{i-1} + 3$$

$$< \frac{3}{2} size_{i-1} - \frac{3}{2} size_{i-1} + 3$$

$$= 3$$

Example:

$$size_i = size_{i-1} = 16, num_i = 8$$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$

$$= 1 + (2 \cdot 8 - 16) - (\frac{16}{2} - 7)$$

$$\leq 3$$

Amortized cost of Table insert is O(1).

TABLE-DELETE

- If $\alpha_{i-1} < \frac{1}{2}$

  α_i does not cause a contraction (i.e., $size_i = size_{i-1}$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$

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

$$= 1 + (\frac{size_i}{2} - num_i) - (\frac{size_i}{2} - (num_i + 1))$$

$$= 2$$

Example:

$$size_i = size_{i-1} = 16,\ num_i = 6$$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (\frac{16}{2} - 6) - (\frac{16}{2} - 7)$$

$$= 2$$

- α_i causes a contraction

$$c_i = num_i + 1 \text{ (actual cost)}$$

$$\frac{size_i}{2} = \frac{size_{i-1}}{4} = num_i + 1$$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= (num_i + 1) + (\frac{size_i}{2} - num_i) - (\frac{size_{i-1}}{2} - num_{i-1})$$
$$= (num_i + 1) + ((num_i + 1) - num_i)$$
$$-((2num_i + 2) - (num_i + 1))$$
$$= 1$$

Example:

$$\frac{size_i}{2} = \frac{size_{i-1}}{4} = num_i + 1 = 4$$

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= (3 + 1) + (\frac{8}{2} - 3) - (\frac{16}{2} - 4)$$
$$= 1$$

- if $\alpha_{i-1} \geq \frac{1}{2}$ (Exercise 18.4.3)
- Amortized cost O(1).