

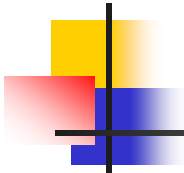


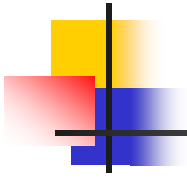
## 17. Amortized analysis

---

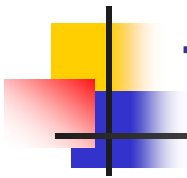
Hsu, Lih-Hsing

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( $S, k$ )

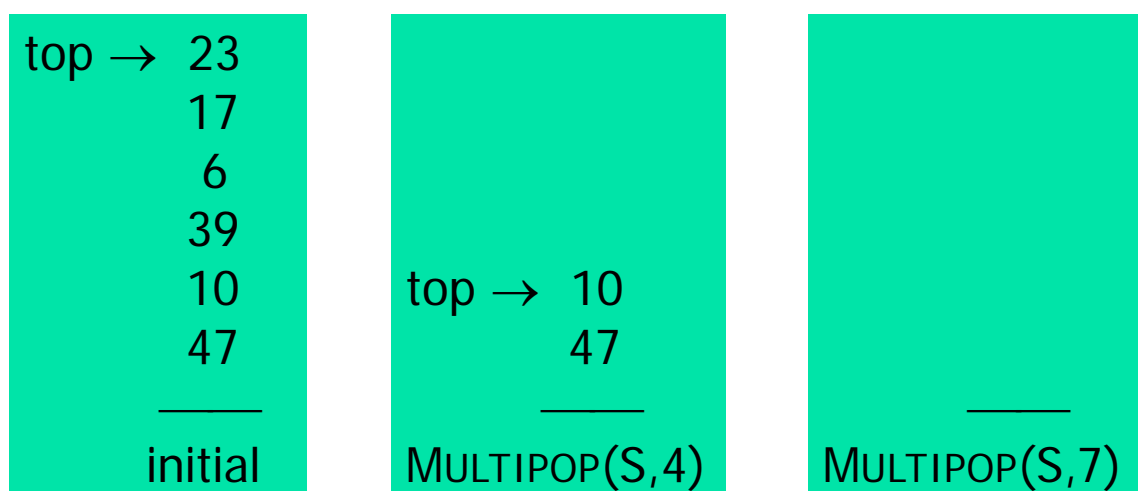
MULTIPOP( $S, k$ )

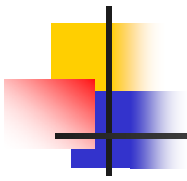
```

1  while not STACK-EMPTY( $S$ ) and  $k \neq 0$ 
2      do POP( $S$ )
3       $k \leftarrow k - 1$ 

```

## Action of MULTIPOP on a stack $S$



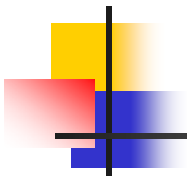


- Analysis a sequence of  $n$  PUSH, POP, and MULTIPOP operation on an **initially empty stack**.
- $O(n^2)$
- $O(n)$  (better bound)
- The amortize cost of an operation is  $\frac{O(n)}{n} = O(1)$ .

Chapter 17

P.7

Computer Theory Lab.



## 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

Chapter 17

P.8



# INCREMENT

INCREMENT( $A$ )

1  $i \leftarrow 0$

2 **while**  $i < \text{length}[A]$  and  $A[i] = 1$

3     **do**  $A[i] \leftarrow 0$

4              $i \leftarrow i + 1$

5 **if**  $i < \text{length}[A]$

6     **then**  $A[i] \leftarrow 1$

Chapter 17

P.9

Computer Theory Lab.



# Analysis:

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

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

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

Chapter 17


P.10

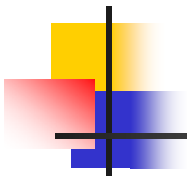


## 17.2 The accounting method

---

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

- 
- 
- When an operation's amortized cost exceeds its actual cost, the difference is assigned to a 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.

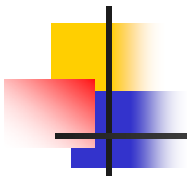


- If we want analysis with amortized costs to show that in the worst cast the average cost per operation is small, the total amortized cost of a sequence of operations must be an **upper bound** on the total actual cost of the sequence.
- Moreover, as in aggregate analysis, this relationship must hold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

Chapter 17

P.13

Computer Theory Lab.



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

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

for all sequence of  $n$  operations.

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

Chapter 17

P.14



## 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  $2 = O(1)$ .




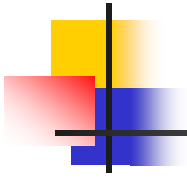


## 17.3 The potential method

---

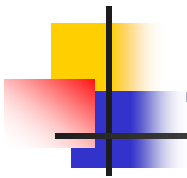
- initial data structure  $D_0$ .
- $D_i$ : the data structure of the result after applying the  $i$ -th operation to the data structure  $D_{i-1}$ .
- $C_i$ : actual cost of the  $i$ -th operation.

- 
- 
- A potential function  $\Phi$  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  $\Phi$  is defined by  $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$ .



$$\begin{aligned}\sum_{i=1}^n \hat{c}_i &= \sum_{i=1}^n (c_i + \Phi(D_i) - \Phi(D_{i-1})) \\ &= \sum_{i=1}^n c_i + \Phi(D_n) - \Phi(D_0)\end{aligned}$$

- 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  $i$ th operation.
- $\Phi(D_0) = 0$
- $\Phi(D_i) \geq 0$



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

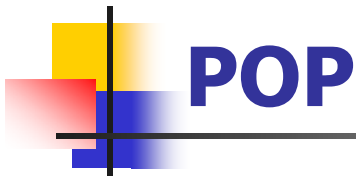
$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 1 = 2$$



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

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

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



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

- 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  $i$ th INCREMENT operation resets  $t_i$  bits.
- $\Phi(D_i) = b_i \leq b_{i-1} - t_i + 1$
- $\Phi(D_i) - \Phi(D_{i-1}) \leq (b_{i-1} - t_i + 1) - b_{i-1} = 1 - t_i$
- $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \leq (t_i + 1) + (1 - t_i) = 2$
- Amortized cost =  $O(1)$

Even if the counter does not start at zero:

$$\begin{aligned}\sum_{i=1}^n c_i &= \sum_{i=1}^n \hat{c}_i - \Phi(D_n) + \Phi(D_0) \\ &\leq \sum_{i=1}^n 2 - b_n + b_0 = 2n - b_n + b_0\end{aligned}$$

## 17.4 Dynamic tables

## 17.4.1 Table expansion

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

## TABLE\_INSERT

```

TABLE_INSERT(T, x)
1  if size[T] = 0
2    then allocate table[T] with 1 slot
3    size[T] ← 1
4  if num[T] = size[T]
5    then allocate new-table with  $2 \cdot \text{size}[T]$  slots
6    insert all items in table[T] in new-table
7    free table[T]
8    table[T] ← new-table
9    size[T] ←  $2 \cdot \text{size}[T]$ 
10 insert x into table[T]
11 num[T] ← num[T] + 1
  
```



## Aggregate method:

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

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

- 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:

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

- $\text{size}_i = \text{size}_{i-1}$  (not expansion)

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

$$= 1 + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1})$$

$$= 1 + (2 \cdot \text{num}_i - \text{size}_i) - (2(\text{num}_i - 1) - \text{size}_i)$$

$$= 3$$



## Example:

$$\text{size}_i = \text{size}_{i-1} = 16, \text{num}_i = 13$$

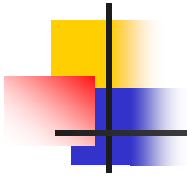
$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$

$$= 1 + (2 \cdot 13 - 16) - (2 \cdot 12 - 16)$$

$$= 1 + (2 \cdot 13 - 16) - (2 \cdot 12 - 16)$$

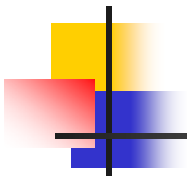
$$= 3$$





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

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



*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.



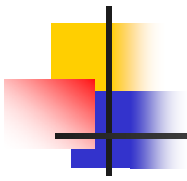
### Goal:

---

- 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  $\frac{1}{2}$ )

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



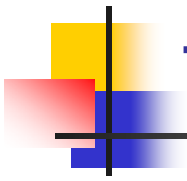
## ■ Initial

$$num_0 = 0$$

$$size_0 = 0$$

$$\alpha_0 = 1$$

$$\Phi_0 = 0$$



## TABLE-INSERT

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

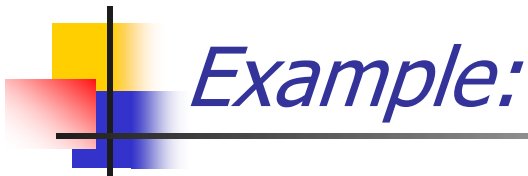
■ if  $\alpha_{i-1} < \frac{1}{2}$   
     if  $\alpha_i < \frac{1}{2}$

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

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

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

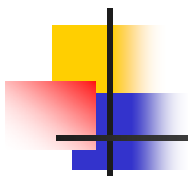
$$= 0$$



*Example:*

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

$$\begin{aligned}\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} = 1 + \left(\frac{16}{2} - 6\right) - \left(\frac{16}{2} - 5\right) \\ &= 0\end{aligned}$$



■ If  $\alpha_i \geq \frac{1}{2}$

$$\begin{aligned}\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2num_i - size_i) - \left(\frac{size_{i-1}}{2} - num_{i-1}\right) \\ &= 1 + (2(num_{i-1} + 1) - size_{i-1}) - \left(\frac{size_{i-1}}{2} - (num_{i-1})\right) \\ &= 3num_{i-1} - \frac{3}{2}size_{i-1} + 3 \\ &= 3\alpha_{i-1}size_{i-1} - \frac{3}{2}size_{i-1} + 3 \\ &< \frac{3}{2}size_{i-1} - \frac{3}{2}size_{i-1} + 3 \\ &= 3\end{aligned}$$



## Example:

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

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

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

$$\leq 3$$

**Amortized cost of Table insert is  $O(1)$ .**



## TABLE-DELETE

■ If  $\alpha_{i-1} < \frac{1}{2}$

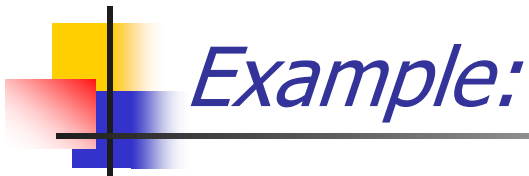
$\alpha_i$  does not cause a contraction (i.e.,  
 $size_i = size_{i-1}$ )

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

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

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

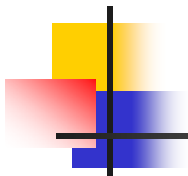
$$= 2$$



*Example:*

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

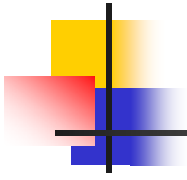
$$\begin{aligned}\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} = 1 + \left(\frac{16}{2} - 6\right) - \left(\frac{16}{2} - 7\right) \\ &= 2\end{aligned}$$



- $\alpha_i$  causes a contraction

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

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

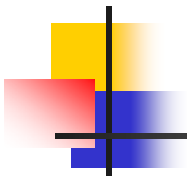


$$\begin{aligned}
 \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
 &= (num_i + 1) + \left(\frac{size_i}{2} - num_i\right) - \left(\frac{size_{i-1}}{2} - num_{i-1}\right) \\
 &= (num_i + 1) + ((num_i + 1) - num_i) \\
 &\quad - ((2num_i + 2) - (num_i + 1)) \\
 &= 1
 \end{aligned}$$

Chapter 17

P.47

Computer Theory Lab.



*Example:*

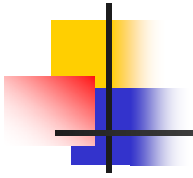
$$\frac{size_i}{2} = \frac{size_{i-1}}{4} = num_i + 1 = 4$$

$$\begin{aligned}
 \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
 &= (3 + 1) + \left(\frac{8}{2} - 3\right) - \left(\frac{16}{2} - 4\right) \\
 &= 1
 \end{aligned}$$

Chapter 17

P.48





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