Amortized Analysis

CSE 344 Design & Analysis of Algorithms

Subhasish Mazumdar

Amortized Analysis

- What is the *worst-case cost* of a *sequence* of n operations?
- Simple Analysis:
  Let the worst case cost per opn = x
  Hence, the worst case time for sequence of n ops = $n \cdot x$

Amortized Analysis

- Instead, focus on the sequence.

  Even in the worst case, one operation may be expensive, but the next one may have to be cheap.

  So even in the worst case, the cost of the sequence may be much less than $n \cdot x$

Amortized Analysis

- So, the question this helps answer is:
  in the worst-case sequence of n opns,
  what is the average cost per operation?
- Yet, this is *not* average-case analysis!
  average case analysis uses probabilities;
  amortized analysis does not.

Amortized Analysis

Main Entry: *amortize*

Function: transitive verb

Etymology: Middle English amortisen to deaden, alienate in mortmain, modification of Middle French amortiss-, stem of amortir, from (assumed) Vulgar Latin admortire to deaden, from Latin ad- + mort-, mors death – more at MURDER

1. to provide for the gradual extinguishment of (as a mortgage) usually by contribution to a sinking fund at the time of each periodic interest payment
2. to amortize an expenditure for
   $<$amortize intangibles$>$
   $<$amortize the new factory$>$

Date: 1882

Example 1: Stack

MULTIPOP(S, k)

while not empty(S) and $k <> 0$ **do**

  POP(S)

  $k \leftarrow k - 1$

Example 1: Stack

Let s = size of stack S
n = num of ops in sequence
Assume constant for cost = 1

Opn	*Actual cost*	*Worst-case cost*
$\text{PUSH}(S, x)$	1	1
$\text{POP}(S)$	1	1
$\text{MULTIPOP}(S, k)$	$min(s, k)$	$O(n)$

- Simple Analysis:
  worst case time per opn = $O(n)$ (last row of table)
  hence, worst case time for sequence = $O(n^2)$.

Stack: Aggregate Analysis Method

- Focus on the sequence.
- An object that is pushed once can be popped only once.
- Hence, total cost of POP and MULTIPOP opns
  $\leq$ Total cost of PUSH opns
- *Maximum total cost of all Push opns = n* (since there can be at most n PUSH opns)
- Hence, *total cost of Pop and Multipop opns* $\leq n$

Stack: Aggregate Analysis Method

- Thus, total cost of sequence $\leq 2n$
  i.e., $O(n)$
- Now redo the analysis using constants
- Amortized cost of one opn = $O(n)\,/\,n = O(1)$
- Note: no probabilities were used and
  we looked at the worst-case scenario.

Example 2: k-bit Binary Counter (initially zero)

- See text for the algorithm.
- Costs:
  1 for setting a bit
  1 for resetting a bit

$k-1$	...	3	2	1	0
0	...	0	0	0	0
0	...	0	0	0	1
0	...	0	0	1	0
0	...	0	0	1	1
0	...	0	1	0	0
0	...	0	1	0	1
0	...	0	1	1	0
0	...	0	1	1	1
0	...	1	0	0	0
0	...	1	0	0	1
0	...	1	0	1	0
0	...	1	0	1	1
0	...	1	1	0	0
0	...	1	1	0	1
0	...	1	1	1	0
0	...	1	1	1	1
	...				

Example 2: k-bit Binary Counter (initially zero)

Opn	*Actual Worst Case cost*
INCREMENT	$\Theta(k)$

- Simple Analysis:
  Worst case time per opn = $\Theta(k)$,
  (from last column of table)
  hence, worst case time for sequence = $\Theta(n \cdot k)$

Counter: Aggregate Analysis

- cost of an INCREMENT opn = # of bits that toggle
- Now focus on the sequence and count how many times bits toggle.
- Assume constant = 1

Counter: Aggregate Analysis

1. Bit 0 toggles n times,
   Bit 1 toggles $n/2$ times, ...
2. Total cost = total # of bit togglings
   $= n + n/2 + n/4 + ... + n/2^{k-1}$
   (assumed n is power of 2;
   else use floors on n/2, n/4,...)
   $< 2n$
   $= O(n)$
3. Amortized cost of one opn = $O(n)/n = O(1)$

- Note: worst-case time is now independent of k.

The Accounting Method

- Assign a charge for each opn — its *amortized cost*.
- Set up an account for the data object with an initial balance of 0. (We can start with any constant c.)
- For each operation in the sequence:
  if charge $>$ actual cost then *credit* the account;
  if charge $<$ actual cost then *debit* the account.
- Constraint: *throughout the sequence, the account balance must never be negative*.

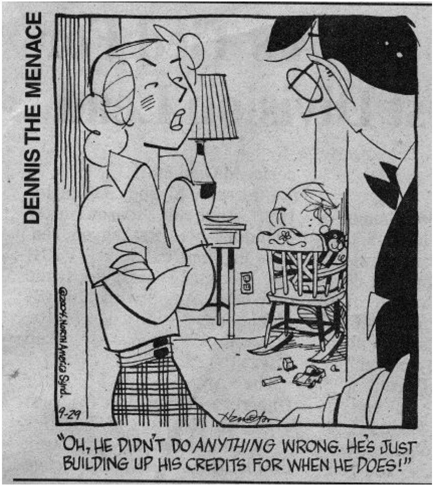
The Accounting Method

- Total amortized cost
  = total charge for the sequence
  is an *upper bound* on the actual cost.
- i.e., total actual cost $\leq$ total amortized cost

The Accounting Method

- Total amortized cost
  = total charge for the sequence
  is an *upper bound* on the actual cost.
- i.e., total actual cost $\leq$ total amortized cost
- So, if the constraint holds, then add up the amortized costs (assigned charges) instead of the actual costs.

The Accounting Method



Stack: Accounting Method

Opn	*Actual cost*	*Assigned Amortized cost*
$\text{PUSH}(S, x)$	1	2
$\text{POP}(S)$	1	0
$\text{MULTIPOP}(S, k)$	$\boxed{min(s, k)}$	$\boxed{0}$

- Useful?
  Yes: variable (actual) cost $\rightarrow$ fixed (amort) cost
  Correct?
  Yes: the constraint holds on any sequence of ops. Why?

Stack: Accounting Method

Opn	*Actual cost*	*Assigned Amortized cost*
PUSH(S, x)	1	2
POP(S)	1	0
MULTIPOP(S, k)	$min(s, k)$	0

- Max amortized cost over all operations = 2 (from last column)
- Total amortized cost for sequence $\leq 2n$

  $= O(n)$

- Hence the total actual cost = $O(n)$ Why?

Counter: Accounting method

- actual cost depends on number of bits toggled. So, we will base our costs on bit operations.

Opn	Actual *Worst Case cost*	Assigned Amortized cost
INCREMENT	$\boxed{\leq k}$	$\boxed{2}$

- Useful?
- Correct?

Counter: Accounting method

Counter = 0

...	0	0	0	0

INCREMENT
(consume \$1 to flip bit 0; store \$1 with the bit)

...	0	0	0	\$1 1

Counter: Accounting method

Counter = 7

...		0	\$1	1	\$1	1	\$1	1

INCREMENT
(consume \$1 for flipping bit 7; store \$1 with the bit))

...	\$1	1		0		0		0

Counter: Accounting method

Opn	Actual *Worst Case cost*	Assigned Amortized cost
INCREMENT	$\leq k$	2

- Total amortized cost of sequence = $2n$ (from last column)
- Hence actual cost is $O(n)$

The Potential Method

- Associate a *potential energy* $\Phi(D)$ with a data structure D.

The Potential Method

- Associate a *potential energy* $\Phi(D)$ with a data structure D.
- As the i^{th} opn changes the data structure
  from D_{i-1} to D_i,
  its potential changes from $\Phi(D_{i-1})$ to $\Phi(D_i)$

The Potential Method

- Define:
  Amortized cost of i^{th} opn =
  actual cost of i^{th} opn $+\Delta\,\Phi$

The Potential Method

- i.e.,
  Amortized cost of i^{th} opn =
  actual cost of i^{th} opn $+\Phi(D_i) - \Phi(D_{i-1})$

The Potential Method

- i.e.,
  Amortized cost of i^{th} opn =
  actual cost of i^{th} opn $+\Phi(D_i) - \Phi(D_{i-1})$
- So, for a sequence, the total amortized cost =
  total actual cost $+\Phi(D_n) - \Phi(D_0)$

The Potential Method

- i.e.,
  Amortized cost of i^{th} opn =
  actual cost of i^{th} opn $+\Phi(D_i) - \Phi(D_{i-1})$
- So, for a sequence, the total amortized cost =
  total actual cost $+\Phi(D_n) - \Phi(D_0)$
- Constraint: choose Φ such that

$$\Phi(D_i) \geq \Phi(D_0) \text{ for all } i$$

The Potential Method

- i.e.,
  Amortized cost of i^{th} opn =
  actual cost of i^{th} opn $+\Phi(D_i) - \Phi(D_{i-1})$
- So, for a sequence, the total amortized cost =
  total actual cost $+\Phi(D_n) - \Phi(D_0)$
- Constraint: choose Φ such that

$$\Phi(D_i) \geq \Phi(D_0) \text{ for all } i$$

- Then total amortized cost of sequence is an *upper bound* on the actual cost.

The Potential Method

- Actually, $\Phi(D_n) \geq \Phi(D_0)$ is enough, but we rarely know which is the last operation.

The Potential Method

- The potential energy is like the account balance in the Accounting Method; an increase in energy is like credit, a decrease like debit. Then, how is this different from the Accounting Method?

Stack: Potential method

- Choose $\Phi(S)$ = # of elements in S

  $\Phi(D_0) = 0$

  Is $\Phi(D_i) \geq \Phi(D_0) = 0$ for all i?

Stack: Potential method

- Actual cost of PUSH = 1
  Amortized cost of PUSH = $1 + 1$ (why?)

  Actual cost of POP = 1
  Amortized cost of POP = $1 - 1$ (why?)

  Actual cost of MULTIPOP = $min(s, k)$
  Amortized cost of MULTIPOP =
  $min(s, k) - min(s, k)$ (why?) $= 0$

  Amortized cost of a seq of n opns = $O(n)$

Counter: Potential method

- Choose Φ to be the # of 1-bits in the counter.
  The counter is initially 0

  $\Phi(D_0) = 0$

  Is $\Phi(D_i) \geq \Phi(D_0) = 0$ for all i?

Counter: Potential method

- INCREMENT resets a variable number of bits but sets exactly ONE bit.
- Let the i^{th} INCREMENT opn *reset* t_i bits
  while *setting* exactly one bit.
  Actual cost = $t_i + 1$

Counter: Potential method

- INCREMENT resets a variable number of bits but sets exactly ONE bit.
- Let the i^{th} INCREMENT opn *reset* t_i bits
  while *setting* exactly one bit.
  Actual cost = $t_i + 1$
  Note: If the counter overflows, then $t_i = k$ and no bit is set!
  In that case, actual cost $< t_i + 1$

  So, actual cost $\leq \ (t_i + 1)$

Counter: Potential method

- $$\Delta\Phi = 1 - t_i \text{ (Why?)}$$

  Note: If the counter overflows, then ...

  $$\Delta\Phi = -t_i$$

  $$\text{So, } \Delta\Phi \leq 1 - t_i$$

Counter: Potential method

- Hence, amortized cost $\leq (t_i + 1) + \Delta\Phi$

$$= (t_i + 1) + (1 - t_i) \;=\; 2$$

Total Amortized cost of a sequence of n opns
$\leq 2n = O(n)$

Counter: non-zero starting state

- Let b_i bits be *set* in the i^{th} state, $0 \leq i \leq n$, i.e.,

$$\Phi(D_0) = b_0$$

  where $0 \leq b_0,\ b_n \leq k$
- We are no longer sure that $\Phi(D_n) \geq \ \Phi(D_0)$.
  Is the total cost still $O(n)$?
- Use the potential method but without the constraints: go back to the definition.

Counter: non-zero starting state

- Total amortized cost
  = Total actual cost $+\Phi(D_n) - \Phi(D_0)$

Counter: non-zero starting state

- Total amortized cost
  = Total actual cost $+\Phi(D_n) - \Phi(D_0)$

  i.e., Total actual cost
  = Total amortized cost $-\Phi(D_n) + \Phi(D_0)$

Counter: non-zero starting state

- Total amortized cost

  = Total actual cost $+\Phi(D_n) - \Phi(D_0)$

  i.e., Total actual cost

  = Total amortized cost $-\Phi(D_n) + \Phi(D_0)$

  $\leq 2n - b_n + b_0$

  $\leq 2n + b_0$ (since $b_n \geq 0$)

  $\leq 2n + O(n)$ ....if $b_0 = O(n)$

  $= O(n)$ ....if $b_0 = O(n)$

Counter: non-zero starting state

- Total amortized cost
  $= \text{Total actual cost } + \Phi(D_n) - \Phi(D_0)$
  i.e., Total actual cost
  $= \text{Total amortized cost } - \Phi(D_n) + \Phi(D_0)$
  $\leq 2n - b_n + b_0$
  $\leq 2n + b_0 \quad (\text{since } b_n \geq 0)$
  $\leq 2n + O(n)$ ....if $b_0 = O(n)$
  $= O(n)$ ....if $b_0 = O(n)$
- When can we be assured that $b_0 = O(n)$?
  Since $b_0 \leq k$, it is enough if $k = O(n)$
  or $n = \Omega(k)$