130A: Amortized Analysis

Subhash Suri

Computer Science Department
Applied Algorithms Group
Algorithms, Complexity, Data Structures, Optimization

February 17, 2015

What is Amortized Analysis?

- Amortize: to put money aside at intervals for gradual payment of a debt. (Webster)
- Data structures typically undergo a sequence of operations, not just a single operation.
- Worst-case analysis simply adds up the worst possible times of the individual operations. Can be unduly pessimistic.
- Average case analysis makes unrealistic assumptions about input, often hard to justify.
- Amortized analysis offer a nice balance.

Why Amortized Analysis?

- Amortized analysis: average runtime per operation over a worst-case sequence of operations.
- If $T(n)$ is total cost over a worst-case sequence of n operations, then

$$\text{Amortized cost per operation} = \frac{T(n)}{n}$$

- Results are both realistic and robust.
- Only an analysis technique, not a algorithm design method.

Amortized Analysis: 3 Methods

- We will discuss 3 different methods for amortized analysis:
  - Aggregate Method
  - Accounting Method
  - Potential Method
- Illustrate on two examples:
  - Stack
  - Binary Counter

Amortized Analysis of Stack with MultiPop

- Stack data structure has two operations:
  - Push(S, x) — push x onto stack S.
  - Pop (S) — pop the top element of S.
- Each of these ops has worst-case cost $O(1)$.
- Add a new operation MultiPop (S, k): it pops top k elements of S.

```
MultiPop $(S, k)$
    while $S \neq \emptyset$ and $k > 0$
        Pop $(S)$
        $k = k - 1$
    end
```

Amortized Analysis of Stack with MultiPop

- Worst-case complexity of n MultiPop Stack ops?
- Because any single MultiPop can be $\Theta(n)$, the standard worst-case bound is $O(n^2)$.
- But this is overly pessimistic.

Amortized Analysis

Theorem: A sequence of n ops, on an initially empty stack, has cost $O(n)$.

Proof:

- Each element is popped at most once (either by Pop or during a MultiPop).
- Total cost of Pops, including MultiPops, is $\leqslant$ number of Pushes, which is at most n.
- Total cost of all operations is $T(n) \leqslant 2n$.
- Thus, amortized cost per operation for this stack is at most 2.
- This bound holds in worst-case, over any sequence!

Amortized Analysis of a Binary Counter

- Binary counter implemented as an array of bits: $A[0 \cdots k-1]$.
- Consider incrementing A starting from 0.

```
Increment ($A$)
    $i = 0$;
    while $i \neq k$ and $A[i] = 1$
        $A[i] = 0$;    $i = i + 1$;
    if $i < k$ then $A[i] = 1$;
```

- Cost of Increment is number of bits flipped.

Amortized Analysis of a Binary Counter

- Binary counter bit sequence:

Value	Bits	Cost
0	$0\cdots000$	0
1	$0\cdots001$	1
2	$0\cdots010$	2
3	$0\cdots011$	1

- In the worst-case, a single increment can flip k bits.
- There are n increments, so by standard worst-case analysis, the cost is $O(nk)$.

Binary Counter Analysis

Theorem: Starting with 0, worst-case cost of n increments is $O(n)$.

Proof:

- Bit $A[0]$ flips each increment.
- Bit $A[1]$ flips every other increment.
- Bit $A[i]$ flips during every $\frac{1}{2^i}$-th increment.
- During n increments, bit $A[i]$ flips $\frac{n}{2^i}$ times.
- Assume $n \leqslant 2^k$; otherwise, the counter resets, and restart analysis.
- Total cost

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

- Amortized cost per increment is 2.

Accounting Method: Second Method of Analysis

- Assign different charges to different operations, some more, some less than true cost.
- These (amortized) charges are artificial, but make the accounting for the total cost easier.
- When an operation's amortized cost $>$ true cost, the data structure accrues credit.
- These credits help pay for operations for which amortized cost $<$ true cost.
- The accounting method makes the first real use of the amortization principle: the aggregate method doesn't really assign varying costs to individual operations.

Accounting Method

- The key is to choose the amortized costs carefully.
- Suppose the actual cost of the ith operation is C_i, and we use some other cost $\hat{C}_i$ as its amortized cost.
- Then, we have to make sure that, for any possible sequence of n operations,

$$\sum_{i=1}^{n} \hat{C}_i \geqslant \sum_{i=1}^{n} C_i$$

- The total credit accrued by the data structure at any point is $(\sum \hat{C}_i - \sum C_i)$, and we better make sure this is never negative.

Accounting Method Analysis of Stack

- The actual costs (C_i) of operations in the stack example are:
  - 1 for Push
  - 1 for Pop
  - $\min(k, s)$ for MultiPop, where s is the stack size when the MultiPop is called.
- Let us assign the following amortized costs $(\hat{C}_i)$ to these ops:
  - 2 for Push
  - 0 for Pop
  - 0 for MultiPop.
- The amortized cost of MultiPop is a constant (in fact, 0), even though it's real cost if variable. The intuition is that the credit accrued through Push operations will be enough to pay for the actual cost of MultiPop.

Accounting Method Analysis of Stack

- Think of stack as a bank, where each deposit and withdrawal costs \$1.
- When an item is Pushed onto the stack, we use \$1 for the Push operation, and leave the second dollar (of its amortized cost) with the item in the bank.
- This second (spare) dollar will be used to pay for the item's Pop.
- The item may be popped either through a single Pop or a MultiPop, but since each item in the stack has a spare dollar allocated to it, the Pop and MultiPop can be entirely paid using that credit.
- The amortized cost of Pop and MultiPop is therefore 0.
- Thus, for any sequence of n Push, Pop, MultiPop, the total amortized cost is $O(n)$, and it is an upper bound on the actual cost.
- Similar analysis for the Binary Counter.

Potential Method: Third Method of Analysis

- Keep track of the potential energy of the data structure.
- Start with initial data structure D_0, and perform n updates.
- Let C_i be the actual cost of ith op;
  let D_i be data structure state after op i.
- Function Φ measures potential of data structure.
- Define amortized cost of operation i as

$$\hat{C}_i = C_i + \Phi(D_i) - \Phi(D_{i-1}).$$

Potential Method

- Since $\hat{C}_i = C_i + \Phi(D_i) - \Phi(D_{i-1})$, by adding them up over the sequence, we get

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

- Rewriting, we get the total actual cost as

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

- Suppose $\Phi(D_i) \geqslant \Phi(D_0)$, for all i, then

$$\sum_{i=1}^{n} C_i \leqslant \sum_{i=1}^{n} \hat{C}_i.$$

Potential Method

- Actual cost of operations is

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

- The last term is non-positive $(\Phi(D_i) \geqslant \Phi(D_0))$.
- So, amortized cost is an upper bound on actual total cost.
- The most creative part of the analysis is often the choice of Φ.

Potential Method Analysis of MultiPop Stack

- Let Φ = number of items on stack.
- Clearly, $\Phi(D_0) = 0$, and $\Phi(D_i) \geqslant \Phi(D_0) = 0$.
- Amortized cost of Push:

$$C_i + (\Phi(D_i) - \Phi(D_{i-1})) \ = \ 1 + 1 \ = \ 2$$

- Amortized cost of Pop:

$$C_i + (\Phi(D_i) - \Phi(D_{i-1})) \ = \ 1 - 1 \ = \ 0$$

- Amortized cost of MultiPop (S, k):

$$C_i + (\Phi(D_i) - \Phi(D_{i-1})) \ = \ \min(s, k) - \min(s, k) \ = \ 0.$$

- Thus, amortized cost per operation is $O(1)$.

Potential Method Analysis of Binary Counter

- Let $\Phi_i = \Phi(D_i)$ be the number of 1's in the counter after ith op.
- Clearly, $\Phi_i \geqslant 0$, for all i.
- Suppose ith Increment resets t_i bits.
- Actual cost $C_i = t_i + 1$.
- If $\Phi_{i-1} = b_i$, then $\Phi_i = b_i - t_i + 1$.
- Amortized cost

$$\begin{aligned}\hat{C}_i &= C_i + (\Phi_i - \Phi_{i-1}) \\ &= (t_i + 1) + (b_i - t_i + 1 - b_i) \\ &= 2\end{aligned}$$

- Total cost is $\leqslant \sum \hat{C}_i = 2n.$