Amortized Analysis of the Multipop Stack

Andreas Klappenecker

Amortized Analysis

The worst case running time might give an overly pessimistic analysis for algorithms performing a sequence of operations on data structures.

In amortized analysis, we average the time required to perform a sequence of operations over the number of operations performed.

Amortized Analysis

There are three different approaches to amortized analysis:

- aggregate method: determine total cost T(n) for a sequence of n operations and calculate the amortized cost per operation as T(n)/n

- accounting method: assign costs to each operation, overcharging some operations and undercharging others. Idea: Early overcharged operations prepay for later undercharged operations.

- potential method: a potential "energy" is associated to the data structure, rather than individual credit to operations.

The Augmented Stack

Consider a stack S that has the following operations:

- Push(S, x) pushes an element x onto the stack S
- Pop(S) pops the top element x of S and returns x
- Multipop(S, k) pops the min(k, |S|) top elements of S

Running time: Push(S,x) is O(1), Pop(S) is O(1),
Multipop(S,k) is O(min(k,|S|)) when implemented by a linked list.

The Aggregate Method

Aggregate Analysis

Suppose that a sequence of n operations takes T(n) time.

Then the amortized cost per operation is defined to be T(n)/n.

The amortized cost applies to each operation, even when there are several different types of operations.

Augmented Stack

In a sequence of n operations, the stack never holds more than n elements. Thus, the cost of a multipop operation is $O(n)$.

It follows that the worst-case running time of any sequence of n stack operations is $O(n^2)$. However, this is an over-estimate!

Aggregate Analysis of a Stack

The total number of pops or multipops in the entire sequence of operations is <= the total number of pushes.

Suppose that the maximum number of Push operations in the sequence is n. So the time for entire sequence is $O(n)$.

Amortized cost per operation: $O(n)/n = O(1)$.

The Accounting Method

Accounting Method

We assign an amortized cost to each operation, where the actual cost might be lower or higher than the amortized cost.

The sum of all the amortized costs in a sequence must be at least the sum of all the actual costs, since we would like to bound the total cost of the sequence by the sum of amortized costs.

How can we ensure this property?

Accounting Method

For each operation in the sequence, we have

- if the amortized cost > actual cost then the difference is stored as a credit with an object in the data structure.
- if the amortized cost < actual cost then use stored credits to make up for the difference.

We have to use stored credit when amortized cost is less than actual cost to make up for the difference, so that sum of costs is always nonnegative (one cannot go into the red).

Accounting Method for Stacks

Let us use the following amortized costs:

- Push: 2
- Pop: 0
- Multipop: 0

The Push operation has cost 1; the additional 1 is stored as credit with the pushed element.

There is always enough credit to pay for each operation.

Each amortized cost is $O(1)$

Thus, the cost of the entire sequence of n operations is $O(n)$.

The Potential Method

The Potential Method

Let D_k denote a data structure after the k^{th} operation, initially starting with a data structure D_0.

A function Φ assigning nonnegative real numbers to data structures such that $\Phi(D_0)=0$ is called a potential function.

Amortized cost: $c_k' = c_k + \Phi(D_k) - \Phi(D_{k-1})$

Potential Function

The total amortized cost of n operations is

$$\begin{aligned}
\sum_{k=1}^{n} c'_k &= \sum_{k=1}^{n} (c_k + \Phi(D_k) - \Phi(D_{k-1})) \\
&= \Phi(D_n) - \Phi(D_0) + \sum_{k=1}^{n} c_k \\
&\geq \sum_{k=1}^{n} c_k
\end{aligned}$$

Potential Function for Stacks

Let S be a stack with m elements.

Define the potential function $\Phi(S)=m$, the number elements on S.

- $c_k' = c_k + \Phi(\text{Push}(S,x)) - \Phi(S) = 1 + 1 = 2$
- $c_k' = c_k + \Phi(\text{Pop}(S,x)) - \Phi(S) = 1 - 1 = 0$
- $c_k' = c_k + \Phi(\text{MultiPop}(S,a)) - \Phi(S) = a - a = 0$

Comparison

In the aggregate method, we first analyze an entire sequence and then calculate amortized cost per operation

In the accounting method, we first first assign amortized cost per operation, and then check that one cannot go into the red.

In the potential method, we define a function a prove that it is a potential function.

The potential method is similar to the accounting method, but one does not need to specify an element to store credit.