Amortization

Sudebkumar Prasant Pal
Department of Computer Science and Engineering and
Centre for Theoretical Studies
IIT Kharagpur 721302, India
email: spp@cse.iitkgp.ernet.in

July 26, 2007

1 Introduction

We wish to use the systematic method of *potential functions* in order to derive amortized computational complexity upper bounds. See [1, 2]

The case of a binary counter

As a simple example to start with, consider the total number of bit operations when an n-bit counter is incremented starting from zero until it reaches its highest count $2^n - 1$. In the worst case carries can trickle down from the LSB to the MSB, requiring n bit operations, for each counter incrementaion. However, this is only the worst case. We can show that the total number of bit operations is $O(n)$, either by direct counting (which requires a smart counting argument), or by using a potential function and a very simple argument. So, we say that the *amortized* cost per counter increment operation is $O(1)$. As an exercise, show that the number of 1's in the counter is a suitable potential function for analysing amortized complexity using the method of potential functions.

[Hint: The potential rises every time the carry trickles by a bit. So, we charge the cost of a bit operation to the falling potential. Is the total rise in potential $O(n)$?]

Overhead costs for 2-4 trees

Imagine starting with an empty 2-4 tree and performing a total of n insertions. Ignoring search costs and considering only overhead costs in maintaining the tree dynamically, we show that the total cost over all the n insertion operations is $O(n)$. So, we say that the amortized cost per operation is $O(1)$.

[Hint: Use the number of nodes with 4 children as the potential function. Observe that a split occur when an insertion is done at such a parent node, and splits trickle destroying such nodes.]

Now we further enrich this problem by allowing deletions too, arbitrarily interspersed between insertions. We claim that the amortized cost per operation remains $O(1)$. As an exercise, design a suitable potential function to establish this amortization result.

References

[1] K. Mehlhorn, Data Structures and Algorithms: Vols. I and III, Springer.

[2] R. Tarjan and C. Wyk, manuscript.