max planck institut informatik

Algorithms and Data Structures **Summer 2008**
K. Mehlhorn and R. Seidel **Fr. Sept 5th**
Exercises 8 and 9

Motivation

We learn more about counters and unbounded arrays and practice amortized analysis. The exercises refer to Sections 3.2 and 3.3 of the book by Mehlhorn and Sanders.
I hope that all of you can do items 1, 2, 3, and 4. For item 5 you should take the argument given in class and edit the text of the proof. Some words need to be changed, but the general format of the proof stays the same. Item 6 requires some thought. Item 7 is tricky.

1. Your manager asks you to change the initialization of α to $\alpha = 2$. He argues that it is wasteful to shrink an array only when three-fourths of it are unused. He proposes to shrink it to an array of size n when $n \leq w/2$, i.e., in *popBack*, the call of *reallocate* is changed to *reallocate*(n). Convince him that this is a bad idea by giving a sequence of m *pushBack* and *popBack* operations that would need time $\Theta(m^2)$ if his proposal was implemented.

2. (Popping many elements) Implement an operation *popBack*(k) that removes the last k elements in amortized constant time independent of k.

3. (Sparse bounded arrays) Implement bounded arrays with constant time for allocating arrays and constant time for the operation $[\cdot]$. All array elements should be (implicitly) initialized to $\perp$. You are not allowed to make any assumptions about the contents of a freshly allocated array. Hint: use an extra array of the same size, and store the number t of array elements to which a value has already been assigned. Therefore $t = 0$ initially. An array entry i to which a value has been assigned stores that value and an index j, $1 \leq j \leq t$, of the extra array, and i is stored in that index of the extra array.

4. (Alternative Global Argument) In class we charged two tokens for each *pushBack* and one token for each *popBack*. Argue that one can alternatively charge three tokens for each *pushBack* and not charge *popBack* at all.

5. (Alternative Local Argument) Charge three tokens for a *pushBack* and no tokens for a *popBack*. Argue that the account contains always at least $n + \max(2(n - w/2), w/2 - n) = \max(3n - w, w/2)$ tokens.

6. (Counters) Consider a nonnegative integer c represented by an array of binary digits, and a sequence of m increment and decrement operations. Initially, $c = 0$.

(a) What is the worst-case execution time of an increment or a decrement as a function of m? Assume that you can work with only one bit per step.

(b) In class, we proved that the amortized cost of the increments is constant if there are no decrements. Give a sequence of m increment and decrement operations with cost $\Theta(m \log m)$.

(c) Give a representation of counters such that you can achieve worst-case constant time for increments and decrements. This item has a one line solution. What is the space requirement of your solution?

(d) Allow each digit d_i to take values from $\{-1, 0, 1\}$. The value of the counter is $c = \sum_i d_i 2^i$. Show that in this *redundant ternary* number system, increments and decrements have constant amortized cost. Is there an easy way to tell whether the value of the counter is zero?

7. (Worst-case constant access time) Suppose, for a real-time application, you need an unbounded array data structure with a *worst-case* constant execution time for all operations. Design such a data structure. Hint: store the elements in up to two arrays. Start moving elements to a larger array well before a small array is completely exhausted.

Have fun with the solutions.