Universität des Saarlandes

Kurt Mehlhorn, Konstantinos Panagiotou and Reto Spöhel

WS 2010-11

Models of Computation, an Algorithmic Perspective

Assignment 12 Tue 18.1.2011

This assignment is **due on January 26/28** in your respective tutorial groups. You are allowed (even encouraged) to discuss these problems with your fellow classmates. All submitted work, however, must be *written individually* without consulting someone else's solutions or any other source like the web.

Dynamization. We study a generic method for deriving dynamic data structures from static ones. We use the dictionary problem for a case study.

Static dictionaries only support operation member, dynamic dictionaries also support insertions and deletions.

We know how to sort and we are familiar with binary search in sorted arrays. However, for some reason, we can do binary search only in arrays whose size is a power of two. So, we know how to support the member-operation for sets whose cardinality n is a power of two. The running time of member is $O(\log n)$. The cost of setting up a binary search structure for $n = 2^k$ elements has cost $O(n)$ if the elements are given in sorted order and is $O(n \log n)$ otherwise.

Exercise 1 [Member for arbitrary set size] Let S be a set of size n. Write n in binary, i.e., $n = \sum_i b_i 2^i$ with $b_i \in \{0, 1\}$. Split S into disjoint sets S_0, S_1, ... where S_i has size $b_i 2^i$. For each S_i, we know how to implement member. Implement member for S. What is the running time?

What is the cost of building the structure if S is unsorted, if S is sorted?

Exercise 2 [Inserts] We add insert to the repertoire of operations. Let $n = \sum_i b_i 2^i$. Let j be minimal such that $b_j = 0$. Observe that $1 + \sum_{i<j} 2^i = 2^j$. Take the new element, call it x, and the sets S_0 to S_{j-1}, and build a binary search data structure for $\{x\} \cup (\cup_{i<j} S_i)$.

What is the cost of building this structure? Do you have to sort or can you avoid sorting? Try to use repeated merging.

What is the cost of an insert in the worst case? What is the cost of n inserts into an initially

empty structure?

Exercise 3 [Warm-Up for Delete] Consider a sorted array. Some of the elements are marked as "deleted". Let S be the set of unmarked elements. Realize member for the set of unmarked elements. This exercise has a two-line answer.

Exercise 4 [Deletes] We come to a general solution for deletes. In order to delete an element, perform member and then mark it as "deleted". Keep two counters: the number n of elements in the structure and the number d of elements marked as deleted.

Whenever $d \geq n/2$, do the following: Let S be the set of unmarked elements. Throw your old data structure away and build a data structure for S as in exercise 1. Set n to $n - d$ and d to zero.

What is the cost of rebuilding? Do you have to sort? Express the cost of rebuilding in terms of d.

Exercise 5 [Sequences of Inserts and Deletes] What is the total cost of a sequence of n inserts and deletes? Show that is suffices to charge $O(\log n)$ to each insert and $O(1)$ to each delete.

The technique above is known since the early 80s. See, for example, Section 7.1 of my books from 1984 (you can find them on my homepage). We know apply the technique to cache-oblivious static search trees and see what we get.

Exercise 6 [Cache-Oblivious Dynamic Dictionaries] Trees in van-Emde-Boas layout implement cache-oblivious static dictionaries with I/O-cost $O(\log_B n)$. Apply the machinery above to them. What do you get?

Hint: I think it is a good idea to keep the search structures for S_0, S_1, to S_k in cache, where k is such that the total space required for these structures is a constant fraction of cache size. This will speed up searches and inserts.