Dynamic Interpolation Search Revisited*

Alexis Kaporis[1,2], Christos Makris[1], Spyros Sioutas[1], Athanasios Tsakalidis[1,2], Kostas Tsichlas[1], and Christos Zaroliagis[1,2]

[1] Dept of Computer Eng and Informatics, University of Patras, 26500 Patras, Greece
[2] Computer Technology Institute, N. Kazantzaki Str, Patras University Campus, 26500 Patras, Greece
{kaporis, makri, sioutas, tsak, tsihlas, zaro}@ceid.upatras.gr

Abstract. A new dynamic Interpolation Search (IS) data structure is presented that achieves $O(\log\log n)$ search time with high probability on unknown continuous or even discrete input distributions with measurable probability of key collisions, including power law and Binomial distributions. No such previous result holds for IS when the probability of key collisions is measurable. Moreover, our data structure exhibits $O(1)$ expected search time with high probability for a wide class of input distributions that contains all those for which $o(\log\log n)$ expected search time was previously known.

1 Introduction

The dynamic dictionary search problem is one of the fundamental problems in computer science. In this problem we have to maintain a set of elements subject to insertions and deletions such that given a query element y we can retrieve the largest element in the set smaller or equal to y. Well known search methods use an arbitrary rule to *select* a splitting element and *split* the stored set into two subfiles; in binary search, each recursive split selects as splitting element, in a "blind" manner, the middle (or a close to the middle) element of the current file. Using this technique, known balanced search trees (e.g., (a, b)-trees [11]) support search and update operations in $O(\log n)$ time when storing n elements. In the Pointer Machine (PM) model of computation, the search time cannot be further reduced, since the lower bound of $\Omega(n \log n)$ for sorting n elements would be violated. In the RAM model of computation, which we consider in this work, a lower bound of $\Omega(\sqrt{\frac{\log n}{\log\log n}})$ was proved by Beame and Fich [4]; a data structure achieving this time bound has been presented by Andersson and Thorup [2].

The aforementioned lower bounds can be surpassed if we take into account the input distribution of the keys and consider expected complexities; in this case, the extra knowledge about the probabilistic nature of the keys stored in the file may lead to better selections of splitting elements. The main representative of

* This work was partially supported by the FET Unit of EC (IST priority – 6th FP), under contracts no. IST-2002-001907 (integrated project DELIS) and no. FP6-021235-2 (project ARRIVAL), and by the Action PYTHAGORAS with matching funds from the European Social Fund and the Greek Ministry of Education.

M. Bugliesi et al. (Eds.): ICALP 2006, Part I, LNCS 4051, pp. 382–394, 2006.

these techniques is the method of *Interpolation Search* (IS) introduced by Peterson [21], where the splitting element was selected close to the expected location of the target key. Yao and Yao [28] proved a $\Theta(\log\log n)$ average search time for stored elements that are uniformly distributed. In [9,10,18,19,20] several aspects of IS are described and analyzed. Willard [26] proved the same search time for the extended class of *regular* input distributions. The IS method was recently generalized [5] to non-random input data that possess enough "pseudo-randomness" for effective IS to be applied. The study of dynamic insertions of elements with respect to the uniform distribution and random deletions was initiated in [8,12]. In [8] an implicit data structure was presented supporting insertions and deletions in $O(n^{\epsilon}), \epsilon > 0$, time and IS with expected time $O(\log\log n)$. The structure of [12] has expected insertion time $O(\log n)$, amortized insertion time $O(\log^2 n)$ and it is claimed, without rigorous proof, that it supports IS. Mehlhorn and Tsakalidis [16] demonstrated a novel dynamic version of the IS method, the *Interpolation Search Tree (IST)*, with $O(\log\log n)$ expected search and update time for a larger class than the regular distributions. In particular, they considered μ-random insertions and random deletions[1] by introducing the notion of a (f_1, f_2)-*smooth* probability density μ, in order to control the distribution of the elements in each subinterval dictated by an ID index. Informally, a distribution defined over an interval I is smooth if the probability density over any subinterval of I does not exceed a specific bound, however small this subinterval is (i.e., the distribution does not contain sharp peaks). The class of smooth distributions is a superset of uniform, bounded, and several non-uniform distributions (including the class of regular distributions). The results in [16] hold for $(n^{\alpha}, \sqrt{n})$-smooth densities, where $1/2 \leq \alpha < 1$. Andersson and Mattson [1], generalized and refined the notion of smooth distributions, presenting a variant of the IST called *Augmented Sampled Forest* extending the class of input distributions for which $\Theta(\log\log n)$ search time is expected. In particular, the time complexities of their structure holds for the larger class of $(\frac{n}{(\log\log n)^{1+\epsilon}}, n^{\delta})$-smooth densities, where $\delta \in (0,1), \epsilon > 0$. Moreover, their structure exhibited $o(\log\log n)$ expected search time for some classes of input distributions. Finally in [13], a finger search version of these structures was presented.

The analysis of all the aforementioned IS structures was heavily based on the assumption that the conditional distribution on the subinterval dictated by an arbitrary interpolation step remains unaffected. In particular, in [1,13,16] IS is performed on each node of a tree structure under the assumption that all elements in the subtree dictated by the previous interpolation step remain μ-random.

Our first contribution in this work (Section 2) is to show that the above assumption is valid only when the produced elements are *distinct* (as indeed assumed in [1,9,10,13,16,19,20,21,26,28]), i.e., they are produced under some continuous distribution where the probability of collision is zero; otherwise, it *fails*.

[1] An insertion is μ-random if the key to be inserted is drawn randomly with density function μ; a deletion is random if every key present in the data structure is equally likely to be deleted.