Mehlhorn-Tsakalidis Revisited

Christos Zaroliagis

(Joint work with A. Kaporis, C. Makris, S. Sioutas, A. Tsakalidis, & K. Tsichlas)
Dynamic Interpolation Search

KURT MEHLHORN

Max-Planck-Institut für Informatik and Universität des Saarlandes, Saarlandes, Germany

AND

ATHANASIOS TSAKALIDIS

Computer Technology Institute, Patras, Greece

Abstract. A new data structure called interpolation search tree (IST) is presented which supports interpolation search and insertions and deletions. Amortized insertion and deletion cost is $O(\log n)$. The expected search time in a random file is $O(\log \log n)$. This is not only true for the uniform distribution but for a wide class of probability distributions.

Categories and Subject Descriptors: E.1 [Data Structures]: trees; F.2 [Analysis of Algorithms and Problem Complexity]

General Terms: Algorithms

Additional Key Words and Phrases: Dynamization, interpolation search, search tree
Dynamic Interpolation Search

KURT MEHLHORN

Max-Planck-Institut für Informatik and Universität des Saarlandes, Saarlandes, Germany

AND

ATHANASIOS TSAKALIDIS

Computer Technology Institute, Patras, Greece

Abstract. A new data structure called interpolation search tree (IST) is presented which supports interpolation search and insertions and deletions. Amortized insertion and deletion cost is \(O(\log n)\). The expected search time in a random file is \(O(\log\log n)\). This is not only true for the uniform distribution but for a wide class of probability distributions.

Categories and Subject Descriptors: E.1 [Data Structures]: trees; F.2 [Analysis of Algorithms and Problem Complexity]

General Terms: Algorithms

Additional Key Words and Phrases: Dynamization, interpolation search, search tree
Given set of keys $S = \{X_1, \ldots, X_n\}, X_i \in [a, b] \subseteq \mathbb{R}$

Maintain (under key insertions and deletions) a non-decreasing ordering $P = \{X_{(1)}, \ldots, X_{(n)}\}$ of $S$ such that:

- given a query element $y$
- find largest $X_{(j)} \in P : X_{(j)} \leq y$
Use arbitrary rule to select *splitting* element \( X_{(k)} \in P \) that splits \( P \) into two subsets, and recurse

e.g., in binary search, splitting element = middle element
Use arbitrary rule to select *splitting* element $X_{(k)} \in P$ that splits $P$ into two subsets, and recurse
e.g., in binary search, splitting element $=$ middle element

Balanced search trees (AVL-trees, red-black trees, $(a, b)$-trees, etc):
search & update time: $O(\log n)$
Dynamic Dictionary Search - Classical methods

- Use arbitrary rule to select **splitting** element $X_{(k)} \in P$ that splits $P$ into two subsets, and recurse
  
e.g., in binary search, splitting element = middle element

- Balanced search trees (AVL-trees, red-black trees, $(a, b)$-trees, etc):
  
  search & update time: $O(\log n)$

- Bounds
  
  - optimal for Pointer Machine
  
  - RAM: $\Theta \left( \sqrt{\frac{\log n}{\log \log n}} \right)$ (Andersson & Thorup, 2001)
Interpolation Search (Peterson, 57)

- Select splitting elements by taking advantage of the statistical properties of the keys

- Splitting elements are spread closer to query key $y$
**Interpolation Search** (Peterson, 57)

- Select splitting elements by taking advantage of the statistical properties of the keys

- Splitting elements are spread closer to query key \( y \)

- *Expected search time* (static problem):
**Interpolation Search** (Peterson, 57)

- Select splitting elements by taking advantage of the statistical properties of the keys

- Splitting elements are spread closer to query key $y$

**Expected search time** (static problem):

- $\Theta(\log \log n)$, uniform distribution
  
  (Yao & Yao, 76), (Gonnet, 77), (Perl & Reingold, 77), (Perl, Itai & Avni, 78),
  (Gonnet, Rogers & George, 80)
Interpolation Search (Peterson, 57)

- Select splitting elements by taking advantage of the statistical properties of the keys

- Splitting elements are spread closer to query key \( y \)

Expected search time (static problem):

- \( \Theta(\log \log n) \), uniform distribution
  - (Yao & Yao, 76), (Gonnet, 77), (Perl & Reingold, 77), (Perl, Itai & Avni, 78), (Gonnet, Rogers & George, 80)

- \( \Theta(\log \log n) \), regular (non-uniform) distribution
  - (Willard, 85)
Dynamic Interpolation Search

- Scenario: $\mu$-random insertions, random deletions

- $\mu$ uniform (Frederickson, 83), (Itai, Konheim & Rodeh, 81)
  - search: $O(\log \log n)$ expected time
  - update: $O(n^\varepsilon)$ time, $0 < \varepsilon < 1$
**Dynamic Interpolation Search**

**$\mu(f_1, f_2)$-smooth** *(peak-less or peak-adjusting)*  (Mehlhorn & Tsakalidis, 85)

\[
\Pr \left[ c_2 - \frac{c_3 - c_1}{f_1(n)} \leq X \leq c_2 \mid c_1 \leq X \leq c_3 \right] \leq \frac{\beta f_2(n)}{n}
\]
Dynamic Interpolation Search

\( \mu (f_1, f_2) \)-smooth (peak-less or peak-adjusting) (Mehlhorn & Tsakalidis, 1985)

\[
\Pr \left[ c_2 - \frac{c_3 - c_1}{f_1(n)} \leq X \leq c_2 \mid c_1 \leq X \leq c_3 \right] \leq \frac{\beta f_2(n)}{n}
\]

- smooth \( \supset \{ \text{uniform, regular, bounded, other non-uniform} \}
- \( \rightarrow \) any probability distribution is \( (f_1, \Theta(n)) \)-smooth
Dynamic Interpolation Search

$\mu (f_1, f_2)$-smooth (peak-less or peak-adjusting) (Mehlhorn & Tsakalidis, 85)

$$\Pr \left[ c_2 - \frac{c_3 - c_1}{f_1(n)} \leq X \leq c_2 \mid c_1 \leq X \leq c_3 \right] \leq \frac{\beta f_2(n)}{n}$$

- smooth $\supset \{\text{uniform, regular, bounded, other non-uniform}\}$
  - any probability distribution is $(f_1, \Theta(n))$-smooth
- search: $O(\log \log n)$ expected time

update: \[\begin{cases} O(\log \log n) \text{ expected time} & \text{(Mehlhorn & Tsakalidis, 85)} \\ O(1) \text{ time (position given)} & \text{(Andersson & Mattson, 93)} \end{cases}\]
Static/Dynamic Interpolation Search

Key Assumption

Conditional distribution on the subinterval dictated by an arbitrary interpolation step remains unaffected (i.e., \( \mu \)-random)
Key assumption is valid

- only when elements are *distinct* (indeed assumed in all previous work)
  - produced under some continuous (or discrete) distribution with zero probability of collision
- otherwise, it *fails*
Key assumption is valid

- only when elements are distinct (indeed assumed in all previous work)
  - produced under some continuous (or discrete) distribution with zero probability of collision
- otherwise, it fails

Probabilistic analyses of previous Interpolation Search structures are inapplicable to sequences of non-distinct elements

- produced by discrete probability distributions with measurable (non-zero) probability of key collision
∃ applications where we **must** store duplicates

- Secondary indices in databases
- Searching tables with alphabetic keys (names, dictionary entries, etc)
  - keys follow a non-uniform, discrete distribution and collisions *do occur*
  - Empirical results showed that Interpolation Search has a very poor performance on such data
    - (Perl & Reingold, 77), (Burton & Lewis, 80), (Santorno & Sidney, 85), (Perl & Gabriel, 92)
  - Heuristics; no rigorous performance analysis

- Storing duplicates once: statistical properties are destroyed
New dynamic interpolation search data structure

- search: $O(\log \log n)$ expected time
- update: $O(1)$ time (position given)

Bounds hold with high probability

Analysis

- always valid irrespectively of the distinctness or not of the elements
- applies to $\left(\frac{n}{(\log \log n)^{1+\epsilon}}, n^\delta\right)$-smooth input distributions, $\delta \in (0, 1), \epsilon > 0$

Robust (no apriori knowledge of the input distribution is required)
Modifications to our data structure yield

- $O(1)$ expected search time w.h.p. for a (largest so far) subclass of smooth distributions

- $O(\log \log n)$ expected search time w.h.p. for **Power-law** and **Binomial** distributions

  - Power-law and Binomial: $\exists$ proper choices of $f_1, f_2$ to achieve $O(\log \log n)$ expected search time
Interpolation Search (Mehlhorn & Tsakalidis, 85), (Andersson & Mattson, 93)

First Interpolation Step

Children of the root node:

Root

Root's ID array

<table>
<thead>
<tr>
<th>ID[1]</th>
<th>ID[2]</th>
<th>...</th>
<th>ID[j]</th>
<th>...</th>
<th>ID[I(n)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a + b - a / I(n)</td>
<td>a + 2 b - a / I(n)</td>
<td>a + (j - 1) b - a / I(n)</td>
<td>a + j b - a / I(n)</td>
<td>a + (f1(n) - 1) b - a / I(n)</td>
</tr>
</tbody>
</table>

Root's REP array

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( \mu \) is \((f_1(n), f_2(n)) = (I(n), n/R(n))\)-smooth
  - root has \( R(n) \) children; a root-child has \( R(n/R(n)) \) children, etc.
- **ID** array: partitions \([a, b]\) into \( I(n) \) equal-length parts
- **REP** array: partitions \( P \) into \( R(n) \) equal-sized subsets, each of size \( n/R(n) \)
  
  \[
  \text{REP}[i] \leftrightarrow P_i = \{ X \in P \mid X((i-1)\frac{n}{R(n)}) < X \leq X(i\frac{n}{R(n)}) \}, \quad i = 1, \ldots, R(n)
  \]
First Interpolation Step

Children of the root node:

Root

1  2  ...  v  ...  R(n)

Root’s ID array

ID[1]  ID[2]  ...  ID[j]  ...  ID[I(n)]

a  a + 1 \frac{b-a}{I(n)}  a + 2 \frac{b-a}{I(n)}  a + (j - 1) \frac{b-a}{I(n)}  a + j \frac{b-a}{I(n)}  a + (f_1(n) - 1) \frac{b-a}{I(n)}  b

Root’s REP array


a  a  a  ...  a  ...  a

Query element y \Rightarrow

j = \lfloor \frac{y-a}{b-a} I(n) \rfloor + 1 \Rightarrow l_j = \left[ a + (j - 1) \frac{b-a}{I(n)}, a + j \frac{b-a}{I(n)} \right]

Search for appropriate REP within $l_j$

- $O(1)$ expected number of REPs
- distribution of elements between consecutive REPs: $\mu$-random (smooth)
[a, b] \( \mu \)-random and smooth, \((a', b') \subseteq [a, b]\)

\[
\Pr[X = \lambda \mid a' < X \leq b'] = \frac{\Pr[X = \lambda]}{\Pr[a' < X < b']}, \quad a' < \lambda < b' \tag{1}
\]
Analysis and the Subtle Case

- \([a, b] \mu\text{-random and smooth, } (a', b') \subseteq [a, b]\)

\[
\Pr[X = \lambda \mid a' < X \leq b'] = \frac{\Pr[X = \lambda]}{\Pr[a' < X < b']}, \quad a' < \lambda < b' \quad (1)
\]

- Consider \(X_1, X_2, X_3 \in [a, b]\), and their ordering \(X_{(1)} \leq X_{(2)} \leq X_{(3)}\)

\[
\Pr[X = \lambda \mid X_{(1)} = a' \cap X_{(3)} = b'] = \frac{\Pr[X = \lambda \cap X_{(1)} = a' \cap X_{(3)} = b']}{\Pr[X_{(1)} = a' \cap X_{(3)} = b']} \quad (2)
\]
Analysis and the Subtle Case

- \([a, b] \mu\text{-random and smooth, } (a', b') \subseteq [a, b]\)

  \[
  \Pr[X = \lambda \mid a' < X \leq b'] = \frac{\Pr[X = \lambda]}{\Pr[a' < X < b']}, \quad a' < \lambda < b' \quad (1)
  \]

- Consider \(X_1, X_2, X_3 \in [a, b]\), and their ordering \(X_{(1)} \leq X_{(2)} \leq X_{(3)}\)

  \[
  \Pr[X = \lambda \mid X_{(1)} = a' \cap X_{(3)} = b'] = \frac{\Pr[X = \lambda \cap X_{(1)} = a' \cap X_{(3)} = b']}{\Pr[X_{(1)} = a' \cap X_{(3)} = b']} \quad (2)
  \]

- Is (2) \(\mu\text{-random and smooth}??\)
Analysis and the Subtle Case

**Distinct keys** ($\Pr[\text{key collision}] = 0$)

- Event $E_1 = \{X = \lambda \cap X_{(1)} = a' \cap X_{(3)} = b'\}$ occurs if $\geq 1$ occurs

  \[
  \begin{align*}
  &\{X_2 = \lambda, X_3 = a', X_1 = b'\}, & \{X_1 = \lambda, X_2 = a', X_3 = b'\}, \\
  &\{X_3 = \lambda, X_1 = a', X_2 = b'\}, & \{X_1 = \lambda, X_3 = a', X_2 = b'\}, \\
  &\{X_3 = \lambda, X_2 = a', X_1 = b'\}, & \{X_2 = \lambda, X_1 = a', X_3 = b'\}
  \end{align*}
  \]

- Event $E_2 = \{X_{(1)} = a' \cap X_{(3)} = b'\}$ occurs if $\geq 1$ occurs

  \[
  \begin{align*}
  &\{X_1 = a', X_2 = b', a' < X_3 < b'\}, & \{X_2 = a', X_1 = b', a' < X_3 < b'\}, \\
  &\{X_1 = a', X_3 = b', a' < X_2 < b'\}, & \{X_3 = a', X_1 = b', a' < X_2 < b'\}, \\
  &\{X_2 = a', X_3 = b', a' < X_1 < b'\}, & \{X_3 = a', X_2 = b', a' < X_1 < b'\}
  \end{align*}
  \]

- \[ (2) = \frac{\Pr[E_1]}{\Pr[E_2]} = \frac{6 \Pr[X=\lambda] \Pr[X=a'] \Pr[X=b']}{{6 \Pr[X=a'] \Pr[X=b'] \Pr[a'<X<b']}} = \frac{\Pr[X=\lambda]}{\Pr[a'<X<b']} = (1) \]
Non-distinct keys \( (Pr[\text{key collision}] \neq 0) \)

- Event \( E_1 = \{ X = \lambda \cap X(1) = a' \cap X(3) = b' \} \) occurs if \( \geq 1 \) occurs
  
  \[ \{ X_2 = \lambda, X_3 = a', X_1 = b' \}, \quad \{ X_1 = \lambda, X_2 = a', X_3 = b' \}, \quad \{ X_3 = \lambda, X_1 = a', X_2 = b' \}, \quad \{ X_1 = \lambda, X_3 = a', X_2 = b' \}, \quad \{ X_3 = \lambda, X_2 = a', X_1 = b' \}, \quad \{ X_2 = \lambda, X_1 = a', X_3 = b' \} \]

- Event \( E_2 = \{ X(1) = a' \cap X(3) = b' \} \) occurs if \( \geq 1 \) occurs
  
  \[ \{ X_1 = a', X_2 = b', a' < X_3 < b' \}, \quad \{ X_2 = a', X_1 = b', a' < X_3 < b' \}, \quad \{ X_1 = a', X_3 = b', a' < X_2 < b' \}, \quad \{ X_3 = a', X_1 = b', a' < X_2 < b' \}, \quad \{ X_2 = a', X_3 = b', a' < X_1 < b' \}, \quad \{ X_3 = a', X_2 = b', a' < X_1 < b' \} \]

\[
\{ X_{1,2} = a', X_3 = b' \}, \quad \{ X_{1,2} = b', X_3 = a' \}, \quad \{ X_{1,3} = a', X_2 = b' \}, \quad \{ X_{1,3} = b', X_2 = a' \}, \quad \{ X_{2,3} = a', X_1 = b' \}, \quad \{ X_{2,3} = b', X_1 = a' \}
\]

\[
(2) = \frac{Pr[E_1]}{Pr[E_2]} = \frac{Pr[X=\lambda]}{Pr[X=a']^2 + Pr[a'<X<b']} \neq (1) !!
\]
The New Data Structure

- $n$ elements drawn $\mu$-randomly from $[a, b]$
  $\mu$ is $(f_1, f_2) = (n^\alpha, n^\delta)$-smooth, $\alpha, \delta \in (0, 1)$

**Key idea**

- Forget about REPds
- Partition $[a, b]$ into $f_1(n)$ intervals, each of size $(b - a)/f_1(n)$
- Recurse on each interval (BIN) until $|\text{BIN}| = O(\text{poly log } n)$
1st Layer of $f_1(n)$ bins.

Each bin receives a ball with probability $O\left(\frac{f_2(n)}{n}\right) = O\left(\frac{n^{\delta}}{n}\right)$.

<table>
<thead>
<tr>
<th>$n_1$ balls</th>
<th>$n_2$ balls</th>
<th>...</th>
<th>$n_{j_1}$ balls</th>
<th>...</th>
<th>$n_{f_1(n)}$ balls</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$a + \frac{b-a}{f_1(n)}$</td>
<td>$a + 2\frac{b-a}{f_1(n)}$</td>
<td>...</td>
<td>$a + (j_1 - 1)\frac{b-a}{f_1(n)}$</td>
<td>$a + j_1\frac{b-a}{f_1(n)}$</td>
</tr>
</tbody>
</table>

BIN(1) BIN(2) ... BIN($f_1(n)$)

BIN($j_1$)’s 2nd Layer of $f_1(n_{j_1})$ bins.

Each bin receives a ball with probability $O\left(\frac{f_2(n_{j_1})}{n_{j_1}}\right) = O\left(\frac{n^{\delta^2}}{n}\right)$.

<table>
<thead>
<tr>
<th>$n_{j_1,1}$ balls</th>
<th>$n_{j_1,2}$ balls</th>
<th>...</th>
<th>$n_{j_1,j_2}$ balls</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{j_1}$</td>
<td>$a_{j_1} + \frac{b_{j_1} - a_{j_1}}{f_1(n_{j_1})}$</td>
<td>$a_{j_1} + 2\frac{b_{j_1} - a_{j_1}}{f_1(n_{j_1})}$</td>
<td>...</td>
</tr>
</tbody>
</table>

BIN($j_1$, 1) BIN($j_1$, 2) ... BIN($j_1$, $f_1(n_{j_1})$)

Leaf BIN: $q^*$-heap (Willard,93) $\Rightarrow$ $O(1)$ search & update time
**Lem. 1** The elements of each BIN (subinterval) are $\mu$-randomly distributed.

**Lem. 2** $|\text{BIN}| = f_2(|\text{parent-BIN}|)$ w.h.p.

\[ \Downarrow \]

- 1st layer: $|\text{BIN}| = O(n^\delta)$
  
  \vdots

- $k$-th layer: $|\text{BIN}| = O(n^{\delta^k})$

\[ \Downarrow \]

- Number of layers: $O(\log \log n)$
First layer of $f_1(n)$ BINS.

Each BIN receives a ball with probability $O\left(\frac{f_2(n)}{n}\right) = O\left(\frac{\ln^O(1)n}{n}\right)$.

<table>
<thead>
<tr>
<th></th>
<th>$n_1$ balls</th>
<th>$n_2$ balls</th>
<th>$n_3$ balls</th>
<th>$n_j$ balls</th>
<th>$n_{f_1(n_1)}$ balls</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIN(1)</td>
<td>$a$</td>
<td>$a + \frac{b-a}{f_1(n)}$</td>
<td>$a + 2\frac{b-a}{f_1(n)}$</td>
<td>$\cdots$</td>
<td>$a + (j - 1)\frac{b-a}{f_1(n)}$</td>
</tr>
<tr>
<td>BIN(2)</td>
<td>$b-a$</td>
<td>$b-a$</td>
<td>$b-a$</td>
<td>$\cdots$</td>
<td>$b-a$</td>
</tr>
<tr>
<td>BIN($j$)</td>
<td>$b-a$</td>
<td>$b-a$</td>
<td>$b-a$</td>
<td>$\cdots$</td>
<td>$b-a$</td>
</tr>
<tr>
<td>BIN($f_1(n_1)$)</td>
<td>$a$</td>
<td>$a + \frac{b-a}{f_1(n)}$</td>
<td>$a + 2\frac{b-a}{f_1(n)}$</td>
<td>$\cdots$</td>
<td>$a + (f_1(n) - 1)\frac{b-a}{f_1(n)}$</td>
</tr>
</tbody>
</table>

- $\mu: (f_1(n), f_2(n)) = \left(\frac{n}{g}, \ln^O(1)n\right)$-smooth, $g$: constant
- $|\text{BIN}| = O(\ln^{O(1)}n)$ w.h.p.
- Implement each bin as $q^*$-heap $\implies O(1)$ search time
**Power-law Distribution**

- $I_1$ is dense $\Rightarrow$ van Emde Boas structure, $O(\log \log |I_1|)$ search time
- $I_2$ is sparse $\Rightarrow$ distribution is $(f_1(n), poly \log n)$-smooth
- Similar idea for Binomial
• **New** dynamic interpolation search data structure

• Supports **non-distinct** (duplicate) elements

• Expected search time $O(\log \log n)$ w.h.p. for smooth and other distributions (e.g., power law)

• $O(1)$ expected search time for a subclass (largest so far) of smooth distributions
Thanks

Thank you Thanassi for introducing me (and others) to the intricacies of elementary data structures. Your contributions to CS&E and for establishing it within Greece, your huge contribution in elevating the Dept of Computer Engineering & Informatics in all respects (infrastructure, procedures, etc) within and out of our University. The artistic touch you gave to CEID... your support and friendship.

Wish you all the best for the future.
Thank you Thanassi for
Thank you Thanassi for

- introducing me (and others) to the intricacies of elementary data structures
Thank you Thanassi for

- introducing me (and others) to the intricacies of elementary data structures
- your contributions to CS&E and for establishing it within Greece

Wish you all the best for the future.
Thank you Thanassi for

- introducing me (and others) to the intricacies of elementary data structures
- your contributions to CS&E and for establishing it within Greece
- your huge contribution in elevating the Dept of Computer Engineering & Informatics in all respects (infrastructure, procedures, etc) within and out of our University
Thank you Thanassi for

- introducing me (and others) to the intricacies of elementary data structures
- your contributions to CS&E and for establishing it within Greece
- your huge contribution in elevating the Dept of Computer Engineering & Informatics in all respects (infrastructure, procedures, etc) within and out of our University
- the artistic touch you gave to CEID
Thank you Thanassi for

- introducing me (and others) to the intricacies of elementary data structures
- your contributions to CS&E and for establishing it within Greece
- your huge contribution in elevating the Dept of Computer Engineering & Informatics in all respects (infrastructure, procedures, etc) within and out of our University
- the artistic touch you gave to CEID
- ... your support and friendship
Thanks

Thank you Thanassi for

- introducing me (and others) to the intricacies of elementary data structures
- your contributions to CS&E and for establishing it within Greece
- your huge contribution in elevating the Dept of Computer Engineering & Informatics in all respects (infrastructure, procedures, etc) within and out of our University
- the artistic touch you gave to CEID
- ... your support and friendship
Thank you Thanassi for

- introducing me (and others) to the intricacies of elementary data structures
- your contributions to CS&E and for establishing it within Greece
- your huge contribution in elevating the Dept of Computer Engineering & Informatics in all respects (infrastructure, procedures, etc) within and out of our University
- the artistic touch you gave to CEID
- ... your support and friendship

Wish you all the best for the future