Advances on learned indexing and compression of integer data

Giorgio Vinciguerra
Compressed rank/select dictionaries

- Given a set $A$ of $n$ elements over an integer universe $0, 1, \ldots, u$
  1. Store them in compressed form
  2. Implement $\text{rank}(x)$: number of elements in $A$ which are $\leq x$
  3. Implement $\text{select}(i)$: return the $i$th smallest element in $A$

- Well-studied building block of succinct data structures

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>18</td>
<td>22</td>
<td>40</td>
<td>43</td>
<td>47</td>
<td>53</td>
</tr>
</tbody>
</table>

$\text{rank}(12) = 3$

$\text{select}(7) = 40$
Recap

In ALENEX ’21 and TALG 2022, we showed how to use piecewise linear $\varepsilon$-approximations (PLAs) to build compressed R/S dictionaries.

**Step 1:** map input to (pos, value)

**Step 2:** build a PLA with error $\varepsilon$

**Step 3:** store segments + corrections

Crucial ingredient: minimising the space of the encoding via a PLA that uses a different $\varepsilon$ for different segments

Joint work with Antonio and Paolo
Recap (cont.)

• In ISAAC ‘21 we showed how to combine PLAs with tools to capture repetitiveness

\[
A = \begin{array}{cccccccc}
2 & 3 & 5 & 6 & 13 & 14 & 16 & 17 & 20 & 24 \\
\end{array} \ldots
\]

\[
\text{Gap string: } \begin{array}{cccccccc}
2 & 1 & 2 & 1 & 7 & 1 & 2 & 1 & 3 & 4 \\
\end{array} \ldots
\]

• Introduced and experimented the block-\(\varepsilon\) tree
• Introduced the \(\text{LZ}_\varepsilon\) parsing

Joint work with Paolo and Giovanni
New results (submitted to journal)

1. Gap vs binary entropy inequality

\[ nH_k(\text{gap string}) \leq uH_k(\text{characteristic bitvector}) \]

Joint work with Paolo and Giovanni
New results (submitted to journal)

2. Improved the $\text{LZ}_\varepsilon$ parsing space bounds

For the new phrase we store

- Indexes $i$, $j$, and $r$
- Slope and intercept of the segment
- Array of $j - i + 1$ corrections, $\lceil\log(2\varepsilon + 1)\rceil$ bits each

Joint work with Paolo and Giovanni
New results (submitted to journal)

2. Improved the $\text{LZ}_\varepsilon$ parsing space bounds

$z = \text{number of phrases}$
$t = \text{number of corrections}$

$z \log z + t[\log(2\varepsilon + 1)] + \mathcal{O}\left(z \log \frac{u}{z}\right)$ bits

Term further bounded by $nH_k(\text{gap string}) + o\left(n \log \frac{u}{n}\right)$
New results (submitted to journal)

3. Implemented and experimented the $LZ_\varepsilon$ parsing on real data
   • 15× slower in rank than LA-vector, 5× slower in rank than block-$\varepsilon$ tree
   • Worse in space than either LA-vector or block-$\varepsilon$ tree...
   • ... mainly because $LZ_\varepsilon$ does not adapt to the “degrees of linearity” of the data by varying $\varepsilon$

Joint work with Paolo and Giovanni
New results (submitted to journal)

4. Experimented the \( \text{LZ}_\varepsilon \) and block-\( \varepsilon \) tree on **synthetic data**
   - Generate noisy data around random segments
   - With probability \( p = \frac{1}{4}, \frac{1}{2}, \frac{3}{4} \) create a copy

Joint work with Paolo and Giovanni