Succinct dictionary matching with no slowdown

Djamal Belazzougui, dbelaz@liafa.jussieu.fr

LIAFA, Univ. Paris Diderot - Paris 7
Dictionary matching problem

- Set of $d$ patterns (strings): $S = \{s_1, s_2, \ldots, s_d\}$.

- $\sum_{i=1}^{d} |s_i| = n$ characters from an alphabet of size $\sigma$.

- Queries: text $T \rightarrow$ occurrences of patterns in $S$. 
Aho-Corasick automaton

- Classical solution to dictionary matching $\rightarrow$ Aho-Corasick automaton with $m \leq n + 1$ states.

- Space usage $\rightarrow O(m)$ words of memory.

- Query time $\rightarrow O(|T| + occ)$ for resporting $occ$ occurrences.
Aho-Corasick automaton (Space usage)

- Aho-Corasick automaton → too much space:
  \[ O(m) \text{ words} = O(m \log m) \text{ bits.} \]

- But patterns occupy \( n \log \sigma \) bits only.

- If \( m = \Omega(n) \) and \( \sigma = O(1) \):
  - Automaton uses \( \Omega(n \log n) \) bits.
  - While the patterns occupy \( O(n) \) bits only.
  - The automaton uses \( \log n \) times the space of the patterns.
AC automaton encoded in just
\[ m(\log \sigma + O(1)) + O(d \log(n/d)) \rightarrow O(n \log \sigma) \] bits in the worst case.

Query time still \( O(|T| + \text{occ}) \).

Use three kinds of succinct data structures:

- 2 Succinct indexable dictionaries.
- 2 Succinctly encoded navigable trees.
- 1 Succinctly encoded integer array.
### Related work

| Algorithm   | Query time                                      |
|-------------|-------------------------------------------------|
| HLSTV1      | $O(|T| \log \log(n) + \text{occ})$             |
| HLSTV2      | $O(|T|(\log^\epsilon(n) + \log(d)) + \text{occ})$ |
| TWLY        | $O(|T|\log(d) + \log \sigma) + \text{occ})$    |
| Ours        | $O(|T| + \text{occ})$                          |

**Table:** Query time of succinct indexes dictionary matching solutions

| Algorithm   | Space usage (in bits)                           |
|-------------|-------------------------------------------------|
| HLSTV1      | $O(n \log \sigma)$                            |
| HLSTV2      | $n \log \sigma (1 + o(1)) + O(d \log(n))$     |
| TWLY        | $n \log \sigma (2 + o(1)) + O(d \log(n))$     |
| Ours        | $n(\log \sigma + 3.443) + O(d \log(n/d))$     |

**Table:** Space usage of succinct dictionary matching indexes
Tools: Succinct navigable trees

- A tree of \( n \) nodes \( \rightarrow \) \( n(2 + o(1)) \) bits.
- Many navigation operations in constant time.
- Nodes can be addressed in DFS lexicographic order.
- We need one operation:
  - A node of DFS label \( i \) \( \rightarrow \) the DFS label of its parent.
Given a set of integers $I$, such that $I \subseteq [0, U - 1]$ and $|I| = t$.

- $\text{rank}(x)$:
  - $x \in I \rightarrow$ number of elements in $I$ smaller than $x$ (example $\text{rank}(13) = 3$).
  - $x \notin I \rightarrow -1$ (example $\text{rank}(16) = -1$)
comparison-based machine model $\rightarrow$ sorted table: $t \log U$ bits and rank in $O(\log t)$ time (binary search).

In the RAM (integer) model, a result of RRR02: $t(\log(U/t) + \log_2 e + o(1))$ bits and rank in $O(1)$ time.

Also select$(i)$ in constant time $\rightarrow$ find the element of $S$ of rank $i$. 
Aho-Corasick automaton

- $P$: the set of all prefixes of strings in $S$.
- States in the automaton correspond to a prefix of strings in $P$.
  Number of states is $m = |P|$.
- Three kinds of transitions: next, failure and report.
Each state is represented as a unique number in $[0, m - 1]$.
The number associated with each state represents the suffix-lexicographic order of the prefix corresponding to that state in the set $P$.

Example: for the set $S = \{"ABC","B","BC","CA"\}$. We have $P = \{"A","B","CA","C","BC","ABC"\}$.
Transitions are stored in an indexable dictionary.

A *next* transition labeled with character $c$:

\[
\text{state}(p) \xrightarrow{c} \text{state}(p).
\]

Transition stored as a pair $(c, \text{state}(p))$:

\[
(c, \text{state}(p)) \xrightarrow{} \log \sigma + \log m \text{ bits (bits of } c \text{ + bits of } \text{state}(p)\text{)}.
\]

Each pair corresponds uniquely to one of the prefixes of $P$ (except the empty prefix).
natural order of stored pairs $\rightarrow$ order of the prefixes.

$m$ transitions $\rightarrow m$ pairs of integers from $U = [0..2^{\log \sigma + m} - 1]$.

Dictionary is succinct $\rightarrow$ space:
$m(\log(U/m) + 1.443 + o(1)) = m(\log \sigma + 1.443 + o(1))$ bits.

Next transition:

$state(p)$ to $state(pc) \rightarrow state(pc) = rank((c, state(p))) + 1$.

$state(pc) = -1 \rightarrow$ transition does not exist.
• **Failure transitions** form a tree rooted at state 0 (arrows reversed). The numbers associated with the states correspond to the DFS lexicographic order of the tree.
• **Succinct encoding of tree** $\rightarrow m(2 + o(1))$ bits of space.
• **Failure transition** $\rightarrow$ parent operation on the tree.
Encoding of report transitions

- Same as failure tree, but with \( \leq d \) internal nodes (only elements of \( S \)).
- Compressed encoding of the tree in \( d(\log(m/d) + O(1)) \) bits.
- Report transition \( \rightarrow \) parent operation on report tree.
Open problems

• Can we remove the $3.443m$ term in the space usage.
• Can query time be improved to $O(|T| \log \sigma/w + \text{occ})$, where $w$ is the size of the processor word. This is obviously the best one could hope for.

Practical Performance:

• Query time is asymptotically the same as that of standard Aho-Corasick.
• But constants in BIG-O notation might be much larger.
• It would be interesting to explore the practicality of our result.