Worst-case efficient multiple string matching in the RAM model

Djamal Belazzougui, dbelaz@liafa.jussieu.fr

LIAFA, Univ. Paris Diderot - Paris 7
A single pattern of length $m$ characters from an alphabet of size $\sigma$.

Goal: preprocess the pattern to answer to queries.

Queries: text $T$ of length $n$ $\rightarrow$ all occurrences of the pattern.

Classical solution: $KMP$ automaton $\rightarrow O(m)$ space and $O(n + occ)$ query time for reporting $occ$ occurrences.
Set of $d$ patterns (strings): $S = \{s_1, s_2, \ldots, s_d\}$.

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

Queries: text $T$ $\rightarrow$ occurrences of patterns in $S$.

Classical solution $AC$ automaton $\rightarrow O(m)$ space and $O(n + occ)$ time for reporting $occ$ occurrences.
The computer operates on words of size \( w \) (usually \( w = 32 \) or \( w = 64 \) bits).

Usual arithmetic and logic operations take \( O(1) \) time.

Each character of the text is encoded using \( \log \sigma \) bits.

We can read the \( \Theta(\frac{w}{\log \sigma}) \) characters at a time.

The best possible query time \( O(n \log \sigma/w + \text{occ}) \). \( \rightarrow \) a factor \( \Theta(w) \) faster than AC and KMP for constant sized alphabets (e.g. DNA).

Can we achieve that query time?
New result

- **Single string matching:**
  - Space usage $O(m)$ (same as $KMP$).
  - Query time $O(n(1/m + \log \sigma/w) + occ)$.
  - For $m \geq w/\log \sigma$ the query time is optimal $O(n \log \sigma/w + occ)$.

- **Multiple string matching:**
  - Space usage $O(m)$ (same as $AC$).
  - Query time $O(n((\log d + \log y + \log \log m)/y + \log \sigma/w) + occ)$.
  - Where $y$ is the length of the shortest pattern.
  - For $y \geq w(\log w + \log d)/\log \sigma$ the query time is optimal $O(n \log \sigma/w + occ)$. 

Djamal Belazzougui, dbelaz@liafa.jussieu.fr
$P$: the set of all prefixes of strings in $S$.

States in the automaton correspond to a prefixes in $P$. Number of states is $m = |P|$.

Three kinds of transitions: next, failure and report.
Order $P$ according to suffix-lexicographic order \(\rightarrow\) strings compared right-to-left instead of left-to-right.

Give each state a unique number \(\rightarrow\) 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 $P = \{"\" = 0, "A" = 1, "CA" = 2, "B" = 3, "AB" = 4, "C" = 5, "BC" = 6, "ABC" = 7\}$. 
A fixed set $P$ of $n$ strings and a query string $x$.

- Longest prefix matching $\rightarrow$ find the longest $y \in P$ such that $y$ is prefix of $x$.

- Longest suffix matching $\rightarrow$ find the longest $y \in P$ such that $y$ is suffix of $x$.

- Same solution for both problems $\rightarrow$ Combination of string B-tree, suffix array and LCP array.

- Query time $O(|x| \log \sigma/w + \log n)$.
New result (Basic Geometric tools)

- **1D stabbing problem** → A query point $x \in U$ and (fixed) set of $n$ intervals $S \subset U$, return the tightest interval $y \in S$ such that $x \in y$.

- **2D stabbing problem**: A query for a point $x \in U^2$ over a fixed set of $n$ rectangles → all occ rectangles enclosing $x$.

1D stabbing → y-fast trie in time $O(\log \log |U|)$.

2D stabbing problem → Chazelle’s solution (1986) $O(\log n + occ)$ time and $O(n)$ space.
New result (Basic Geometric tools)

Figure: 2D stabbing: a query for the blue point reports the blue rectangles
Read the text $T$ in blocks of $b = w/\log \sigma$ characters.
At step $i$ read $q = T[ib, (i + 1)b]$.
Current state represented by a number $x$.
Report occurrences using 2D stabbing and longest suffix matching.
Do transitions on $b$ characters $\rightarrow$ minimal perfect hashing, 1D stabbing and longest prefix matching.
Do a longest **suffix** matching query on string $q$ relative to set $P$.

Convert the returned suffix into a number $y$.

Finally do a 2D stabbing query on the point $(x, y)$ → all occurrences ending at positions in on $[ib, (i + 1)b]$. 
New result (transitions)

- Convert string $q$ into a number $y$ using MPHF.
- Concatenate the bits of $y$ with bits of $x$ ($y$ as MSB).
- Do a 1D stabbing query for the number $xy$.
- Query successful $\rightarrow$ next state corresponding to a prefix of length $\geq b$.
- Query failed $\rightarrow$ do a longest prefix matching $\rightarrow$ next state corresponding to a prefix of length $< b$. 
We have proposed a new solution for single and multiple string matching problems in the RAM model.

Get the optimal query time $O(m \frac{\log \sigma}{w} + occ)$ when pattern(s) sufficiently long.

Open question : can we achieve the optimal query time for any pattern length?