String Matching with Involutions

Florin Manea

Challenges in Combinatorics on Words – April 2013
Fields Institute, Toronto

String matching

Given two words T (text) and P (pattern), find all occurrences of P in T.

String matching

Given two words T (text) and P (pattern), find all occurrences of P in T.

$$P = acgttgcacg$$
$$T = atatatata\,acgttgcacg\,ttgcacg\,aaaaaaacgttgcacg\,aataat\,acgttgcacg$$
$$acacacacaacgttgcacg\,aaaaaaagc\,aag\,gt\,cg\,aataat\,acgttgcacg\,tttttt$$

String matching

Given two words T (text) and P (pattern), find all occurrences of P in T.

$$P = acgttgcacg$$
$$T = atatatata\,acgttgc\,acg\,ttgcacg\,aaaaaacgttgcacg\,aataat\,acgttgcacg$$
$$acacacacaacgttgcacg\,aaaaaagc\,aag\,gt\,cg\,aataat\,acgttgcacg\,tttttt$$

String matching

Given two words T (text) and P (pattern), find all occurrences of P in T.

$$
\begin{aligned}
P &= acgttgcacg \\
T &= atatatata\,acgttgcacgttgcacg\,aaaaaa\,acgttgcacg\,aataat\,acgttgcacg \\
&\quad acacacaca\,acgttgcacg\,aaaaaaagcaaggtcg\,aataat\,acgttgcacg\,tttttt
\end{aligned}
$$

String matching

Given two words T (text) and P (pattern), find all occurrences of P in T.

$$
\begin{aligned}
P &= acgttgcacg \\
T &= atatatata\,acgttgc\,acg\,ttgcacg\,aaaaaa\,acgttgcacg\,aataat\,acgttgcacg \\
  &\quad acacacaca\,acgttgcacg\,aaaaaaagc\,aag\,gtcg\,aataat\,acgttgcacg\,tttttt
\end{aligned}
$$

Solution: $\mathcal{O}(|T| + |P|)$, e.g., the Knuth-Morris-Pratt algorithm.

String matching with involutions

Antimorphic involution $f : V^* \to V^*$: f-mirroring.
$[f(w) = f(w[n])f(w[n-1]) \cdots f(w[1])$, $f^2 = Id]$.

String matching with involutions

Antimorphic involution $f : V^* \to V^*$: f-mirroring.
$[f(w) = f(w[n])f(w[n-1]) \cdots f(w[1])$, $f^2 = Id]$.

Given T and P and an antimorphic involution $f : V^* \to V^*$, find all factors P' of T obtained by non-overlapping f-mirrorings from P.

String matching with involutions

Antimorphic involution $f : V^* \rightarrow V^*$: f-mirroring.
$[f(w) = f(w[n])f(w[n-1]) \cdots f(w[1]),\ f^2 = Id]$.

Given T and P and an antimorphic involution $f : V^* \rightarrow V^*$, find all factors P' of T obtained by non-overlapping f-mirrorings from P.

$$\begin{array}{lcl} P & = & acgttgcacg \\ f & : & f(a) = a, f(c) = c, f(g) = g, f(t) = t \\ T & = & atatatata\,acgttgc\,acg\,ttgcacg\,aaaaaa\,acgttgcacg\,aataat\,acgttgcacg \\ & & acacacaca\,acgttgcacg\,aaaaaagcatacgtcg\,aataat\,acgacgttcg\,tttttt \end{array}$$

String matching with involutions

Antimorphic involution $f : V^* \to V^*$: f-mirroring.
$[f(w) = f(w[n])f(w[n-1]) \cdots f(w[1]), f^2 = Id]$.

Given T and P and an antimorphic involution $f : V^* \to V^*$, find all factors P' of T obtained by non-overlapping f-mirrorings from P.

$$\begin{aligned} P &= acgttgcacg \\ f &: f(a) = a, f(c) = c, f(g) = g, f(t) = t \\ T &= atatatata\,acgttgc\,acg\,ttgcacg\,aaaaaa\,acgttgcacg\,aataat\,acgttgcacg \\ &\quad acacacaca\,acgttgcacg\,aaaaaa\,gcat\,acgt\,cg\,aataat\,acg\,acgtt\,cg\,tttttt \end{aligned}$$

String matching with involutions

Antimorphic involution $f : V^* \rightarrow V^*$: f-mirroring.
$[f(w) = f(w[n])f(w[n-1]) \cdots f(w[1]), \; f^2 = Id]$.

Given T and P and an antimorphic involution $f : V^* \rightarrow V^*$, find all factors P' of T obtained by non-overlapping f-mirrorings from P.

$$\begin{array}{lcl} P & = & acgttgcacg \\ f & : & f(a) = a, f(c) = c, f(g) = g, f(t) = t \\ T & = & atatatata\,acgttgcacg\,ttgcacg\,aaaaaa\,acgttgcacg\,aataat\,acgttgcacg \\ & & acacacaca\,acgttgcacg\,aaaaaa\,gcatacgtcg\,aataat\,acgacgttcg\,tttttt \end{array}$$

$$\begin{array}{lcl} P & = & acgttgcacg \\ f & : & f(a) = t, f(c) = g, f(g) = c, f(t) = a \\ T & = & atatatata\,acgttgcacg\,tcgcacg\,aaaaaa\,acgttgcacg\,aataat\,acgttgcacg \\ & & acacacaca\,acgttgcacg\,aaaaaacgttagcaacg\,aataat\,acgtgcaacg\,tttttt \end{array}$$

String matching with involutions

Antimorphic involution $f : V^* \to V^*$: f-mirroring.
$[f(w) = f(w[n])f(w[n-1]) \cdots f(w[1]), \; f^2 = Id]$.

Given T and P and an antimorphic involution $f : V^* \to V^*$, find all factors P' of T obtained by non-overlapping f-mirrorings from P.

$$
\begin{aligned}
P &= acgttgcacg \\
f &: f(a) = a, f(c) = c, f(g) = g, f(t) = t \\
T &= atatatata\,acgttgc\,acg\,ttgcacg\,aaaaaa\,acgttgcacg\,aataat\,acgttgcacg \\
&\quad acacacaca\,acgttgcacg\,aaaaaa\,gcatacgtcg\,aataat\,acgacgttcg\,tttttt
\end{aligned}
$$

$$
\begin{aligned}
P &= acgttgcacg \\
f &: f(a) = t, f(c) = g, f(g) = c, f(t) = a \\
T &= atatatata\,acgttgcacg\,tcgcacg\,aaaaaa\,acgttgcacg\,aataat\,acgttgcacg \\
&\quad acacacaca\,acgttgcacg\,aaaaaa\,cgt\,t\,agca\,acg\,aataat\,acg\,tgcaa\,cg\,tttttt
\end{aligned}
$$

Why string matching with involutions?

- Approximate string matching: find all the factors of T obtained from P by a series of simple operations (e.g., edit operations).

Why string matching with involutions?

- Approximate string matching: find all the factors of T obtained from P by a series of simple operations (e.g., edit operations).
- Bio-inspired operations: affect the pattern on a larger scale, e.g., mirroring of factors, translocations, etc.
  [Cantone, Cristofaro, Faro, Giaquinta, Grabowski, 2009 - 2011]: string matching with rotations and translocations,

Why string matching with involutions?

- Approximate string matching: find all the factors of T obtained from P by a series of simple operations (e.g., edit operations).
- Bio-inspired operations: affect the pattern on a larger scale, e.g., mirroring of factors, translocations, etc.
  [Cantone, Cristofaro, Faro, Giaquinta, Grabowski, 2009 - 2011]: string matching with rotations and translocations,
  [Czeizler, Czeizler, Kari, Seki, 2008 - 2011]: combinatorics on words for repetitions with involutions: $xf(x)xxf(x)\ldots$,

Why string matching with involutions?

- Approximate string matching: find all the factors of T obtained from P by a series of simple operations (e.g., edit operations).
- Bio-inspired operations: affect the pattern on a larger scale, e.g., mirroring of factors, translocations, etc.
  [Cantone, Cristofaro, Faro, Giaquinta, Grabowski, 2009 - 2011]: string matching with rotations and translocations,
  [Czeizler, Czeizler, Kari, Seki, 2008 - 2011]: combinatorics on words for repetitions with involutions: $xf(x)xxf(x)\ldots$,
  [Gawrychowski, Manea, Müller, Mercaș, Nowotka, 2012 - 2013]: algorithmics and combinatorics on words for general pseudo-repetitions.

Known results

$|T| = n, |P| = m$

- Mirroring: $\mathcal{O}(nm)$ time in the worst case, $\mathcal{O}(m^2)$ space complexity [Cantone et al., CPM 2011].

Known results

$|T| = n, |P| = m$

- Mirroring: $\mathcal{O}(nm)$ time in the worst case, $\mathcal{O}(m^2)$ space complexity [Cantone et al., CPM 2011].
- Translocations are allowed: $\mathcal{O}(nm^2)$ time in the worst case, $\mathcal{O}(m)$ space, $\mathcal{O}(n)$ average time (subject to some artificial restriction). [Grabowski et al., Inf. Proc. Lett. 2011]

Known results

$|T| = n, |P| = m$

- Mirroring: $\mathcal{O}(nm)$ time in the worst case, $\mathcal{O}(m^2)$ space complexity [Cantone et al., CPM 2011].
- Translocations are allowed: $\mathcal{O}(nm^2)$ time in the worst case, $\mathcal{O}(m)$ space, $\mathcal{O}(n)$ average time (subject to some artificial restriction). [Grabowski et al., Inf. Proc. Lett. 2011]
- Open problem: linear average time, with $\mathcal{O}(nm)$ or better time in worst case, $\mathcal{O}(m^2)$ or better space complexity. [Cantone et al., CPM 2011].

(our) Latest Results:

- Antimorphic involutions: generalized mirroring.

(our) Latest Results:

- Antimorphic involutions: generalized mirroring.
- Novel (simpler) strategy: greedy (but with complex data structures) vs. dynamic programming.

(our) Latest Results:

- Antimorphic involutions: generalized mirroring.
- Novel (simpler) strategy: greedy (but with complex data structures) vs. dynamic programming.
- $\mathcal{O}(nm)$ worst case time complexity, $\mathcal{O}(m)$ space complexity.

(our) Latest Results:

- Antimorphic involutions: generalized mirroring.
- Novel (simpler) strategy: greedy (but with complex data structures) vs. dynamic programming.
- $\mathcal{O}(nm)$ worst case time complexity, $\mathcal{O}(m)$ space complexity.
- $\mathcal{O}(n)$ average time (subject to some simple restrictions on the input alphabet, depending on the involution).

(our) Latest Results:

- Antimorphic involutions: generalized mirroring.
- Novel (simpler) strategy: greedy (but with complex data structures) vs. dynamic programming.
- $\mathcal{O}(nm)$ worst case time complexity, $\mathcal{O}(m)$ space complexity.
- $\mathcal{O}(n)$ average time (subject to some simple restrictions on the input alphabet, depending on the involution).
- Online algorithm.

(our) Latest Results:

- Antimorphic involutions: generalized mirroring.
- Novel (simpler) strategy: greedy (but with complex data structures) vs. dynamic programming.
- $\mathcal{O}(nm)$ worst case time complexity, $\mathcal{O}(m)$ space complexity.
- $\mathcal{O}(n)$ average time (subject to some simple restrictions on the input alphabet, depending on the involution).
- Online algorithm.
- Open problems: better complexities (for what kind of alphabets?)

(our) Latest Results:

- Antimorphic involutions: generalized mirroring.
- Novel (simpler) strategy: greedy (but with complex data structures) vs. dynamic programming.
- $\mathcal{O}(nm)$ worst case time complexity, $\mathcal{O}(m)$ space complexity.
- $\mathcal{O}(n)$ average time (subject to some simple restrictions on the input alphabet, depending on the involution).
- Online algorithm.
- Open problems: better complexities (for what kind of alphabets?), use also translocations

(our) Latest Results:

- Antimorphic involutions: generalized mirroring.
- Novel (simpler) strategy: greedy (but with complex data structures) vs. dynamic programming.
- $\mathcal{O}(nm)$ worst case time complexity, $\mathcal{O}(m)$ space complexity.
- $\mathcal{O}(n)$ average time (subject to some simple restrictions on the input alphabet, depending on the involution).
- Online algorithm.
- Open problems: better complexities (for what kind of alphabets?), use also translocations, simpler solutions.