Repeats and Palindromes: an Overview

Sara H. Geizhals
Advisor: Dina Sokol
The Graduate Center

30 July 2014
Introductory Concepts
   KMP
   Suffix Tree

Tandem Repeats
   Exact Tandem Repeat
   Approximate Tandem Repeat
      Group 1: neighboring - k-repetition
      Group 1: neighboring - k-run
      Group 1: neighboring - k-edit repeat
      Group 2: consensus - consensus model
      Group 2: consensus - k-MAR

Related Research

Palindromes
   Linear-time Algorithm #1 for (Exact) Palindromes
   Linear-time Algorithm #2 for (Exact) Palindromes
   Variations of Palindromes
   Approximate Palindromes in RLE-compressed Data

Research Directions
Introductory Concepts

KMP
Suffix Tree

Tandem Repeats
Exact Tandem Repeat
Approximate Tandem Repeat
  Group 1: neighboring - k-repetition
  Group 1: neighboring - k-run
  Group 1: neighboring - k-edit repeat
  Group 2: consensus - consensus model
  Group 2: consensus - k-MAR

Related Research

Palindromes
Linear-time Algorithm #1 for (Exact) Palindromes
Linear-time Algorithm #2 for (Exact) Palindromes
Variations of Palindromes
Approximate Palindromes in RLE-compressed Data

Research Directions
Introductory Concepts

KMP

Suffix Tree

Tandem Repeats

Exact Tandem Repeat

Approximate Tandem Repeat

- Group 1: neighboring - k-repetition
- Group 1: neighboring - k-run
- Group 1: neighboring - k-edit repeat
- Group 2: consensus - consensus model
- Group 2: consensus - k-MAR

Related Research

Palindromes

- Linear-time Algorithm #1 for (Exact) Palindromes
- Linear-time Algorithm #2 for (Exact) Palindromes
- Variations of Palindromes
- Approximate Palindromes in RLE-compressed Data

Research Directions
Pattern matching problem:

- Let $\Sigma$ be an alphabet.
- **INPUT:**
  - string text, of length $n$: $T = t_1 t_2 \ldots t_n$
  - string pattern, of length $m$: $P = p_1 p_2 \ldots p_m$
  - $t_i, p_i \in \Sigma$
- **OUTPUT:** all positions $i$ in $T$, where there is an occurrence of $P$, i.e., $T[i + k] = P[k], 1 \leq k \leq m$
- Example: $P = ababc$ in $T = caababccdddddababccc$
- Example: $P = aba$ in $T = cabaabaddababac$
Naive solution - $O(nm)$: since each text position is a possible pattern start, compare each text position with the pattern.
KMP

- Naive solution - $O(nm)$: since each text position is a possible pattern start, compare each text position with the pattern.
- Better solution: KMP (Knuth, Morris, and Pratt, 1977)
Stage 1: $O(m)$ to construct KMP automaton for $P$

- Each of $m + 1$ nodes has success arc (to character on right) and failure arc (to border)
  - $Border$ of $X =$ longest substring that is both proper prefix and proper suffix of $X$
  - Example: $abab$’s border is $ab$, $ababa$’s border is $aba$, and $ababc$’s border is empty string
Stage 2: \( O(n) \) to go through characters of \( T \) to look for \( P \)

- If character in \( T \) matches first character of \( P \), algorithm starts at first node of automaton
- For each character match, automaton follows success arc
- For each character mismatch, automaton follows failure arc
- If reaches last node of automaton, then \( P \) is in \( T \)
KMP

Example: $P = ababc$, $T = cabab \ldots$

If $T$’s next character is:

- $c$ - success arc to node 5, and $P$ is in $T$
- $a$ - failure arc to node 2 (border of $abab$)
- anything else
Introductory Concepts

KMP

Suffix Tree

Tandem Repeats

Exact Tandem Repeat

Approximate Tandem Repeat
  Group 1: neighboring - k-repetition
  Group 1: neighboring - k-run
  Group 1: neighboring - k-edit repeat
  Group 2: consensus - consensus model
  Group 2: consensus - k-MAR

Related Research

Palindromes

Linear-time Algorithm #1 for (Exact) Palindromes
Linear-time Algorithm #2 for (Exact) Palindromes
Variations of Palindromes
Approximate Palindromes in RLE-compressed Data

Research Directions
Suffix Tree

Indexing problem:

- Same as pattern matching problem, except that searches for any number of pattern strings in one text string
- Therefore, want queries to be answered in time proportional to (short) $P$s, rather than (long) $T$
Stage 1 of solution for indexing problem: construct suffix tree (ST), which is trie of all suffixes of $T$

- $O(n)$ construction, when alphabet size is constant, as per Weiner (1973), McCreight (1976), and Ukkonen (1995)
Suffix Tree

Stage 2 of solution for indexing problem: \( O(m) \) to decide whether \( P \) is in \( T \):

- Start at root
Stage 2 of solution for indexing problem: $O(m)$ to decide whether $P$ is in $T$:

- Start at root
- Use characters of $P$ to follow paths of ST
Stage 2 of solution for indexing problem:
$O(m)$ to decide whether $P$ is in $T$:

- Start at root
- Use characters of $P$ to follow paths of ST
- If gets stuck, then $P$ is not in $T$; if does not get stuck, then $P$ is in $T$
$T = xabxac$

- $P = xab$ is in $T$
- $P = xaa$ is not in $T$

**Figure:** ST for $T$ of $xabxac$
Use Generalized Suffix Tree (GST) to index several texts.

Numbers at leaf = text, starting position of suffix in text.

Figure: GST for $T_1 = xabxa$ and $T_2 = babxba$
Suffix Tree

- LCP ($v, w$) = length of longest common prefix between strings $v$ and $w$
LCP \((\nu, \omega)\) = length of longest common prefix between strings \(\nu\) and \(\omega\)

LCA (lowest common ancestor) of tree nodes \(\nu\) and \(\omega\) = lowest node with \(\nu\) and \(\omega\) as descendants
LCP \((v, w)\) = length of longest common prefix between strings \(v\) and \(w\)

LCA (lowest common ancestor) of tree nodes \(v\) and \(w\) = lowest node with \(v\) and \(w\) as descendants

Harel and Tarjan (1984): preprocess tree with \(n\) nodes in \(O(n)\) time; following that, can answer LCA queries in constant time
LCP \((v, w)\) = length of longest common prefix between strings \(v\) and \(w\)

LCA (lowest common ancestor) of tree nodes \(v\) and \(w\) = lowest node with \(v\) and \(w\) as descendants

Harel and Tarjan (1984): preprocess tree with \(n\) nodes in \(O(n)\) time; following that, can answer LCA queries in constant time

Solution to LCP problem on strings is same as solution to LCA problem on ST (to be shown)
LCP \((v, w)\) = length of longest common prefix between strings \(v\) and \(w\)

LCA (lowest common ancestor) of tree nodes \(v\) and \(w\) = lowest node with \(v\) and \(w\) as descendants

Harel and Tarjan (1984): preprocess tree with \(n\) nodes in \(O(n)\) time; following that, can answer LCA queries in constant time

Solution to LCP problem on strings is same as solution to LCA problem on ST (to be shown)

Therefore, ST solves LCP with the same runtime
Solution to LCP problem on strings is same as solution to LCA problem on ST

- LCA (6, 9) = 10; LCP (babba$, bba$) = b
- LCA (1, 2) = 3; LCP (ababba$, abba$) = ab
- LCA (5, 9) = root; LCP (a$, bba$) = empty string

Figure: ST for T of ababba$
Suffix Tree

Solution to LCP problem on strings is same as solution to LCA problem on ST

- LCA (6, 9) = 10;
  LCP (babba$, bba$) = b

- LCA (1, 2) = 3;
  LCP (ababba$, abba$) = ab

- LCA (5, 9) = root;
  LCP (a$, bba$) = empty string
Solution to LCP problem on strings is same as solution to LCA problem on ST

- LCA (6, 9) = 10;
  LCP (babba$, bba$) = b
- LCA (1, 2) = 3;
  LCP (ababba$, abba$) = ab
- LCA (5, 9) = root;
  LCP (a$, bba$) = empty string

Figure: ST for T of $ababba$
Introductory Concepts
KMP
Suffix Tree

Tandem Repeats

Exact Tandem Repeat

Approximate Tandem Repeat
- Group 1: neighboring - k-repetition
- Group 1: neighboring - k-run
- Group 1: neighboring - k-edit repeat
- Group 2: consensus - consensus model
- Group 2: consensus - k-MAR

Related Research

Palindromes
- Linear-time Algorithm #1 for (Exact) Palindromes
- Linear-time Algorithm #2 for (Exact) Palindromes
- Variations of Palindromes
- Approximate Palindromes in RLE-compressed Data

Research Directions
Tandem Repeats

- DNA contains consecutive copies of pattern, e.g. `cagcagcag`
- Known as *tandem repeat* (TR)
- Identification of TRs is useful
  - Linked to over thirty hereditary disorders in humans
  - Used as genetic markers in DNA fingerprinting, mapping genes, comparative genomics, and evolution studies
  - Used in population studies and conservation biology
Introductory Concepts

KMP
Suffix Tree

Tandem Repeats

Exact Tandem Repeat

Approximate Tandem Repeat
  Group 1: neighboring - k-repetition
  Group 1: neighboring - k-run
  Group 1: neighboring - k-edit repeat
  Group 2: consensus - consensus model
  Group 2: consensus - k-MAR

Related Research

Palindromes

Linear-time Algorithm #1 for (Exact) Palindromes
Linear-time Algorithm #2 for (Exact) Palindromes
Variations of Palindromes
Approximate Palindromes in RLE-compressed Data

Research Directions
A string $r = a_1 \ldots a_n$ is *periodic* if there exists some $p$ such that $a_i = a_{i+p}$ for all $i$, when $1 \leq i, i + p \leq n$. 
A string $r = a_1 \ldots a_n$ is *periodic* if there exists some $p$ such that $a_i = a_{i+p}$ for all $i$, when $1 \leq i, i+p \leq n$

$r = u^h u'$

$h$ occurrences of $u$   $(\text{possibly null})$ prefix of $u$
A string $r = a_1 \ldots a_n$ is periodic if there exists some $p$ such that $a_i = a_{i+p}$ for all $i$, when $1 \leq i, i+p \leq n$.

$\Rightarrow r = u^h u'$

- $u$ or $|u| = \text{period of } r$
A string $r = a_1 \ldots a_n$ is periodic if there exists some $p$ such that $a_i = a_{i+p}$ for all $i$, when $1 \leq i, i+p \leq n$.

$r = u^h u'$

- $u$ or $|u|$ = period of $r$
- $abc \ abc \ abc \ abc \ abc$ vs. $abcabc \ abcabc$
  - $h = 4$, $u = abc = 3$ vs. $h = 2$, $u = abcabc = 6$
Exact Tandem Repeat

- **Primitive**
  - If not periodic: e.g., $abcd$
  - If periodic: last period is partial - e.g., $abcdabcdab$
Exact Tandem Repeat

- **Primitive**
  - If not periodic: e.g., \textit{abcd}
  - If periodic: last period is partial - e.g., \textit{abcdabcdab}

- \textit{The period} = the primitive period
  - \textit{abc abc abc abc} - not \textit{abcabc abcabc}
    - \( h = 4, u = abc = 3 \) - not \( h = 2, u = abcabc = 6 \)
Exact Tandem Repeat

- **Primitive**
  - If not periodic: e.g., *abcd*
  - If periodic: last period is partial - e.g., *abcdabcdab*
- *The period = the primitive period*
  - *abc abc abc abc* - not *abcabc abcabc*
    - *h = 4, u = abc = 3* - not *h = 2, u = abcabc = 6*
- Periodic string also called *repetition or exact tandem repeat (ETR) - a repeat*, for short.
Exact Tandem Repeat

- $r = caaaad$ or $cababababad$
  - Periodic in all characters but first and last
Exact Tandem Repeat

- \( r = caaaad \) or \( cababababad \)
  - Periodic in all characters but first and last
- Search \( r \) for repeats - in particular, for maximal repeats = period cannot be extended right or left by even one character
  - \( r = caaaad \):
    - Repeat \( aaaa \) is maximal
    - Repeats \( aa \) and \( aaa \) are not maximal
- \( The \) repeat = the maximal repeat
Exact Tandem Repeat

- \( r = caaaaad \) or \( cababababad \)
  - Periodic in all characters but first and last
- Search \( r \) for repeats - in particular, for \textit{maximal repeats} = period cannot be extended right or left by even one character
  - \( r = caaaaad \):
    - repeat \( aaaa \) is maximal
    - repeats \( aa \) and \( aaa \) are not maximal
  - \textit{The} repeat = the maximal repeat
- Algorithms to find repeats in string \( r \) of length \( n \):
  - \( O(n \log n) \) - Crochemore (1981), Apostolico (1983), Main and Lorentz (1984)
  - \( O(n) \) - KK = Kolpakov and Kucherov (1999)
Introductory Concepts
  KMP
  Suffix Tree

Tandem Repeats
  Exact Tandem Repeat
  Approximate Tandem Repeat
    Group 1: neighboring - k-repetition
    Group 1: neighboring - k-run
    Group 1: neighboring - k-edit repeat
    Group 2: consensus - consensus model
    Group 2: consensus - k-MAR

Related Research

Palindromes
  Linear-time Algorithm #1 for (Exact) Palindromes
  Linear-time Algorithm #2 for (Exact) Palindromes
  Variations of Palindromes
  Approximate Palindromes in RLE-compressed Data

Research Directions
Approximate Tandem Repeat

- DNA has mutations, translocations, reversals, etc.
- *Approximate tandem repeat* or ATR
- Metrics include:
  - Hamming distance - *mismatch*
    - one character substituted by another character
    - 1 mismatch between *abc* and *abd*
  - Edit distance - *error*
    - one character substituted by another character or one character inserted or deleted
    - 2 errors between *abc* and *abdd*
Approximate Tandem Repeat

Search for approximate tandem repeat

Types:

- **Group 1: neighboring** - $k$-repetition, $k$-run, $k$-edit repeat
  - Each occurrence is similar to adjacent (or neighboring) occurrences

- **Group 2: consensus** - consensus model, $k$-MAR
  - Each occurrence is similar to consensus string
Definition

(KK, 2003) A string \( r[1\ldots n] \) is called a \textit{k-repetition} of period \( p \), \( p \leq n/2 \), iff the Hamming distance of \( r[1\ldots n - p] \) and \( r[p+1 \ldots n] \) is \( \leq k \).

- \textit{ataa atta ctta ct} is two-repetition, of period four (total of two mismatches, when comparing each occurrence to next occurrence)
Group 1: neighboring - k-run

Definition

(KK, 2003) A string $r[1 \ldots n]$ is called a $k$-run, of period $p$, $p \leq n/2$, iff for every $i \in [1 \ldots n - 2p + 1]$, the Hamming distance of $r[i \ldots i + p - 1]$ and $r[i + p \ldots i + 2p - 1] \leq k$.

One-run, of period four (up to one mismatch, when comparing each occurrence to next occurrence):

- $ataa\ atta\ ctta\ ggta\ ag$ - four mismatches
- $ataa\ atta\ atta\ atta\ at$ - one mismatch
- Total number of mismatches unbounded
Group 1: neighboring - k-edit repeat

Definition
(Sokol 2007, Sokol 2013) A string $r$ is called a $k$-edit repeat if it can be partitioned into consecutive substrings, $r = u''u_1u_2...u_lu'$, $l \geq 2$, such that the sum of edit distance errors of each occurrence with its neighbor does not exceed $k$.

- $k$-edit repeat = $k$-repetition, except that uses edit distance
- $caagct ccagct ccgc$ is two-edit repeat
  - $ccgc$ is partial period, so it is not counted as deletion of $t$
  - with edit distance, period size is not fixed and so it not given
Definition
(Sagot and Myers, 1998) Each occurrence is called a *wagon*, and the sequence of wagons is a *train*. A *prefix model* is when the total number of edit distance errors, from each wagon to the consensus string, does not exceed $k$. Jumps of a certain size are allowed (a jump is from the left of one wagon to the left of the next wagon). A *consensus model* is a prefix model, with an additional allowance - that gaps of a certain size are allowed (a gap is from the right of one wagon to the left of the next wagon).
Group 2: consensus - consensus model

- Search for trains in satellite DNA (type of DNA that is located near region of centromere and is involved in chromosome segregation during mitosis and meiosis)
- Algorithm outputs any number of trains, where all of the following hold true:
  - there are at least \text{min\_repeat} wagons in a train, and each wagon’s length is between \text{min\_range} and \text{max\_range} (where \text{min\_repeat}, \text{min\_range}, and \text{max\_range} are inputs)
  - there are at most \( k = 15-20\% \) edit distance errors from each wagon to the consensus string (where \( k \) is an input)
  - jumps and gaps are of allowed sizes
Group 2: consensus - consensus model

Stage one: filtering = which substrings likely contain repeats

- Sum values in upper right region of local alignment dynamic programming matrix (Smith-Waterman, 1981)
- ETRs tend to have high sum - e.g., gacgac - sum of 20
- non-ETRs tend not to - e.g., gtcagt - sum of 9

<table>
<thead>
<tr>
<th></th>
<th>g</th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>g</th>
<th>t</th>
<th>c</th>
<th>a</th>
<th>g</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
Stage two: brute force search of substrings that were not filtered

- Set $L_m$ stores left-end positions of wagons known to be valid for consensus string $m$

- Attempt to find another valid consensus string by taking this previously good consensus string $m$ and inserting to left, substituting on left, or deleting on left
**Group 2: consensus - k-MAR**

**Definition**

(Amit, 2013) A *k-maximal approximate run* (or *k-MAR*) is a non-empty string \( r \), such that the modification of at most \( k \) characters in \( r \) generates a maximal (exact) run. A modification is a substitution of a single character with a different single character.

- \( aba\ abc\ aba \) - one-MAR of period three and three occurrences
- \( ab\ ab\ ab\ ac\ ac\ ab \) - two-MAR of period two and six occurrences
- \( k \)-MAR is consensus, since string will be modified in order to match consensus string
## Approximate Tandem Repeat

<table>
<thead>
<tr>
<th></th>
<th>neighboring</th>
<th>consensus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k$-repetition</td>
<td>$k$-run</td>
</tr>
<tr>
<td>Hamming distance</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Edit distance</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>All occurrences similar required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All occurrences similar not required</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Total count for number of variations</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Individual count for number of variations</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Total number of variations bounded</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Total number of variations not bounded</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Table 4: Alternative Grouping of ATRs
1. Hamming distance mismatches vs. edit distance errors
Approximate Tandem Repeat

<table>
<thead>
<tr>
<th></th>
<th>neighboring</th>
<th>consensus</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k$-repetition</td>
<td>$k$-run</td>
<td>$k$-edit repeat</td>
</tr>
<tr>
<td>All occurrences similar required</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>All occurrences similar not required</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

2. Overall, all occurrences similar to all others or whether string is evolutive = only adjacent occurrences must be similar (it is possible that evolve so much that far occurrences bear little resemblance)

- Consensus model - occurrences must be somewhat similar, as all occurrences similar to consensus string
- $k$-MARs are unusual - even though grouped as consensus in original grouping, they do not require similar occurrences (if all $k$ mismatches are in few occurrences, these few occurrences may be non-similar to other occurrences)
Approximate Tandem Repeat

3. Counting variations is total count (count over all occurrences) or individual count (counter is reset for each occurrence)

4. $\Sigma v =$ total number of variations - bounded ($\Sigma v$ does not exceed $k =$ fixed number known in advance) or not ($\Sigma v$ depends on number of occurrences, so $\Sigma v$ has no bound known in advance)

- Above two seem similar, but are different
- $k$-repetition, $k$-edit repeat, and $k$-MAR - total count, $\Sigma v$ bounded by $k$
- $k$-run - individual count, $\Sigma v$ unbounded ($k$ mismatches allowed for each occurrence, so $\Sigma v$ unbounded)
- Consensus model unusual - individual counter (counter is reset), yet $\Sigma v$ bounded (algorithm works when $\leq$ 15-20% edit distance errors between input string and consensus string)
Approximate Tandem Repeat

<table>
<thead>
<tr>
<th></th>
<th>neighboring</th>
<th>consensus model</th>
<th>k-MAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k$-repetition</td>
<td>$k$-run</td>
<td>$k$-edit repeat</td>
</tr>
<tr>
<td>Total count for number of variations</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Individual count for number of variations</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Total number of variations bounded</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Total number of variations not bounded</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Introductory Concepts
  KMP
  Suffix Tree

Tandem Repeats
  Exact Tandem Repeat
  Approximate Tandem Repeat
    Group 1: neighboring - k-repetition
    Group 1: neighboring - k-run
    Group 1: neighboring - k-edit repeat
    Group 2: consensus - consensus model
    Group 2: consensus - k-MAR

Related Research

Palindromes
  Linear-time Algorithm #1 for (Exact) Palindromes
  Linear-time Algorithm #2 for (Exact) Palindromes
  Variations of Palindromes
  Approximate Palindromes in RLE-compressed Data

Research Directions
Nested Tandem Repeat (NTR) = two distinct TRs, where one is nested within the other - e.g., NTR of the form $XxxxXxxxxXxxXxxXxXxxx$, such as:

- CTTGAGATT ACAT ACAT ACAT ACAT
- CTTGAGATT ACAT ACAT ACAT ACAT ACAT
- CTTGAGATT ACAT ACAT
- CTTGAGATT ACAT
- CTTGAGATT ACAT ACAT ACAT
Related Research

(Amir, 2010 and Eisenberg, 2012) Period Recovery Problem:

- String $r$ is periodic, but unknown.
- INPUT: string $r' = r$ corrupted with up to one edit distance error per occurrence, on average.
- OUTPUT: What’s $r$’s period? Identify the candidate set.

Example: $r' = abcabdabeabcabcabe$:

- $abcabdabe (abcabdabe abcabcabe)$
- $abcabcabe (abcabdabe abcabcabe)$
- $abc (abc abd abe abc abc abe)$

Example: $r' = abbabdacc$:

a. $abc (abb abd acc)$

b. $abd (abb abd acc)$
c. $abb (abb abd acc)$
d. not acc $(abb abd acc)$
Introductory Concepts
  KMP
  Suffix Tree
Tandem Repeats
  Exact Tandem Repeat
  Approximate Tandem Repeat
    Group 1: neighboring - k-repetition
    Group 1: neighboring - k-run
    Group 1: neighboring - k-edit repeat
    Group 2: consensus - consensus model
    Group 2: consensus - k-MAR
Related Research
Palindromes
  Linear-time Algorithm #1 for (Exact) Palindromes
  Linear-time Algorithm #2 for (Exact) Palindromes
  Variations of Palindromes
  Approximate Palindromes in RLE-compressed Data
Research Directions
Palindromes

String $r = uau^R$ is palindrome

- **radius** of palindrome = length of one arm
- **diameter** of palindrome = sum of lengths of arms
Palindromes

▶ The palindrome refers to *maximal* one
  ▶ $r = abababa$
  ▶ the palindrome is *abababa*
  ▶ not maximal: *aba*
The palindrome refers to maximal one

- \( r = abababa \)
- the palindrome is abababa
- not maximal: aba

Search string for substrings that are palindromes

One common approach is to find longest palindrome centered at each position
Introductory Concepts
KMP
Suffix Tree
Tandem Repeats
Exact Tandem Repeat
Approximate Tandem Repeat
  Group 1: neighboring - k-repetition
  Group 1: neighboring - k-run
  Group 1: neighboring - k-edit repeat
  Group 2: consensus - consensus model
  Group 2: consensus - k-MAR

Related Research

Palindromes
Linear-time Algorithm #1 for (Exact) Palindromes
Linear-time Algorithm #2 for (Exact) Palindromes
Variations of Palindromes
Approximate Palindromes in RLE-compressed Data

Research Directions
Linear-time Algorithm #1 for (Exact) Palindromes

- Manacher (1975): first linear-time algorithm to find (exact) palindromes

<table>
<thead>
<tr>
<th>position</th>
<th>0</th>
<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>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>algorithm</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Table:** Longest palindrome centered at each position
Linear-time Algorithm #1 for (Exact) Palindromes

- Manacher (1975): first linear-time algorithm to find (exact) palindromes

<table>
<thead>
<tr>
<th>position</th>
<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>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>algorithm</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table: Longest palindrome centered at each position

- Intuition is that if \( r = uau^R \), then lengths of palindromes centered at positions of left arm, \( u \), will mostly be mirror image of lengths of palindromes centered at positions of right arm, \( u^R \). Therefore, calculations of values in right arms are based on previously-calculated values of left arms.
Introductory Concepts
- KMP
- Suffix Tree

Tandem Repeats
- Exact Tandem Repeat
- Approximate Tandem Repeat
  - Group 1: neighboring - k-repetition
  - Group 1: neighboring - k-run
  - Group 1: neighboring - k-edit repeat
  - Group 2: consensus - consensus model
  - Group 2: consensus - k-MAR

Related Research

Palindromes
- Linear-time Algorithm #1 for (Exact) Palindromes
- Linear-time Algorithm #2 for (Exact) Palindromes
- Variations of Palindromes
- Approximate Palindromes in RLE-compressed Data

Research Directions
Create, in linear time, GST of $r$ and of $r^R$. Preprocess in order to be able to answer LCA query (which, for strings, is LCP query) in constant time.

For each position, perform type of LCP query called *longest common extension* (or LCE) query (see table below).

- LCE query for position $x$ = how many positions on right of $x$ match positions on left of $x$.

If any LCP query results in non-zero value, then there is palindrome, with radius size of that value, centered there.
Linear-time Algorithm #2 for (Exact) Palindromes

LCP queries:

- `abccba` - palindrome with radius three (diameter six)
- `abbaba` - palindrome with radius two (diameter four)
Linear-time Algorithm #2 for (Exact) Palindromes

LCP queries:

- *cabbad* - palindrome with radius two (diameter four)
- *abdcba* - no palindrome, since every LCP query returns zero

<table>
<thead>
<tr>
<th></th>
<th>cabbad</th>
<th>abbad</th>
<th>bbad</th>
<th>bad</th>
<th>ad</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCP</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>dabbac</th>
<th>c</th>
<th>ac</th>
<th>bac</th>
<th>bbac</th>
<th>abbac</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>abcdba</th>
<th>bcdba</th>
<th>cdba</th>
<th>dba</th>
<th>ba</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>abdcba</th>
<th>a</th>
<th>ba</th>
<th>cba</th>
<th>dcba</th>
<th>bdcba</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Introductory Concepts
  KMP
  Suffix Tree

Tandem Repeats
  Exact Tandem Repeat
  Approximate Tandem Repeat
    Group 1: neighboring - k-repetition
    Group 1: neighboring - k-run
    Group 1: neighboring - k-edit repeat
    Group 2: consensus - consensus model
    Group 2: consensus - k-MAR

Related Research

Palindromes
  Linear-time Algorithm #1 for (Exact) Palindromes
  Linear-time Algorithm #2 for (Exact) Palindromes
  Variations of Palindromes
  Approximate Palindromes in RLE-compressed Data

Research Directions
Palstar

Palstars = concatenation of palindromes

- KMP (1977): concatenation of even-length palindromes: a string is a palstar iff $r = a_1, \ldots, a_n$, where each $a_i$, $1 \leq i \leq n$, is a palindrome of even length.
  - Example: string of length zero, $abba$, $abbacc$, not $abaabba$
Palstar

Palstars = concatenation of palindromes

- KMP (1977): concatenation of even-length palindromes: a string is a palstar iff $r = a_1, \ldots, a_n$, where each $a_i$, $1 \leq i \leq n$, is a palindrome of even length.
  - Example: string of length zero, \textit{abba}, \textit{abbacc}, \textit{not ababba}

- Galil (1978): a string is a palstar iff $r = a_1, \ldots, a_n$, where each $a_i$, $1 \leq i \leq n$, is a palindrome, of any length (even or odd), and $|r| \geq 2$.
  - Example: \textbf{not} string of length zero, \textbf{not abba}, \textit{abbacc}, \textit{abaabba}
(KK, 2008) Gapped palindrome = $uau^R$, $|a| \geq 2$

Types:

1. **long-armed**: $|u| > |a|$ - e.g., $abbcddbbba$
2. **length-constrained**: $|u|$ and $|a|$ constrained by inputted sizes
Introductory Concepts
- KMP
- Suffix Tree

Tandem Repeats
- Exact Tandem Repeat
- Approximate Tandem Repeat
  - Group 1: neighboring - k-repetition
  - Group 1: neighboring - k-run
  - Group 1: neighboring - k-edit repeat
  - Group 2: consensus - consensus model
  - Group 2: consensus - k-MAR

Related Research

Palindromes
- Linear-time Algorithm #1 for (Exact) Palindromes
- Linear-time Algorithm #2 for (Exact) Palindromes
- Variations of Palindromes
  - Approximate Palindromes in RLE-compressed Data

Research Directions
Approximate Palindromes in RLE-compressed Data

Run-length encoding (RLE): encode string as runs of consecutive and identical symbols

- $bb \ cccc \ e \ dd \ aaaaa =$
  - $b^2 \ c^2 \ c^2 \ e^1 \ d^2 \ a^2 \ a^2 \ a^1$ (not optimal RLE)
  - $b^2 \ c^4 \ e^1 \ d^2 \ a^5$ (optimal RLE)
Approximate Palindromes in RLE-compressed Data

Run-length encoding (RLE): encode string as runs of consecutive and identical symbols

- \( bb \ cccc \ e \ dd \ aaaaa = \)
  - \( b^2 c^2 e^1 d^2 a^2 a^1 \) (not optimal RLE)
  - \( b^2 c^4 e^1 d^2 a^5 \) (optimal RLE)

Algorithm from Chen (2012):

- INPUT: string \( r \) of length \( n \) in optimal RLE-form, specified center position \( c \), allowed number of Hamming distance mismatches \( k \)
- OUTPUT: radius of approximate (\( \leq k \) Hamming distance mismatches) palindrome in \( r \) centered at \( c \)
- Without decompressing
Approximate Palindromes in RLE-compressed Data

Preprocessing: construct array $A[1 \cdots m]$, where $m =$ number of runs in compressed string. Array elements are start positions of run, when decompressed.

| position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|----------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| character| a | a | a | b | a | a | c | c | a | a | a | b | c | c | a | a | b | b |
| run      | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ | $X_9$ |
| RLE ($r$)| $a^3$ | $b^1$ | $a^2$ | $c^2$ | $a^3$ | $b^1$ | $c^2$ | $a^2$ | $b^2$ |

Table: $A$ for $r = aaabaaccaabcaabb$. There are $m = 9$ runs.
Approximate Palindromes in RLE-compressed Data

Preprocessing: construct array $A[1 \cdots m]$, where $m = \text{number of runs in compressed string}$. Array elements are start positions of run, when decompressed.

<table>
<thead>
<tr>
<th>position</th>
<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>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>run</td>
<td>$X_1$</td>
<td>$X_2$</td>
<td>$X_3$</td>
<td>$X_4$</td>
<td>$X_5$</td>
<td>$X_6$</td>
<td>$X_7$</td>
<td>$X_8$</td>
<td>$X_9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RLE ($r$)</td>
<td>$a^3$</td>
<td>$b^1$</td>
<td>$a^2$</td>
<td>$c^2$</td>
<td>$a^3$</td>
<td>$b^1$</td>
<td>$c^2$</td>
<td>$a^2$</td>
<td>$b^2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: $A$ for $r = aaabaaccaabccaabb$. There are $m = 9$ runs.

$LCP (\text{RLE (} r \text{)}, 5, 6) = 2$, since $X_4 = X_7$ and $X_3 = X_8$ (same character and same number of that character), but $X_2 \neq X_9$
Approximate Palindromes in RLE-compressed Data

- Stage one: binary search on RLE \( r \), in order to find in which run \( c \) is.
Approximate Palindromes in RLE-compressed Data

- Stage one: binary search on RLE \((r)\), in order to find in which run \(c\) is.
- Stage two: iterate, in order to extend palindrome outward, as much as possible.
  - Iteration ends when hits run boundary on left or on right.
  - If hit left and right simultaneously, then additional LCP query is performed, so that algorithm can “jump” over other runs, in order to extend even more.
  - End iterating when use \(k\) mismatches or when hit string boundary (position 1 or \(n\)).
Approximate Palindromes in RLE-compressed Data

INPUT:

- RLE ($r$) for $aaabaaccaaaabcccaabb = a^3ba^2c^2a^3bc^2a^2b^2$
- $c = 10.5$
- $k = 1$

<table>
<thead>
<tr>
<th>position</th>
<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>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>run</td>
<td>$X_1$</td>
<td>$X_2$</td>
<td>$X_3$</td>
<td>$X_4$</td>
<td>$X_5$</td>
<td>$X_6$</td>
<td>$X_7$</td>
<td>$X_8$</td>
<td>$X_9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Approximate Palindromes in RLE-compressed Data

**Input:**
- RLE \((r)\) for \(aaabaaccaaaabccaabb = a^3ba^2c^2a^3bc^2a^2b^2\)
- \(c = 10.5\)
- \(k = 1\)

<table>
<thead>
<tr>
<th>position</th>
<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>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>run</td>
<td>(X_1)</td>
<td>(X_2)</td>
<td>(X_3)</td>
<td>(X_4)</td>
<td>(X_5)</td>
<td>(X_6)</td>
<td>(X_7)</td>
<td>(X_8)</td>
<td>(X_9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stage one - binary search: \(c\) is in \(X_5\)
Approximate Palindromes in RLE-compressed Data

Stage two - iterations:

1. Extend to left and right of position 10.5, to positions 10 and 11, until right hits boundary.

<table>
<thead>
<tr>
<th>position</th>
<th>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>a a a b a a c c a a a b c c a a b b</td>
</tr>
<tr>
<td>run</td>
<td>$X_1$ $X_2$ $X_3$ $X_4$ $X_5$ $X_6$ $X_7$ $X_8$ $X_9$</td>
</tr>
</tbody>
</table>
Approximate Palindromes in RLE-compressed Data

2. Extend to positions 9 and 12, which do not match, so 1 mismatch is found. Left boundary hits $X_5$ at the same time that right boundary hits $X_6$, so there is a query of LCP (RLE ($r$), 5, 6), which returns two (as explained above), meaning that two runs will be jumped over, and next iteration looks at $X_2$ and $X_9$.

<table>
<thead>
<tr>
<th>position</th>
<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>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>run</td>
<td>$X_1$</td>
<td>$X_2$</td>
<td>$X_3$</td>
<td>$X_4$</td>
<td>$X_5$</td>
<td>$X_6$</td>
<td>$X_7$</td>
<td>$X_8$</td>
<td>$X_9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

radius = 6
mismatches = 1
Approximate Palindromes in RLE-compressed Data

Iterations:

3. $X_2$ and $X_9$ contain $b$. Left hits boundary of $X_2$ before right hits boundary.

<table>
<thead>
<tr>
<th>position</th>
<th>1 2 3</th>
<th>4</th>
<th>5 6</th>
<th>7 8</th>
<th>9 10 11</th>
<th>12</th>
<th>13 14</th>
<th>15 16</th>
<th>17 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>a a a</td>
<td>b</td>
<td>a a</td>
<td>c c</td>
<td>a a a</td>
<td>b</td>
<td>c c</td>
<td>a a</td>
<td>b b</td>
</tr>
<tr>
<td>run</td>
<td>$X_1$</td>
<td>$X_2$</td>
<td>$X_3$</td>
<td>$X_4$</td>
<td>$X_5$</td>
<td>$X_6$</td>
<td>$X_7$</td>
<td>$X_8$</td>
<td>$X_9$</td>
</tr>
</tbody>
</table>

radius = 7
mismatches = 1
Approximate Palindromes in RLE-compressed Data

Iterations:

4. $X_1$ and $X_9$ contain different characters. No more mismatches allowed, so algorithm stops.

| position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|----------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| character| a | a | a | b | a | a | c | c | a | a | a | b | c | c | a | a | b | b |
| run      | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ | $X_9$ |     |     |     |     |     |     |     |     |

radius = 7
mismatches = 1
Introductory Concepts
  KMP
  Suffix Tree

Tandem Repeats
  Exact Tandem Repeat
  Approximate Tandem Repeat
    Group 1: neighboring - k-repetition
    Group 1: neighboring - k-run
    Group 1: neighboring - k-edit repeat
    Group 2: consensus - consensus model
    Group 2: consensus - k-MAR

Related Research

Palindromes
  Linear-time Algorithm #1 for (Exact) Palindromes
  Linear-time Algorithm #2 for (Exact) Palindromes
  Variations of Palindromes
  Approximate Palindromes in RLE-compressed Data

Research Directions
Research Directions

Warburton (2004):

- Palindrome = *inverted repeat* (IR)
- IR = substring followed by gap and then by substring’s reverse
  = gapped palindrome
  - (and when gap is length zero, then IR = palindrome)
- Repeats and palindromes really are similar problems
- We want to apply research from repeats towards palindromes
Research Directions

2D palindrome = square block of text rotated around center results in same square block of text
Research Directions

- Discussed how to find palindromes in RLE-compressed data, without decompressing.
- Want to find palindromes (or 2D palindromes) in data that was compressed using other compression schemes, without decompressing.