An Improved Algorithm for Approximate String Matching

Yuanlei Zhang

Reference:
An Improved Algorithm for Approximate String Matching
ZVI GALIL and KUNSOO PARK

To Be Covered …

- Introduction
- Formalization
- $O(mn)$ algorithm 1
- $O(mn)$ algorithm 2
- The new algorithm
- Conclusion
Introduction

The problem
- Given a text string, a pattern string, and an integer k, find all occurrences of the pattern string in the text string with at most k differences.
- String matching with k differences

Difference
- A difference is one of the following:
  1) A character of the pattern corresponds to a different character of the text.
  2) A character of the text corresponds to no character in the pattern.
  3) A character of the pattern corresponds to no character in the text.

An example

Formalization

The ith character of a string x is denoted by x(i).

A substring of x from the ith through the jth characters is denoted by x(i)…x(j).

If the minimum number of differences between the pattern y and any substring of the text x ending at x(j) is less than k, we say that y occurs at position j of x with at most k differences.

The problem of string matching with k differences is defined as follows:
- Given a text x of length n, a pattern y of length m, and an integer k (k<=m<=n), find all positions of x where y occurs with at most k differences.
O(mn) Algorithm 1

- Let \( D(i,j) \), \( 0 \leq i \leq m \) and \( 0 \leq j \leq n \), be the minimum number of differences between \( y(1) \ldots y(i) \) and any substring of \( x \) ending at \( x(j) \). For \( 1 \leq i \leq m \) and \( 1 \leq j \leq n \), the differences between \( y(1) \ldots y(i) \) and \( x(h) \ldots x(j) \) for some \( h \), \( 1 \leq h \leq j \) are either:
  1. Difference between \( y(1) \ldots y(i-1) \) and \( x(h) \ldots x(j-1) \) + the difference between \( y(i) \) and \( x(j) \)
  2. Difference between \( y(1) \ldots y(i) \) and \( x(h) \ldots x(j-1) \) + a difference of type 2) at \( x(j) \)
  3. Difference between \( y(1) \ldots y(i-1) \) and \( x(h) \ldots x(j) \) + a difference of type 3) at \( y(i) \)
- \( D(i,j) = \min\{D(i-1,j-1)+s(i,j), D(i,j-1)+1, D(i-1,j)+1\} \), where \( s(i,j)=0 \) if \( x(j)=y(i) \); it is 1 otherwise.

- \( D(0,0) = 0 \) for \( 0 \leq i \leq m \); \( D(0,j) = 0 \) for \( 0 \leq j \leq n \).
- \( D(m,j) \leq k \) if and only if the pattern occurs at position \( j \) of the text with at most \( k \) differences.

Algorithm MN1
for \( i=0 \) to \( m \) do \( D(i,0) := i \) end for
for \( j=0 \) to \( n \) do \( D(0,j) := 0 \) end for
for \( j=1 \) to \( n \) do
  for \( i=1 \) to \( m \) do
    row := \( D(i-1,j) + 1 \)
    col := \( D(i,j-1) + 1 \)
    if \( x(j)=y(i) \) then diag := \( D(i-1,j-1) \)
    else diag := \( D(i-1,j-1) + 1 \) end if
    \( D(i,j) := \min\{\text{row, col, diag}\} \)
  end for
end for
O(mn) Algorithm 1

- The algorithm computes the D table column by column. Since there are $O(mn)$ entries and each entry takes constant time to be computed, algorithm MN1 takes time $O(mn)$.

- Example
  - Let $x=abbdadcbbc$, $y=adbbc$, and $k=2$. The computation of table $D(i,j)$ is shown below. The pattern occurs at positions 3, 4, 7, 8 and 9 of the text with at most two differences.

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

\[
C(2,1)=4
\]

O(mn) Algorithm 2

- Let $D$-diagonal $d$ be the entries of table $D(i,j)$ such that $j-i=d$. For a $D$-diagonal $d$ and a difference $e$, let $C(e,d)$ be the largest column $j$ such that $D(j-d, j)=e$.

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

- We can store the information of the $D$ table in a more compact way as the $C$ table.
**O(mn) Algorithm 2**

- Example of the C table

<table>
<thead>
<tr>
<th>C</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>-2</td>
</tr>
<tr>
<td>-1</td>
<td>-N</td>
</tr>
<tr>
<td>0</td>
<td>-N</td>
</tr>
<tr>
<td>1</td>
<td>-N</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

- C(e,d) = m+d for some e<=k iff the pattern occurs at position m+d of the text with at most k differences.
- For D-diagonal d=-2, -1, 2, 3, and 4, C(2,d) = 5+d. Thus the pattern occurs at positions 3, 4, 7, 8, and 9 of the text with at most two differences.

---

**O(mn) Algorithm 2**

- Computation of the C table
  - For D-diagonal d>=0, C(-1,d) = d-1 (green)
  - For D-diagonal -(k+1)<=d<=-1, C(|d|-2, d) = -N (red)
  - Let C-diagonal c be the entries of table C(e,d) such that e+d=c, we can compute the rest of the C table C-diagonal by C-diagonal.
O(mn) Algorithm 2

- **Algorithm MN2**
  
  for $d:=0$ to $n-m+k+1$ do $C(-1,d):=d-1$ end for
  
  for $d:=-(k+1)$ to $-1$ do
    $C([-d],-1):=-1$
    $C([-d|-2],-d):=-N$
  end for
  
  for $c:=0$ to $n-m+k$ do
    for $e:=0$ to $k$ do
      $d:=c-e$
      $col:=\max(C(e-1,d-1)+1, C(e-1,d)+1, C(e-1,d+1))$
      while $col<n$ and $col<m+d$ and $x(col+1)=y(col+1-d)$ do
        $col:=col+1$
      end while
      $C(e,d):=\min(col, m+d, n)$
    end for
  end for

- The computation of each C-diagonal takes time $O(m)$. Since there are $n-m+k+1$ C-diagonals, algorithm MN2 takes time $O(mn)$.

The New Algorithm

- The new algorithm consists of preprocessing of the pattern followed by processing of the text.
  
  The preprocessing builds an upper triangular table $Prefix(i,j)$, $1\leq i\leq j\leq m$, where $Prefix(i,j)$ is the length of the longest common prefix of $y(i)\ldots y(m)$ and $y(j)\ldots y(m)$.
  
  The text processing consists of $n-m+k+1$ iterations, one for each C-diagonal, as algorithm MN2 does. Whereas algorithm MN2 relies only on direct comparisons of the text with the pattern, the new algorithm uses both direct comparisons and lookups of the Prefix table.
  
  Reference triples help decide whether to compare a substring of the pattern with a substring of the text or to compare two substrings of the pattern.
The New Algorithm

The new algorithm

Initializations
for c:=0 to n-m+k do
  for e:=0 to k do
    d:=c-e
    col:= max(C(e-1,d-1)+1, C(e-1,d)+1, C(e-1,d+1))
    found:=false
    while not found do
      if within some reference triple then
        look up Prefix table to compare two substrings of the pattern
      else
        // compare a substring of the pattern with a substring of the text
        if col<n and col<m+d and x(col+1)=y(col+1-d) then
          col:=col+1
          found:=true
        else
          update reference triple
        end if
      end if
    end while
    C(e,d):=min(col, m+d, n)
  end for
end for

The computation of each C-diagonal only takes time O(k) now. Since there are n-m+k+1 C-diagonals, the text processing of the new algorithm takes time O(kn).

The preprocessing takes time O(m^2) to build the Prefix table.

Taking into account both preprocessing and text processing, the new algorithm takes time O(kn+m^2).
Conclusion

- The new algorithm for the string matching with k differences improves upon the known algorithms given $k \leq m$. 