Dictionary Matching With Uneven Gap

Yilin Yang
Joint work with Wing-Kai Hon, Tak-Wah Lam, Rahul Shah, Sharma Thankachan, Hing-Fung Ting
Outline

- Problem Definition:
  Dictionary Matching With Uneven Gap

- Previous Work

- Main Idea

- Symmetric Problem:
  Dictionary Matching With Missing Substring
Outline

- Problem Definition:
  Dictionary Matching With Uneven Gap

- Previous Work

- Main Idea

- Symmetric Problem:
  Dictionary Matching With Missing Substring
Problem

- Preprocess: a dictionary with $d$ gapped patterns
- Gapped pattern:

  $p_i$ gap $[a_i, b_i]$ $Q_i$

- Input: text $T$
- Output: all the occurrences
Example

- Pattern: \textcolor{blue}{ab} [1,2] \textcolor{green}{cd}

- Text: abcdcdabcdabcccccd

- One match:
  \textcolor{blue}{abcdcdabcccccd}
  ab??cd
Outline

· Problem Definition:
  Dictionary Matching With Uneven Gap

· Previous Work

· Main Idea

· Symmetric Problem:
  Dictionary Matching With Missing Substring
Previous Work

- Dictionary matching with one gap (Amir et al. CPM 2014)

- The **same** lower bound and upper bound of gap through all patterns

```
\begin{align*}
\text{p}_1 & \quad \text{gap } [a, b] & \quad Q_1 \\
\text{p}_2 & \quad \text{gap } [a, b] & \quad Q_2 \\
\text{p}_i & \quad \text{gap } [a, b] & \quad Q_i
\end{align*}
```
Our Problem

- Dictionary matching with uneven gap

\[ p_1 \quad \text{gap} \ [a_1 , b_1] \quad Q_1 \]
\[ p_2 \quad \text{gap} \ [a_2 , b_2] \quad Q_2 \]
\[ p_i \quad \text{gap} \ [a_i , b_i] \quad Q_i \]
## Result

<table>
<thead>
<tr>
<th></th>
<th>Amir et al.</th>
<th>Amir et al.</th>
<th>Ours (uneven gap)</th>
<th>Ours (uneven gap)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>$O(</td>
<td>T</td>
<td>\cdot r \log^2 n \log \log d + \text{occ})$</td>
<td>$O(</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>$O(n + d \log^k d)$</td>
<td>$O(n + d^2)$</td>
<td>$O(n)$</td>
<td>$O(n + d^{1+\epsilon})$</td>
</tr>
</tbody>
</table>

Note: $r = (B - A + 1)$, $A = \min(a_i)$, $B = \max(b_i)$, $\lambda \leq d$
Outline

- Problem Definition:
  Dictionary Matching With Uneven Gap

- Previous Work

- Main Idea

- Symmetric Problem:
  Dictionary Matching With Missing Substring
Search Framework

- If a gapped pattern \( i \) appears in the text \( T \) at position \( k \):
  - We can find \( \text{rev}(P_i) \) as a prefix of \( T[k : 1] \)
  - We can find \( Q_i \) as a prefix of \( T[k+g : |T|] \)
Search Framework

- So we construct two suffix trees:

  - $T_1$: for $\text{rev}(P_i)$
  - $T_2$: for $Q_i$
Search Framework

- When searching, we iterate position $k$ and the gap size $g$
Search Framework

- When searching, we iterate position k and the gap size g

- Simulate “Insertion” of T[k : 1] into T₁ and T[k+g : |T|] into T₂ to get two locus nodes: u and v
Search Framework

- We may find some \( \text{rev}(P_i) \) appear as a prefix of \( T[k : 1] \), and some \( Q_i \) appear as a prefix of \( T[k+g : |T|] \)
Search Framework

- Now, we need to output the intersection of them.
- Ex: we find pattern 3 and pattern 6 at position k with gap size g.
Problem translation

- With the framework above, the problem becomes to “Tree Path Intersection” problem:

  - Given two trees and two paths on each tree
  - Output the intersection
Previous solution

- Amir et al. uses heavy path decomposition and relabels the heavy path with contiguous number

- Each pattern forms a 2D point

- Then the query path would be cut into at most $\log n$ heavy paths

- Each query forms $\log^2 n$ rectangles

- Time complexity: $O(|T| (B-A+1) \log^2 n \log\log d + \text{occ})$
Our idea

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Amir et al.</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D point</td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text</th>
<th>Amir et al.</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\log^2 n$ rectangles</td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>
Our idea

- With different view ...

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Amir et al.</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2D point</td>
<td>3D rectangle</td>
</tr>
<tr>
<td>Text</td>
<td>$\log^2 n$ rectangles</td>
<td>“only one” point !</td>
</tr>
</tbody>
</table>
Index method

- The method is very simple: we relabel the tree using **preorder traversal rank**.
Index method

- Property: any subtree would contain contiguous relabeled number
Index method

- For each gapped pattern $i$, we can find

- locus node $x$ of $\text{rev}(P_i)$
- locus node $y$ of $Q_i$
Index method

- And construct a 3D rectangle:
  - x coordinate: $[\text{pre}(x), \text{pre}(x)+\text{size}(x)]$
  - y coordinate: $[\text{pre}(y), \text{pre}(y)+\text{size}(y)]$
  - z coordinate: gap size $[a_i, b_i]$
Query

- Then, each query can be changed into only one point: \((\text{pre}(u), \text{pre}(v), g)\)

- rectangle stabbing query
Index method

- Space: $O(n + d \log d)$
- Time: $O(|T| (B - A + 1) \log^2 d + occ)$
Our idea

- With different view ...

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Amir et al.</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>$\log^2 n$ rectangles</td>
<td>“only one” point!</td>
</tr>
</tbody>
</table>
# Result

<table>
<thead>
<tr>
<th></th>
<th>Amir et al.</th>
<th>Amir et al.</th>
<th>Ours (uneven gap)</th>
<th>Ours (uneven gap)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>$O(</td>
<td>T</td>
<td>\cdot r \log^2 n \log \log d + \text{occ})$</td>
<td>$O(</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>$O(n + d \log^6 d)$</td>
<td>$O(n + d^2)$</td>
<td>$O(n)$</td>
<td>$O(n + d^{1+\varepsilon})$</td>
</tr>
</tbody>
</table>

Note: $r = (B-A+1)$, $A = \min(a_i)$, $B = \max(b_i)$, $\lambda \leq d$
Linear space

- Idea: we categorize the patterns into long and short patterns
- Then we can achieve the same time complexity, but linear space
More trade-offs

- Using more space: $O(n + d^{1+\varepsilon})$
- Faster query time: $O(|T| (B-A+1) + \text{occ})$
More trade-offs

- Succinct space: $n \log \sigma + O(d \log n) + o(n \log \sigma)$ bits

- Query time: $O(|T| (B-A+1) \log \lambda \log^{2+\epsilon} n + \text{occ})$
Outline

- Problem Definition:
  Dictionary Matching With Uneven Gap

- Previous Work

- Main Idea

- Symmetric Problem:
  Dictionary Matching With Missing Substring
Symmetric Problem

- Dictionary Matching With Missing Substring

- Pattern:

  - Find exact match in T
Symmetric Problem

- Pattern: abcd, b = 2
- Text T: acdabdad
- Three occurrences:

  acdabdad
  acd

  acdabdad
  abd

  acdabdad
  ad
Symmetric Problem

- It’s like that we insert a gap into text $T$

- Dictionary matching with uneven gap:
Symmetric Problem

- For each pattern, if we know the starting position (only one) of missing substring:
  - Space: $O(n)$
  - Time: $O(|T| \log n + \text{occ})$
Symmetric Problem

- For each pattern, if we don’t know the starting position of missing substring: (means it would happen anywhere in the pattern)

- Space : $O(n \log n)$

- Time : $O(k|T| \log n + \text{occ})$, $k = \max(|\text{Pattern}|)$
Open Problems

- In dictionary matching with uneven gap problem, can we solve it with more than one gap?

- In dictionary matching with missing substring problem, can we obtain succinct solution?