Text Indexing: Lecture 5

Simon Gog – gog@kit.edu
Document Listing

Problem
Given a query \( q \), list all the distinct documents which contain \( q \).

Solution by Muthukrishnan ([3])
- Precompute text index + document array \( D \)
- Precompute array \( E \) with \( E[i] = \max j \mid j < i \land D[j] = D[i] \)
- Use text index to get lex. range of \( q \)
- Use range minimum queries on \( E \) to get distinct documents in the lex. range of \( q \)
- Running time in number of distinct documents
Example

\[ i = \begin{array}{cccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
\end{array} \]

\[ C = \begin{array}{ccccccccccccc}
\end{array} \]

\[ D = \begin{array}{cccccccccccccc}
0 & 3 & 1 & 2 & 3 & 1 & 2 & 2 & 2 & 3 & 1 & 3 & 1 & 2 & 3 \\
\end{array} \]

\[ E = \begin{array}{cccccccccccccc}
-1 & -1 & -1 & -1 & 1 & 2 & 3 & 6 & 7 & 4 & 5 & 9 & 10 & 8 & 11 \\
\end{array} \]

\[ \text{SA} = \begin{array}{cccccccccccccc}
14 & 13 & 3 & 8 & 12 & 2 & 7 & 6 & 5 & 10 & 0 & 11 & 1 & 4 & 9 \\
\end{array} \]
Example \((q = TA)\)

<table>
<thead>
<tr>
<th>(i)</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>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C)</td>
<td>A</td>
<td>T</td>
<td>A</td>
<td>$#$</td>
<td>T</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>$#$</td>
<td>T</td>
<td>A</td>
<td>T</td>
<td>A</td>
<td>$#$</td>
<td>$$</td>
</tr>
</tbody>
</table>

\(d_1\) \hspace{2cm} \(d_2\) \hspace{2cm} \(d_3\)

<table>
<thead>
<tr>
<th>(i)</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>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>(SA)</td>
<td>14</td>
<td>13</td>
<td>3</td>
<td>8</td>
<td>12</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>(D)</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(E)</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>11</td>
</tr>
</tbody>
</table>
Example ($q = TA$)

$$
\begin{align*}
  i &= 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \\
  C &= A \ T \ A \ # \ T \ A \ A \ A \ # \ T \ A \ T \ A \ # \ $ \\
  D &= d_1 \ 3 \ 2 \ 2 \ 2 \ 3 \ 1 \ 1 \ 2 \ 2 \ 2 \ 3 \ 1 \ 1 \ 2 \ 3 \\
  E &= d_2 \ -1 \ -1 \ -1 \ -1 \ 1 \ 2 \ 3 \ 6 \ 7 \ 4 \ 5 \ 9 \ 10 \ 8 \ 11 \\
  SA &= d_3 \ 14 \ 13 \ 3 \ 8 \ 12 \ 2 \ 7 \ 6 \ 5 \ 10 \ 0 \ 11 \ 1 \ 4 \ 9 \\
  i &= \ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14
\end{align*}
$$
Example ($q = TA$)

\[
\begin{align*}
  i &= 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11 \quad 12 \quad 13 \quad 14 \\
  C &= \begin{array}{cccccccccccc}
\end{array} \\
  d_1 &= \begin{array}{cccccccccccc}
        \end{array} \\
  d_2 &= \begin{array}{cccccccccccc}
        \end{array} \\
  d_3 &= \begin{array}{cccccccccccc}
        \end{array}
\end{align*}
\]

\[
\begin{align*}
  i &= 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11 \quad 12 \quad 13 \quad 14 \\
  SA &= \begin{array}{cccccccccccc}
        14 & 13 & 3 & 8 & 12 & 2 & 7 & 6 & 5 & 10 & 0 & 11 & 1 & 4 & 9 \\
\end{array} \\
  D &= \begin{array}{cccccccccccc}
        0 & 3 & 1 & 2 & 3 & 1 & 2 & 2 & 2 & 3 & 1 & 3 & 1 & 2 & 3 \\
\end{array} \\
  E &= \begin{array}{cccccccccccc}
        -1 & -1 & -1 & -1 & 1 & 2 & 3 & 6 & 7 & 4 & 5 & 9 & 10 & 8 & 11 \\
\end{array}
\end{align*}
\]
Example \((q = TA)\)

\[
\begin{array}{cccccccccccccccc}
 i & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
 C & = & \text{A T A #} & \text{T A A A #} & \text{T A A A #} & \$
\end{array}
\]

\[
\begin{array}{cccccccccccccccc}
 i & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
 SA & = & 14 & 13 & 3 & 8 & 12 & 2 & 7 & 6 & 5 & 10 & 0 & 11 & 1 & 4 & 9 \\
 D & = & 0 & 3 & 1 & 2 & 3 & 1 & 2 & 2 & 2 & 3 & 1 & 3 & 1 & 2 & 3 \\
 E & = & -1 & -1 & -1 & -1 & 1 & 2 & 3 & 6 & 7 & 4 & 5 & 9 & 10 & 8 & 11 \\
\end{array}
\]
Document Listing

Algorithm

document_listing(q)
  [i, j] ← backward_search (CSA, q)
  document_listing_rec([i, j], i)

document_listing_rec([i, j], sp)
  if (j ≥ i) then
    p ← RMQ(i, j)
    if E[p] < sp then
      output D[p]
      document_listing(i, p − 1, sp)
      document_listing(p + 1, j, sp)
Range Minimum Query

Given an array $A[0, n - 1]$ of elements from a totally ordered set. A range minimum query $rmq(i, j)$ calculates the index $k$ of the smallest element $A[k]$ in a range $[i, j]$.

Range Minimum Query Index

Precalulate $M[0, n - 1][\log n]$ with $M[i][j] = rmq(i, i + 2^j - 1)$. Then

$$rmq(i, j) = \begin{cases} M[i][r] & \text{if } A[M[i][r]] \leq A[M[j - 2^r + 1][r]] \\ M[j - 2^r + 1][r] & \text{otherwise} \end{cases}$$

where $r = \max\{r \mid 2^r \leq j - i + 1\}$. This solution takes $O(n \log^2 n)$ bits of space and achieves $O(1)$ query time. This is much better than the naive variant of precalculate all $O(n^2)$ answers. There is a solution which takes just $2n + o(n)$ bits and achieves $O(1)$ query time.
Problem
Given one query term (or phrase) $q$ of length $m$ and parameter $k$. Report the top-$k$ documents with respect to term frequency.

First solution
- Retrieve $x$ distinct documents $\{d_{r_1}, \ldots, d_{r_x}\}$ in which $q$ occurs
- Determine frequency $f_{d_{r_i},q}$ of $q$ in each $d_{r_i}$ (question: How can this be done in $\log n$ time per document?)
- Maintain min-heap of (frequency,document)-pairs of size $k$
- Total time complexity: $O(m + x \log k)$

Already a solution which is not dependent on $n$. Drawback: $x$ can be as large as $N$, the number of documents.
Optimal solution provides \(O(m + k)\) query time in \(O(n \log n)\) bits.

We will discuss a simplified but very practical version presented recently (see [1]). Query time \(O(m \cdot t_{LF} + \log n \cdot k)\).

The optimal version was discovered by Navarro & Nekrich [4] and is based on the framework of Hon et al. [2].
Top-\(k\) for Single-Term Frequency

Build generalized suffix tree (GST)

Direction of the edge labels different to last lecture ;)

Simon Gog:
Text Indexing: Lecture 5
**Top-$k$ for Single-Term Frequency**

Add document ids to inner nodes

A document id $i$ is added to node $v$, if $v$ is the lowest common ancestor of two leaf nodes marked by $i$. 

```
A document id $i$ is added to node $v$, if $v$ is the lowest common ancestor of two leaf nodes marked by $i$. 
```
Top-\( k \) for Single-Term Frequency

Number nodes
Top-$k$ for Single-Term Frequency

Node numbering

Assign each inner node $v$ an identifier $\text{id}(v)$ which is the index of the rightmost leaf in the subtree of $v$’s leftmost child plus one. We get three properties:

- $\text{id}(v) \neq \text{id}(w)$ for all inner nodes $v, w$ of $T$
- $\text{id}(v) \in [1, n]$
- $\text{id}(v) - 1 \in [\text{lb}(v), \text{rb}(v)]$, where $\text{lb}(v)$ and $\text{rb}(v)$ are the index of the index of the leftmost leaf and rightmost leaf in $v$’s subtree ($[\text{lb}(v), \text{rb}(v)]$ = $v$’s SA-interval)

In the following we denote a node $w$ by $v_{\text{id}(w)}$.

Proof of first property

Assume there are two nodes $v \neq w$ and $\text{id}(v) = \text{id}(w)$.

- Case 1: $\text{LCA}(v, w) \in \{v, w\}$ ($\text{LCA}(v, w) = w$ analogously).
- Case 2: $\text{LCA}(v, w) \notin \{v, w\}$
Top-$k$ for Single-Term Frequency

Connect nodes with closest ancestor nodes

Connect node id $i$ at node $v_x$ to the closest ancestor $v_y$ which also contains id $i$
Top-\(k\) for Single-Term Frequency

Observations:

- The subset of nodes which are marked with document id \(i\) correspond to the suffix tree of document \(i\).
- Document id \(i\) occurs at most \(n_i\) times in the leaves of the GST and \(n_i - 1\) times in inner nodes.
- Summing up all document labels results in at most \(2n - N\) entries.

Next: Answering a query on the index.
Top-$k$ for Single-Term Frequency

Query for $q = TA$

Select the *locus* of $q = TA$, which is the first node $v$ such that the path labels from the root to $v$ is prefixed by $q$. 
Observations:

- Per document $i$ there is at most one pointer leaving the subtree of the locus $v$.
- We associate a weight with each pointer for a document $i$. The weight is the frequency of document $i$ in the subtree rooted by the pointer’s source node.
- The pointer of document $i$ leaving the subtree of the locus $v$ has the maximum weight of all pointers for document $i$ in $v$’s subtree.
- Document listing corresponds to determine all pointers leaving the subtree of the locus.
- Top-$k$ term frequency corresponds to retrieving the $k$ pointers (respectively their associated documents) with the highest weights leaving the subtree of the locus.

Next: Orthogonal range queries can be used to solve the problem.
In the following we will only consider documents which occur more than two times. I.e. we remove all pointers of weight 1...
Top-$k$ for Single-Term Frequency

GST including all pointers of weight $\geq 2$

...we will handle weight-1 documents later.
Top-$k$ for Single-Term Frequency

GST including all pointers of weight $\geq 2$

...we will handle weight-1 documents later.
Top-$k$ for Single-Term Frequency

- We assign each pointer to a $(x, y)$-coordinate.
  - The $x$-component is dependent on the numbering of the pointers source node
  - The $y$-coordinate to the string depth of the pointers target node
- Let $\text{depth}(v)$ be the string depth of locus node $v$. Than all pointers which leave the subtree have a $y$-coordinate $< \text{depth}(v)$
Top-\(k\) for Single-Term Frequency

GST, x-mapping H, and grid G

\[
H = \begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1
\end{pmatrix}
\]

\[
DOC = \begin{pmatrix}
1 & 2 & 3 & 2 & 3 \\
0 & 1 & 2 & 3 & 4 \\
0 & 0 & 0 & 1 & 0 \\
2 & 3 & 2 & 2 & 2
\end{pmatrix}
\]
Top-$k$ for Single-Term Frequency

Example: $q = A$

Example graph with nodes and edges.
Top-$k$ for Single-Term Frequency

- Note: Mapping from pointer to $x$-coordinate similar to mapping in document frequency data structure (see Lecture 4)
- DOC stores the documents associated with a pointer
- Possible data structure for grid $G$:
  - A wavelet tree augmented over $y$-coordinates augmented by an RMQ structure over $w$ on each level
  - A $K^2$-treap
- If there are less than $k$ documents reported from the grid, we can use the document listing structure to report single-occurrence documents
- Space for document listing: $2n + o(n)$ bits + space to retrieve the document numbers. $2N + N \log \frac{n}{N} + o(N)$ bits for bitvector $B$ which indicates a separator symbol in $C$ and rank structure on $B$
Top-$k$ for Single-Term Frequency

<table>
<thead>
<tr>
<th>M0</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

N0: (0,3)-8
N1: (4,4)-7
N2: (6,6)-3
N3: 2 1 0

Simon Gog: Text Indexing: Lecture 5

Institute of Theoretical Informatics
Algorithmics
Top-\(k\) for Single-Term Frequency

Institute of Theoretical Informatics
Algorithmics

coord[0] \((0,3)\)
coord[1] \((2,1)\) \((0,2)\) \((0,0)\)
coord[2] \((0,0)\) \((1,0)\) \((0,0)\) \((0,0)\) \((0,1)\) \((0,0)\)
values \(8 -- 1 1 1 -- 2 5 0 3 2 1 4 -- 4 4 2 3 3 2 2 5 1 2 3\)
first \(0 1 4 11\)
Experiments from [1] – Collections

| Collection          | $n$          | $N$     | $n/N$ | $\sigma$ | $|C|$ in MB |
|---------------------|--------------|---------|-------|----------|-------------|
| **character alphabet** |             |         |       |          |             |
| ENWIKI-SML$_c$      | 68,210,334   | 4,390   | 15,538| 206      | 65          |
| ENWIKI-BIG$_c$      | 8,945,231,276| 3,903,703| 2,291 | 211      | 8,535       |
| **word alphabet**   |             |         |       |          |             |
| ENWIKI-SML$_w$      | 12,741,343   | 4,390   | 2,902 | 281,577  | 29          |
| ENWIKI-BIG$_w$      | 1,690,724,944| 3,903,703| 433  | 8,289,354| 4,646       |
| GOV2$_w$            | 23,468,782,575| 25,205,179| 931  | 39,177,922| 72,740      |
Experiments [1] – Detailed space breakdown
Experiments [1] – Detailed query time analysis

Query times for IDX_GN on ENWIKI-BIG$^C$, with $m = 5$. Left: Query time depending on $k$ with a detailed breakdown of the three query phases. Right: Average time per document (mixed), per $K^2$-treap retrieved document and RmQC+CSA retrieved document. CSA matching time is included in all cases.
Experiments [1] – Query time dependence on $K^2$-treap bitvector

![Graph showing query time per query in microseconds for different datasets and bitvectors.]

- ENWIKI-SML
- ENWIKI-BIG
- ENWIKI-SML
- ENWIKI-BIG
- GOV2

Legend:
- rrr_vector<63>
- bit_vector

Average time per query [μs]
Experiments [1] – Query time dependent on pattern length

Pattern length $m$

ENWIKI-SML$^c$

Avg. time per query [$\mu$s]

ENWIKI-BIG$^c$

ENWIKI-SML$^w$

ENWIKI-BIG$^w$

