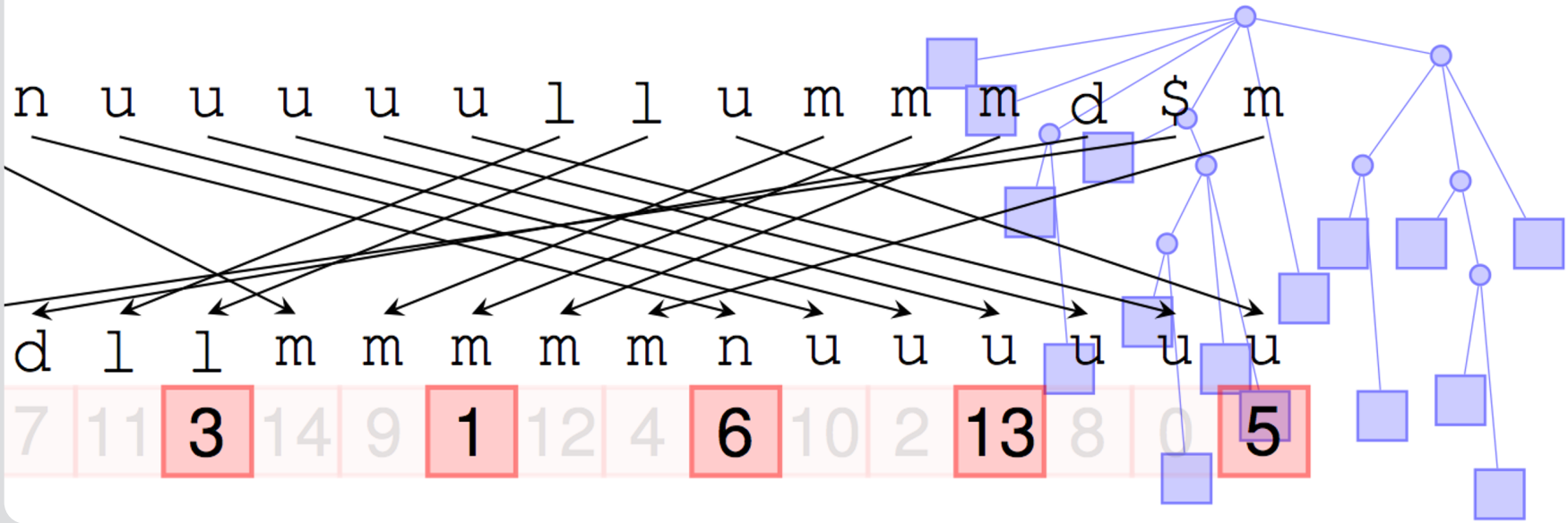
Text Indexing: Lecture 12

Simon Gog – *gog@kit.edu*

Institute of Theoretical Informatics - Algorithmics



Geometric Burrows Wheeler Transform

General idea

Use two indexes and a $2d$-range query structure to combine the result of the two indexes to the result of locate and count queries.

- First index: A *sparse* index over T to reduce space
- Second index: An index supporting prefix searches over the BWT of the reversed meta characters of T

- A sparse index groups d characters (each of $\log \sigma$ bits) to a meta character of $d \log \sigma$ bits.
- I.e. a text T of length n is transformed into a meta text T' of length $\lceil \frac{n}{d} \rceil$

Sparse indexes

Example (for $d = 3$)

```
                1           2
    0123456789012345678901234567890
T = mmiissiissiippiissiissiippimm$

     0   1   2   3   4   5   6   7   8   9
T' = mmi iss iis sii ppi iss iis sii ppi mm$
```

Sparse indexes

Example (for $d = 3$)

```
                                                    1
     0   1   2   3   4   5   6   7   8   9   0
T' = mmi iss iis sii ppi iss iis sii ppi mm$ $$$
```

i	BWT'	SA'	
0	mm$	10	$$$
1	iss	2	iis sii ppi iss iis sii ppi mm$ $$$
2	iss	6	iis sii ppi mm$ $$$
3	mmi	1	iss iis sii ppi iss iis sii ppi mm$ $$$
4	ppi	5	iss iis sii ppi mm$ $$$
5	ppi	9	mm$ $$$
6	$$$	0	mmi iss iis sii ppi iss iis sii ppi mm$ $$$
7	sii	4	ppi iss iis sii ppi mm$ $$$
8	sii	8	ppi mm$ $$$
9	iis	3	sii ppi iss iis sii ppi mm$ $$$
10	iis	7	sii ppi mm$ $$$

Space of the index

- The sparse SA + text takes $\frac{n}{d} \cdot \log \frac{n}{d} + n \log \sigma$ bits
- For $d = \frac{1}{2} \log_\sigma n$ we get

$$\begin{aligned} \frac{2n}{\log_\sigma n} \cdot \log \left(\frac{2n}{\log_\sigma n} \right) &\overset{\sigma \leq n}{\leq} \frac{2n}{\log_\sigma n} \cdot \log n \\ &= 2n \frac{\log_2 n}{\log_\sigma n} = 2n \frac{\log_2 \sigma^{\log_\sigma n}}{\log_\sigma n} \\ &= 2n \frac{\log_\sigma n \cdot \log_2 \sigma}{\log_\sigma n} = 2n \log_2 \sigma \end{aligned}$$

- I.e. the sparse SA takes less than two times the size of the original text

Matching

- First consider a pattern P with $|P| = m \geq d$
- Only occurrences of P at suffixes of the form $i \cdot d$ can be found with the sparse suffix array

Example

$P =$`ippi` can not be found, but its suffix `ppi` (SA-range $[7..8]$)

Idea: Split the pattern in a prefix $P[0..i]$ and suffix $P[i+1..n'-1]$ for all $i < s$.

Index for small patterns

- Handle the case $|P| < d$
- Build a generalized suffix tree (GST) over the collection C of all distinct meta characters

```
                                                  1
      0    1 2    3 4    5 6    7 8    9 0    1 2    3
C = $$$ # iis # iss # mm$ # mmi # ppi # sii #
```

- Length $|C|$ of C: $2 \cdot \sigma^d + 1$; for $d = \frac{1}{2} \log_\sigma n$ we get at most

$$|C| = 2\sqrt{n} + 1$$

- *GST* contains at most $2 \cdot |C|$ nodes
- For each leaf with corresponds to a meta character M_i we store an increasing list L_i of occurrences of M_i in T'

Index for small patterns

- Total size of lists L_i is $|T'|$; so we need at most $\frac{n}{d} \cdot \log \frac{n}{d}$ bits (which is $O(n \log \sigma)$ bits for $d = \frac{1}{2} \log_\sigma n$)
- For each leaf which represents a string s of a meta character we store pointers to all L_is whose meta-characters have s as a suffix
- As each meta character has d suffixes the total number of pointers is bounded by $O(d \cdot \sigma^d)$ which is $o(n)$ for $d = \frac{1}{2} \log_\sigma n$
- Matching: Get node v with path label P. For each leaf in subtree of v chase pointers to L_is and report.