Text Indexing

Lecture 13: Recap and Q&A

Florian Kurpicz

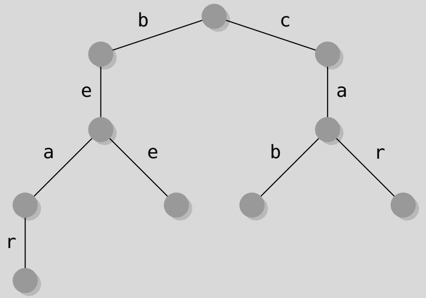
Warning

This is just a very succinct overview.
Please refer to the lecture slides for more details.

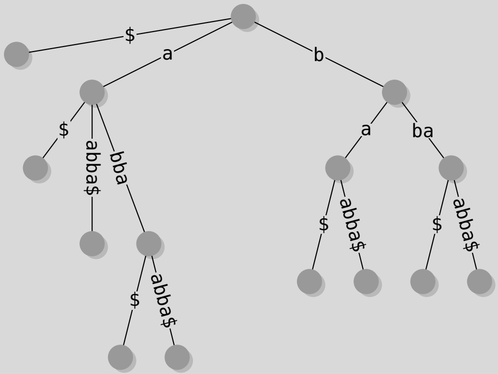
Tries & Suffix Trees

Trie Representations

- different trie representations
- space-time trade-off



Suffix Tree (Compact Trie)



Suffix Array

Suffix Array

Given a text T of length n, the **suffix array** (SA) is a permutation of $[1..n]$, such that for $i \leq j \in [1..n]$

$$T[SA[i]..n] \leq T[SA[j]..n]$$

	1	2	3	4	5	6	7	8	9	10	11	12	13
T	a	b	a	b	c	a	b	c	a	b	b	a	$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
LCP	0	0	1	2	2	5	0	2	1	1	4	0	3

SAIS

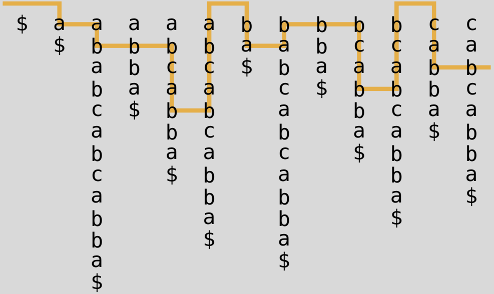
- linear time suffix array construction
- induced copying and recursion
  - classification
  - sorting special suffixes
  - inducing other suffixes

SA Construction in EM

- Prefix Doubling
- DC3

LCP-Array & LCE-Queries

	1	2	3	4	5	6	7	8	9	10	11	12	13
T	a	b	a	b	c	a	b	c	a	b	b	a	$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
LCP	0	0	1	2	2	5	0	2	1	1	4	0	3



- speed up pattern matching in suffix array
- suffix tree construction
- compression

Longest Common Extensions

- lcp-value between any suffix
- scan or RMQ
- Rabin-Karp fingerprints
- string synchronizing sets

Compression

Entropy

Given a text T of length n over an alphabet $\Sigma = [1, \sigma]$ and its histogram *Hist*, then

$$H_k = (1/n) \sum_{S \in \Sigma^k} |T_S| \cdot H_0(T_S)$$

Huffman Codes

- variable length codes
- more frequent characters get shorter codes
- canonical Huffman-codes
- Shannon-Fano codes can be worse, but
- are still optimal

LZ77

$T =$ abababbbbaba$

- $f_1 =$ a
- $f_2 =$ b
- $f_3 =$ abab
- $f_4 =$ bbb
- $f_5 =$ aba
- $f_6 =$ $

LZ78

$T =$ abababbbbaba$

- $f_1 =$ a
- $f_2 =$ b
- $f_3 =$ ab
- $f_4 =$ abb
- $f_5 =$ bb
- $f_6 =$ aba
- $f_7 =$ $

Burrows-Wheeler Transform

Burrows-Wheeler Transform

Given a text T of length n and its suffix array SA, for $i \in [1, n]$ the **Burrows-Wheeler transform** is

$$BWT[i] = \begin{cases} T[SA[i] - 1] & SA[i] > 1 \\ \$ & SA[i] = 1 \end{cases}$$

	1	2	3	4	5	6	7	8	9	10	11	12	13
T	a	b	a	b	c	a	b	c	a	b	b	a	$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
BWT	a	b	$	c	c	b	b	a	a	a	a	b	b

LF-Mapping

Given a BWT, its C-array, and its $rank$-Function, then

$$LF(i) = C[BWT[i]] + rank_{BWT[i]}(i)$$

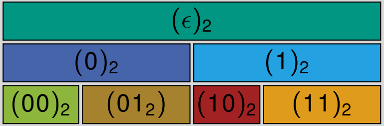
- transform back to text
- used in backwards search

Compression using BWT

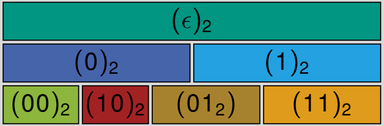
- move-to-front
- run-length compression

Wavelet Tree

Wavelet Tree



Wavelet Matrix



- generalize rank and select to alphabets of size > 2
-

Compression

- build over text compressed with canonical Huffman codes

Bit Vectors

- rank and select queries on bit vectors in $O(1)$ time and $o(n)$ space

FM-Index & r-Index

Function *BackwardsSearch*($P[1..n], C, rank$)**:**

$s = 1, e = n$
for $i = m, \dots, 1$ **do**
  $s = C[P[i]] + rank_{P[i]}(s-1) + 1$
  $e = C[P[i]] + rank_{P[i]}(e)$
  if $s > e$ **then**
    return $\emptyset$
return $[s, e]$

FM-Index

- use (compressed wavelet tree for rank)
- compress bit vectors further

r-Index

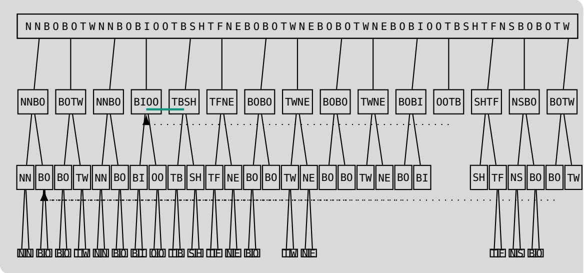
- store lots of arrays
- containing information for each run
- size proportional to number of runs
- queries become harder

Compressed Indices

Karlsruhe Institute of Technology

Block Tree

- answer rank and select queries
- size proportional to number of LZ-factors



Number of Runs and LZ-Factors

Let T be a text of length n, then

$$r(T) \in O(z(T) \lg^2 n)$$

Inverted Index

Karlsruhe Institute of Technology

```
1 The old night keeper keeps the keep in the town
2 In the big old house in the big old gown
3 The house in the town had the big old keep
4 Where the old night keeper never did sleep
5 The night keeper keeps the keep in the night
6 And keeps in the dark and sleeps in the light
```

term t	f_t	$L(t)$
and	1	[6]
big	2	[2, 3]
dark	1	[6]
...	...	...
had	1	[3]
house	2	[2, 3]
in	5	[1, 2, 3, 5, 6]
...	...	...

Encodings

- unary/ternary encoding
- Fibonacci encoding
- Elias-δ/γ encoding
- Golomb encoding

List Interseciong

- binary/exponential search
- two levels

Document Retrieval

Document Listing

- optimal with document array and chain array

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	A	T	A	#	T	A	A	A	#	T	A	T	A	#	$
SA	15	14	4	9	13	3	8	7	6	11	1	12	2	5	10
DA	0	3	1	2	3	1	2	2	2	3	1	3	1	2	3
CA	0	0	0	0	2	3	4	7	8	5	6	10	11	9	12

- $P = TA$