Projekt Algorithm Engineering
Themenvorstellung

Uwe Baier

Institut für theoretische Informatik
Universität Ulm

Sommersemester 2019

Inhaltsübersicht

Einführung

Thema 1: Runminimierung in einer Multi-String BWT

Thema 2: Kantenminimierung in DeBruijn - Graphen

Thema 3: Tunneln einer eXtended BWT

Zusammenfassung und Ziele

Burrows Wheeler Transformation

Geschichte

- reversible Texttransformation [Burrows and Wheeler, 1994]
- ehemaliges Einsatzgebiet: Datenkompression

BWT heute

- zum Teil noch Einsatz in der Datenkompression,
  z.B. `bzip2` [Seward, 1996]
  Problem: langsame Dekompressionsgeschwindigkeit
- Vielfältiger Einsatz in der Bioinformatik,
  z.B. `BWA` für read alignment [Li and Durbin, 2010]
- Einsatz in komprimierter Volltextindizierung,
  z.B. FM-Index [Ferragina and Manzini, 2005]

BWT in der Zukunft

- Fortschritte in der Volltextindizierung [Gagie et al., 2018]
- Fortschritte bei Kompressionsraten [Baier, 2018]

BWT

"Die BWT L *ist ein String, der durch das konkatenieren der zyklisch vorhergehenden Zeichen aller lexikographisch sortierten Suffixe eines Strings S entsteht."*

BWT Generierung für $S =$ `easypeasy$`

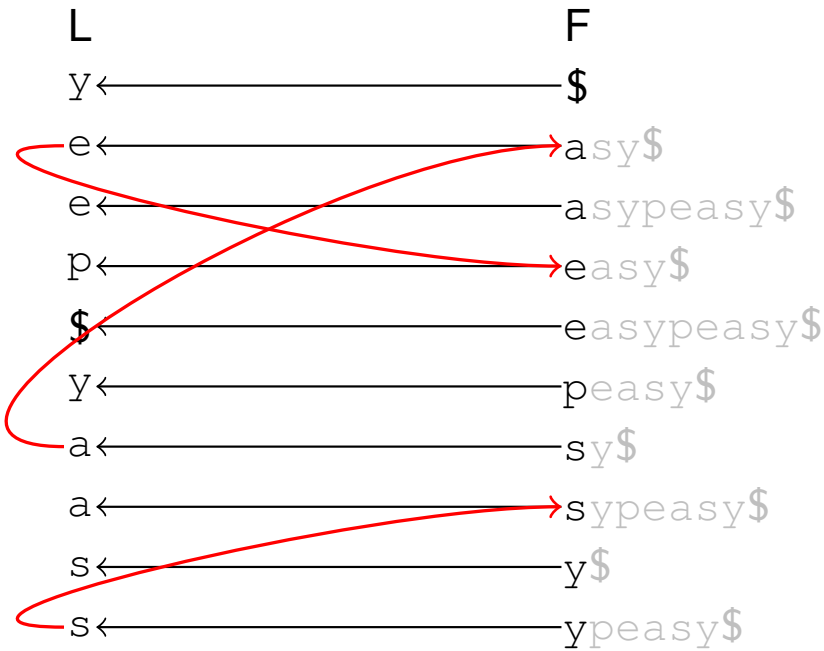
	Suffixe
y	$
s	y$
a	sy$
e	asy$
p	easy$
y	peasy$
s	ypeasy$
a	sypeasy$
e	asypeasy$
$	easypeasy$

sortieren →

L	sortierte Suffixe
y	$
e	asy$
e	asypeasy$
p	easy$
$	easypeasy$
y	peasy$
a	sy$
a	sypeasy$
s	y$
s	ypeasy$

BWT - Rückwärtsschritt

- F - Spalte: erste Spalte der sortierten Suffixe
  Berechenbar durch die Sortierung der Zeichen in L
- *k*-tes Vorkommen des Zeichens *c* in L korrespondiert mit *k*-tem Vorkommen des Zeichens *c* in F (LF-mapping)



- Eine Tour, bei der dem LF-mapping gefolgt und die besuchten Buchstaben in L nacheinander aufgeschrieben werden ergibt den reversen Ausgangsstring

Multi-String BWT

- Mehrere Strings $S_1, \ldots, S_m$ werden mit Trennsymbol \$ zu neuem String $S := S_1\$S_2\$\cdots\$S_m\$$ konkateniert
- sortierte Suffixe werden nur bis zum Trennsymbol betrachtet
- restliche Definition und Rückwärtsschritt analog

Stringkonkatenation

$S_1 = \texttt{GCA}$ $S_2 = \texttt{GGTGA}$

$S_3 = \texttt{GGTGC}$ $S_4 = \texttt{GG}$

$S = \texttt{GCA\$GGTGA\$GGTGC\$GG\$}$

Multi-String BWT

L	sortierte Suffixe
A	$
A	$
C	$
G	$
C	A$
G	A$
G	C$
G	CA$
G	G$
T	GA$
T	GC$
$	GCA$
$	GG$
$	GGTGA$
$	GGTGC$
G	GTGA$
G	GTGC$
G	TGA$
G	TGC$

Thema 1: Runminimierung in einer Multi-String BWT

- Run: längenmaximaler Substring mit Wiederholung desselben Zeichens

$$\cdots \mathtt{CC}\underbrace{\mathtt{AAAAAAAAAAA}}_{\text{Run mit 11 Zeichen}}\mathtt{TT}\cdots$$

- Run durch Lauflängenkodierung komprimierbar

$$\underbrace{\mathtt{AAAAAA}}_{6=110_2} \quad \rightarrow \quad \underbrace{\mathtt{A10}}_{110_2=6\ \mathtt{A}\text{'s}}$$

- Komprimierbarkeit der BWT hängt von der Anzahl an Runs ab

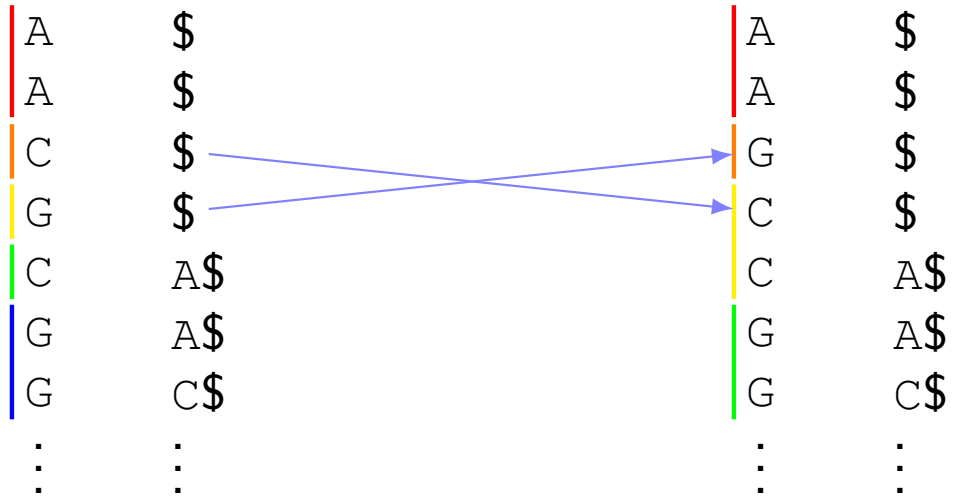
L	sortierte Suffixe
A	$
A	$
C	$
G	$
C	A$
G	A$
G	C$
G	CA$
G	G$
T	GA$
T	GC$
$	GCA$
$	GG$
$	GGTGA$
$	GGTGC$
G	GTGA$
G	GTGC$
G	TGA$
G	TGC$

[Cazaux and Rivals, 2018]

In Multi-String BWT: Zeichen mit gleichem sortiertem Suffix können vertauscht werden

Thema 1: Runminimierung in einer Multi-String BWT

- Ziel: Zeichen mit gleichem Suffix derart vertauschen, dass die Anzahl der Runs minimiert wird



- Eingabe: BWT in Form eines Wavelet Tree [Foschini et al., 2006] sowie Bitvektor mit Markierungen bei Suffixtransitionen
- Algorithmik mittels einfacher Fallunterscheidung [Cazaux and Rivals, 2018]

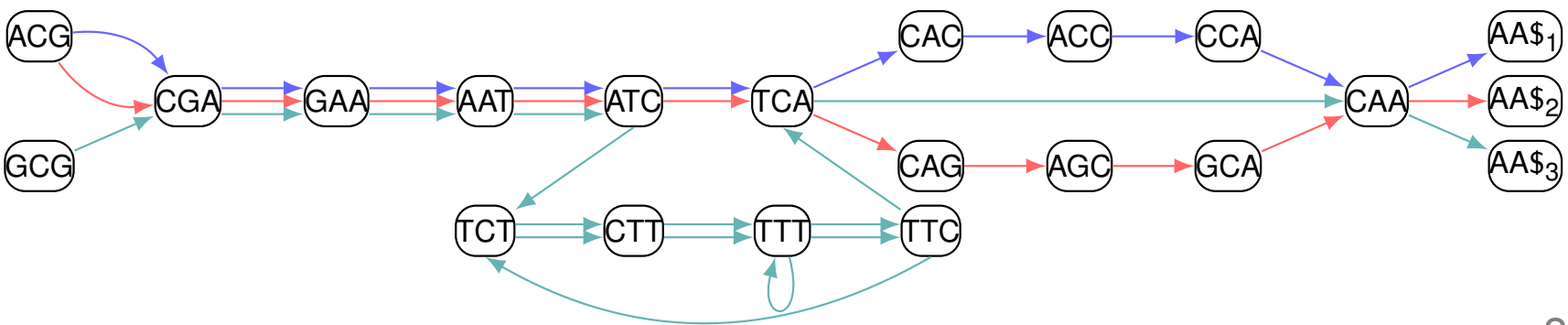
Thema 2: Kantenminimierung in DeBruijn - Graphen

DeBruijn - Graph der Ordnung k

- Grundlage: ein oder mehrere Strings
- Knotenmenge: Menge aller k-langen Teilstrings in S
- Kante von u nach $v \Leftrightarrow$ Teilstrings von u und v überschneiden sich mit genau $k-1$ Zeichen im String
- Multigraph, für jede Überschneidung gibt es eine Kante

DeBruijn - Graph der Ordnung $k = 3$

$S_1 = \texttt{ACGAATCACCAA\$}_1$
$S_2 = \texttt{ACGAATCAGCAA\$}_2$
$S_3 = \texttt{GCGAATCTTTCTTTTCAA\$}_3$

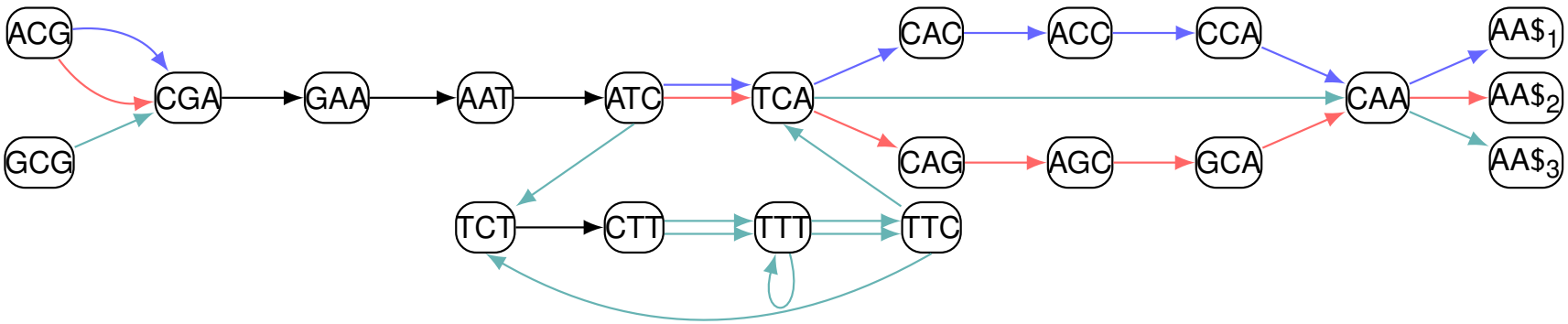


Thema 2: Kantenminimierung in DeBruijn - Graphen

- Kompakte Graphrepräsentation mittels BWT
- Einsatz in Bioinformatik (Genomunterschiede darstellen)

Kantenminimierung [Beller and Ohlebusch, 2016]

Seien u und v zwei Knoten. Ist v einzigster Nachfolger von u und u einzigster Vorgänger von u, so genügt es, nur eine Kante von u nach v zu zeichnen

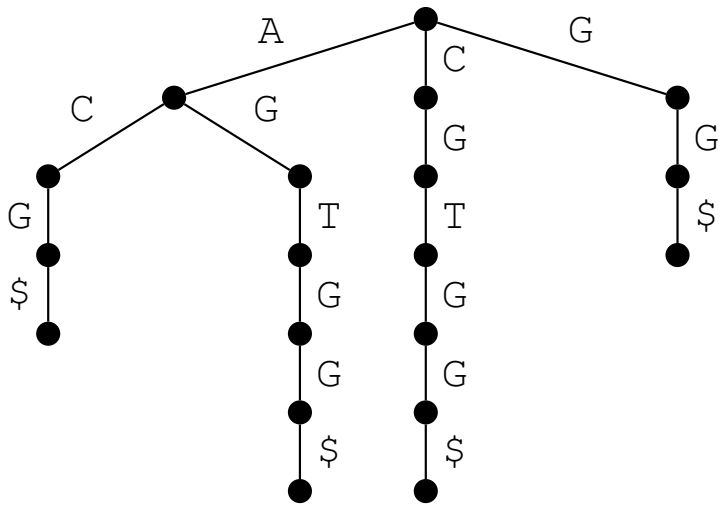


- Ziel: finde k für das Kantenzahl minimiert wird
- Mögliche Einsatzgebiete:
  - Bioinformatik (kompakter Graph $\hat{=}$ viel Information)
  - kompaktere Graphrepräsentation (Tunneln [Baier, 2018])

Thema 3: Tunneln einer eXtended BWT

Trie [Fredkin, 1960]

- baumartige Darstellung einer Menge von Strings
- darstellbar per eXtended BWT
- Einsatz z.B. in Multi-Pattern Suche [Aho and Corasick, 1975]



$S_1 = \texttt{ACG}$ $S_2 = \texttt{AGTGG}$

$S_3 = \texttt{CGTGG}$ $S_4 = \texttt{GG}$

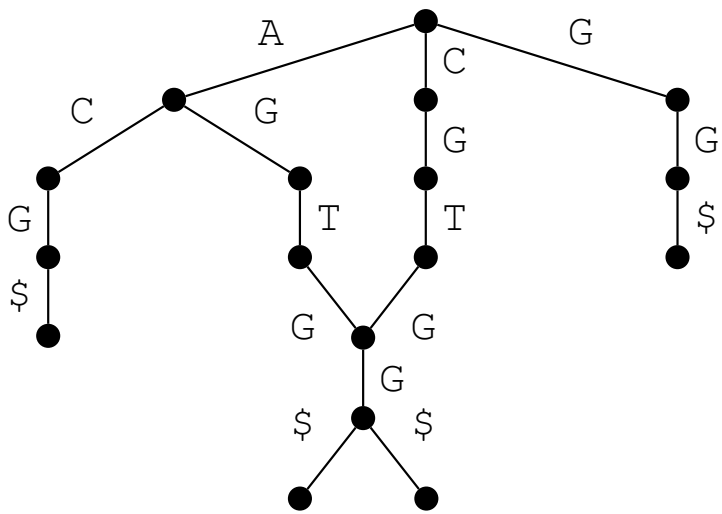
eXtended BWT

last	L	reverser Kontext
0	A	ε
0	C	
1	G	
0	C	A
1	G	
1	G	C
1	G	CA
1	G	G
1	T	GA
1	T	GC
1	$	GCA
1	$	GG
1	$	GGTGA
1	$	GGTGC
1	G	GTGA
1	G	GTGC
1	G	TGA
1	G	TGC

Thema 3: Tunneln einer eXtended BWT

Tunneln [Baier, 2018]

- sortierte Kontexte mit gleichem Präfix können “kompaktifiziert” werden
- Funktionsumfang bleibt vollständig erhalten



$S_1 = \texttt{ACG}$ $S_2 = \texttt{AGTGG}$

$S_3 = \texttt{CGTGG}$ $S_4 = \texttt{GG}$

eXtended BWT

last	L	reverser Kontext
0	A	ε
0	C	
1	G	
0	C	A
1	G	
1	G	C
1	G	CA
1	G	G
1	T	GA
1	T	GC
1	$	GCA
1	$	GG
1	$	GGTGA
1	$	GGTGC
1	G	GTGA
~~1~~	~~G~~	~~GTGC~~
1	G	TGA
1	G	TGC

Rahmenbedingungen

Bereitgestellte Materialien

- Projektkonzept
  - Grundlagen
  - Beschreibung sowie Pseudocode der zu implementierenden Algorithmen
- Quellcode als Programmiergrundlage (C++)
  - Enthält alle benötigten Basisoperationen
  - Beispiele, wie die Operationen benutzt werden

Aufgaben

- Implementierung (C++)
- Durchführen von Experimenten
- Anfertigen einer Ausarbeitung
  - Problemstellung, Lösung etc. in eigenen Worten
  - Darstellung der Experimente und derer Ergebnisse
- Projektvortrag

Zusammenfassung

- Implementierung + Experimente + Ausarbeitung + Vortrag
- Eigenständiges Arbeiten
- Hilfestellung bei Bedarf
- Auch mehrfache Themenvergabe möglich

Interesse? Einfach melden:

uwe.baier@uni-ulm.de

References I

Aho, A. V. and Corasick, M. J. (1975).

Efficient String Matching: An Aid to Bibliographic Search.

Communications of the ACM, 18(6):333–340.

Baier, U. (2018).

On Undetected Redundancy in the Burrows-Wheeler Transform.

In Annual Symposium on Combinatorial Pattern Matching (CPM 2018), CPM '18, pages 3:1–3:15.

Beller, T. and Ohlebusch, E. (2016).

A representation of a compressed de Bruijn graph for pan-genome analysis that enables search.

Algorithms for Molecular Biology, 11(1).

Burrows, M. and Wheeler, D. J. (1994).

A block-sorting lossless data compression algorithm.

Technical Report 124, Digital Equipment Corporation.

References II

Cazaux, B. and Rivals, E. (2018).

Strong link between BWT and XBW via Aho-Corasick automaton and applications to Run-Length Encoding.

CoRR, abs/1805.10070.

Ferragina, P. and Manzini, G. (2005).

Indexing Compressed Text.

Journal of the ACM, 52(4):552–581.

Foschini, L., Grossi, R., Gupta, A., and Vitter, J. S. (2006).

When Indexing Equals Compression: Experiments with Compressing Suffix Arrays and Applications.

ACM Transactions on Algorithms, 2(4):611–639.

Fredkin, E. (1960).

Trie Memory.

Communications of the ACM, 3(9):490–499.

References III

Gagie, T., Navarro, G., and Prezza, N. (2018).

Optimal-Time Text Indexing in BWT-runs Bounded Space.

In Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '18, pages 1459–1477.

Li, H. and Durbin, R. (2010).

Fast and accurate long-read alignment with Burrows–Wheeler transform.

Bioinformatics, 26(5):589–595.

Seward, J. (1996).

`bzip2` File Compressor.

`http://bzip.org/`.

last visited January 2018.