Burrows-Wheeler Algorithmus

Seminar Kompressionsalgorithmen

Daniel Schiller

RWTH Aachen

02.05.2012

Einführung

- Veröffentlicht 1994 von Michael Burrows
  - Algorithmus basiert auf einer nicht veröffentlichten Arbeit von David Wheeler aus dem Jahre 1983
- Stufenbasierter Block-Sortierungsalgorithmus
- Ein- und Ausgaben der Stufen sind Blöcke

  Beispiel:

  Eingabe: | A | B | C | D | E | F | **Ausgabe:** | G | H | I | J | K | L |

Einführung

- Veröffentlicht 1994 von Michael Burrows
  - Algorithmus basiert auf einer nicht veröffentlichten Arbeit von David Wheeler aus dem Jahre 1983
- Stufenbasierter Block-Sortierungsalgorithmus
- Ein- und Ausgaben der Stufen sind Blöcke

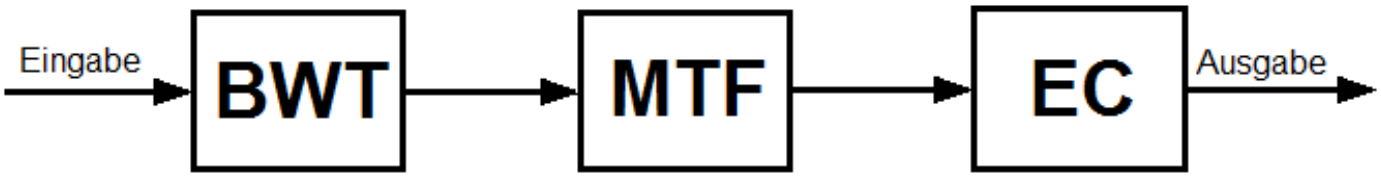
  Beispiel:

  Eingabe: A B C D E F **Ausgabe:** G H I J K L
- Verlustfreie Datenkompression
- Dekompression: Stufen in umgekehrter Reihenfolge abarbeiten

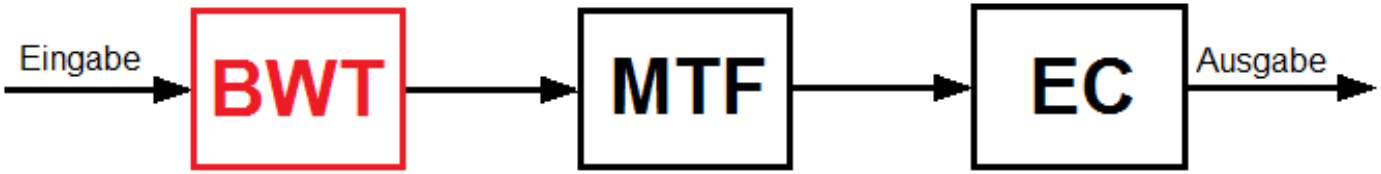
Übersicht

Übersicht

Kompression



Kompression

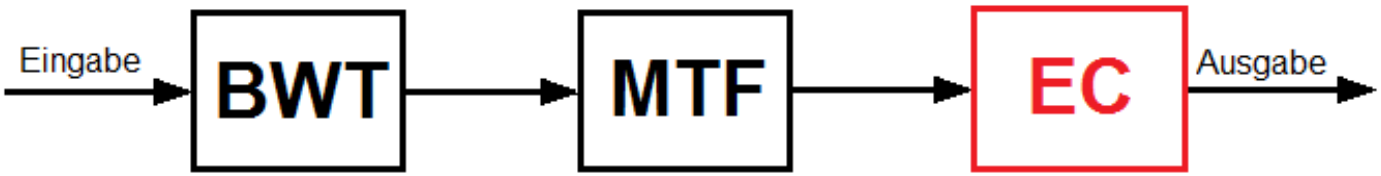


- 1. Stufe: Burrows-Wheeler Transformation
  - Ziel: Identische Zeichen nah beieinander anordnen

Kompression

- 1. Stufe: Burrows-Wheeler Transformation
  - Ziel: Identische Zeichen nah beieinander anordnen
- 2. Stufe: Move-To-Front Transformation
  - Ziel: Zeichen der Eingabe einen globalen Index zuordnen und dabei Zeichen die häufiger vorkommen einen kleineren Index zuordnen

Kompression



- 1. Stufe: Burrows-Wheeler Transformation
  - Ziel: Identische Zeichen nah beieinander anordnen
- 2. Stufe: Move-To-Front Transformation
  - Ziel: Zeichen der Eingabe einen globalen Index zuordnen und dabei Zeichen die häufiger vorkommen einen kleineren Index zuordnen
- 3. Stufe: Entropiekodierung
  - Für Kompression zuständig
  - Normalerweise Huffman Kodierung oder Arithmetische Kodierung

Burrows-Wheeler Transformation

Burrows-Wheeler Transformation

Erinnerung

Ziel ist eine Umsortierung der Eingabe damit gleiche Zeichen nah beieinander angeordnet sind

Burrows-Wheeler Transformation

Erinnerung

Ziel ist eine Umsortierung der Eingabe damit gleiche Zeichen nah beieinander angeordnet sind

Vorgehen

1. Ordne Eingabe n (Länge der Eingabe) mal untereinander und rotiere dabei jede Zeile um ein Zeichen gegenüber der vorherigen nach rechts

Beispiel

Eingabe: PANAMA

Burrows-Wheeler Transformation

Erinnerung

Ziel ist eine Umsortierung der Eingabe damit gleiche Zeichen nah beieinander angeordnet sind

Vorgehen

1. Ordne Eingabe n (Länge der Eingabe) mal untereinander und rotiere dabei jede Zeile um ein Zeichen gegenüber der vorherigen nach rechts
2. Sortiere Zeilen lexikographisch

Beispiel

Eingabe: PANAMA

Burrows-Wheeler Transformation

Erinnerung

Ziel ist eine Umsortierung der Eingabe damit gleiche Zeichen nah beieinander angeordnet sind

Vorgehen

1. Ordne Eingabe n (Länge der Eingabe) mal untereinander und rotiere dabei jede Zeile um ein Zeichen gegenüber der vorherigen nach rechts
2. Sortiere Zeilen lexikographisch
3. Ausgabe: L-Spalte und Index in der sich die ursprüngliche Eingabe befindet

Beispiel

Eingabe: PANAMA

Burrows-Wheeler Transformation

Erinnerung

Ziel ist eine Umsortierung der Eingabe damit gleiche Zeichen nah beieinander angeordnet sind

Vorgehen

1. Ordne Eingabe n (Länge der Eingabe) mal untereinander und rotiere dabei jede Zeile um ein Zeichen gegenüber der vorherigen nach rechts
2. Sortiere Zeilen lexikographisch
3. Ausgabe: L-Spalte und Index in der sich die ursprüngliche Eingabe befindet

Beispiel

Eingabe: PANAMA $\Longrightarrow$ Ausgabe: NPMAAA 5

Move-To-Front Transformation

Move-To-Front Transformation

Erinnerung

Ziel ist es jedem Zeichen der Eingabe einen globalen Indexwert zuzuweisen und dabei Zeichen die häufiger vorkommen einen kleineren Indexwert zuzuweisen

Move-To-Front Transformation

Erinnerung

Ziel ist es jedem Zeichen der Eingabe einen globalen Indexwert zuzuweisen und dabei Zeichen die häufiger vorkommen einen kleineren Indexwert zuzuweisen

Vorgehen

Zusätzlich: Globale Liste $Y = [A, M, N, P]$

Beispiel

Eingabe: NPMAAA 5

Move-To-Front Transformation

Erinnerung

Ziel ist es jedem Zeichen der Eingabe einen globalen Indexwert zuzuweisen und dabei Zeichen die häufiger vorkommen einen kleineren Indexwert zuzuweisen

Vorgehen

Zusätzlich: Globale Liste $Y = [A, M, N, P]$

1. Speicher Indexwert von Y an der sich das erste Zeichen der Eingabe befindet

Beispiel

Eingabe: NPMAAA 5

Move-To-Front Transformation

Erinnerung

Ziel ist es jedem Zeichen der Eingabe einen globalen Indexwert zuzuweisen und dabei Zeichen die häufiger vorkommen einen kleineren Indexwert zuzuweisen

Vorgehen

Zusätzlich: Globale Liste $Y = [A, M, N, P]$

1. Speicher Indexwert von Y an der sich das erste Zeichen der Eingabe befindet
2. Schiebe dieses Zeichen in Y auf Indexposition 0 vor

Beispiel

Eingabe: NPMAAA 5

Move-To-Front Transformation

Erinnerung

Ziel ist es jedem Zeichen der Eingabe einen globalen Indexwert zuzuweisen und dabei Zeichen die häufiger vorkommen einen kleineren Indexwert zuzuweisen

Vorgehen

Zusätzlich: Globale Liste $Y = [A, M, N, P]$

1. Speicher Indexwert von Y an der sich das erste Zeichen der Eingabe befindet
2. Schiebe dieses Zeichen in Y auf Indexposition 0 vor
3. Führe die Schritte 1 und 2 für die restlichen Zeichen der Eingabe nacheinander durch

Beispiel

Eingabe: NPMAAA 5

Move-To-Front Transformation

Erinnerung

Ziel ist es jedem Zeichen der Eingabe einen globalen Indexwert zuzuweisen und dabei Zeichen die häufiger vorkommen einen kleineren Indexwert zuzuweisen

Vorgehen

Zusätzlich: Globale Liste $Y = [A, M, N, P]$

1. Speicher Indexwert von Y an der sich das erste Zeichen der Eingabe befindet
2. Schiebe dieses Zeichen in Y auf Indexposition 0 vor
3. Führe die Schritte 1 und 2 für die restlichen Zeichen der Eingabe nacheinander durch

Beispiel

Eingabe: NPMAAA 5 $\Longrightarrow$ Ausgabe: 233300 5

Nullfolgenkodierung

Nullfolgenkodierung

Ziel

Nullfolgen zu kürzeren Zeichenketten kodieren

Nullfolgenkodierung

Ziel

Nullfolgen zu kürzeren Zeichenketten kodieren

Vorgehen

1. Alle Zeichen der Eingabe die Größer als 0 sind um 1 erhöhen

Beispiel

Eingabe: 233300 5

Nullfolgenkodierung

Ziel

Nullfolgen zu kürzeren Zeichenketten kodieren

Vorgehen

1. Alle Zeichen der Eingabe die Größer als 0 sind um 1 erhöhen
2. Nullfolgen werden durch Kombinationen der Zahlen 0 und 1 kodiert

Beispiel

Eingabe: 233300 5

Nullfolgenkodierung

Ziel

Nullfolgen zu kürzeren Zeichenketten kodieren

Vorgehen

1. Alle Zeichen der Eingabe die Größer als 0 sind um 1 erhöhen
2. Nullfolgen werden durch Kombinationen der Zahlen 0 und 1 kodiert

Beispiel

Eingabe: 233300 5 $\Longrightarrow$ Ausgabe: 34441 5

Übersicht

Dekompression

Dekompression

Erinnerung

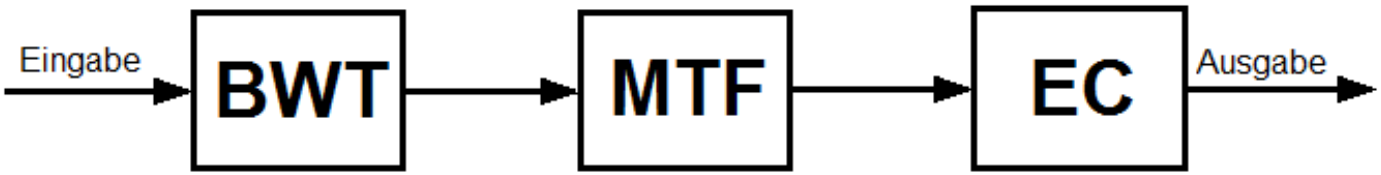
Stufen in umgekehrter Reihenfolge durchführen

Dekompression

Erinnerung

Stufen in umgekehrter Reihenfolge durchführen

Kompression:



Dekompression

Erinnerung

Stufen in umgekehrter Reihenfolge durchführen

Dekompression:

Move-To-Front Rücktransformation

Vorgehen

Zusätzlich: Globale Liste $Y = [A, M, N, P]$

1. Speicher Zeichen welches sich am ersten Indexwert der Eingabe in Y befindet

Beispiel

Eingabe: 233300 5

Move-To-Front Rücktransformation

Vorgehen

Zusätzlich: Globale Liste $Y = [A, M, N, P]$

1. Speicher Zeichen welches sich am ersten Indexwert der Eingabe in Y befindet
2. Schiebe dieses Zeichen in Y auf Indexposition 0 vor

Beispiel

Eingabe: 233300 5

Move-To-Front Rücktransformation

Vorgehen

Zusätzlich: Globale Liste $Y = [A, M, N, P]$

1. Speicher Zeichen welches sich am ersten Indexwert der Eingabe in Y befindet
2. Schiebe dieses Zeichen in Y auf Indexposition 0 vor
3. Führe die Schritte 1 und 2 für die restlichen Indexwerte der Eingabe nacheinander durch

Beispiel

Eingabe: 233300 5

Move-To-Front Rücktransformation

Vorgehen

Zusätzlich: Globale Liste $Y = [A, M, N, P]$

1. Speicher Zeichen welches sich am ersten Indexwert der Eingabe in Y befindet
2. Schiebe dieses Zeichen in Y auf Indexposition 0 vor
3. Führe die Schritte 1 und 2 für die restlichen Indexwerte der Eingabe nacheinander durch

Beispiel

Eingabe: 233300 5 $\Longrightarrow$ Ausgabe: NPMAAA 5

Burrows-Wheeler Rücktransformation

Vorgehen

Beispiel

Eingabe: NPMAAA 5

Burrows-Wheeler Rücktransformation

Vorgehen

1. Sortiere L-Spalte alphabetisch $\Rightarrow$ F-Spalte

Beispiel

Eingabe: NPMAAA 5

Burrows-Wheeler Rücktransformation

Vorgehen

1. Sortiere L-Spalte alphabetisch $\Rightarrow$ F-Spalte
2. Speicher Zeichen an Eingabeindex in F-Spalte

Beispiel

Eingabe: NPMAAA 5

Burrows-Wheeler Rücktransformation

Vorgehen

1. Sortiere L-Spalte alphabetisch $\Rightarrow$ F-Spalte
2. Speicher Zeichen an Eingabeindex in F-Spalte
3. Gehe in L-Spalte zu dem Zeichen an Index i, das
   - identisch zum letzten gespeicherten Zeichen
   - vor dem genauso viele identische Zeichen wie in der F-Spalte liegen

Beispiel

Eingabe: NPMAAA 5

Burrows-Wheeler Rücktransformation

Vorgehen

1. Sortiere L-Spalte alphabetisch $\Rightarrow$ F-Spalte
2. Speicher Zeichen an Eingabeindex in F-Spalte
3. Gehe in L-Spalte zu dem Zeichen an Index i, das
   - identisch zum letzten gespeicherten Zeichen
   - vor dem genauso viele identische Zeichen wie in der F-Spalte liegen
4. Speicher Zeichen an Index i in F-Spalte

Beispiel

Eingabe: NPMAAA 5

Burrows-Wheeler Rücktransformation

Vorgehen

1. Sortiere L-Spalte alphabetisch $\Rightarrow$ F-Spalte
2. Speicher Zeichen an Eingabeindex in F-Spalte
3. Gehe in L-Spalte zu dem Zeichen an Index i, das
   - identisch zum letzten gespeicherten Zeichen
   - vor dem genauso viele identische Zeichen wie in der F-Spalte liegen
4. Speicher Zeichen an Index i in F-Spalte
5. Wiederhole Schritt 3 und 4, bis i gleich dem Eingabeindex ist

Beispiel

Eingabe: NPMAAA 5

Burrows-Wheeler Rücktransformation

Vorgehen

1. Sortiere L-Spalte alphabetisch $\Rightarrow$ F-Spalte
2. Speicher Zeichen an Eingabeindex in F-Spalte
3. Gehe in L-Spalte zu dem Zeichen an Index i, das
   - identisch zum letzten gespeicherten Zeichen
   - vor dem genauso viele identische Zeichen wie in der F-Spalte liegen
4. Speicher Zeichen an Index i in F-Spalte
5. Wiederhole Schritt 3 und 4, bis i gleich dem Eingabeindex ist

Beispiel

Eingabe: NPMAAA 5 $\Longrightarrow$ Ausgabe: PANAMA

Übersicht

Rückblick Burrows-Wheeler Transformation

Rückblick Burrows-Wheeler Transformation

Vorgehen

1. Ordne Eingabe n (Länge der Eingabe) mal untereinander und rotiere dabei jede Zeile um ein Zeichen gegenüber der vorherigen nach rechts
2. Sortiere Zeilen lexikographisch

Beispiel

Eingabe: PIZZA

Rückblick Burrows-Wheeler Transformation

Vorgehen

1. Ordne Eingabe n (Länge der Eingabe) mal untereinander und rotiere dabei jede Zeile um ein Zeichen gegenüber der vorherigen nach rechts
2. Sortiere Zeilen lexikographisch
3. Ausgabe: L-Spalte und Index in der sich die ursprüngliche Eingabe befindet

Beispiel

Eingabe: PIZZA $\Longrightarrow$ Ausgabe: ZPAZI 3

Rückblick Burrows-Wheeler Transformation

Vorgehen

1. Ordne Eingabe n (Länge der Eingabe) mal untereinander und rotiere dabei jede Zeile um ein Zeichen gegenüber der vorherigen nach rechts
2. Sortiere Zeilen lexikographisch
3. Ausgabe: L-Spalte und Index in der sich die ursprüngliche Eingabe befindet

Beispiel

Eingabe: PIZZA $\Longrightarrow$ Ausgabe: ZPAZI 3

Die Sortierung hat sich verschlechtert!

Rückblick Burrows-Wheeler Transformation

Vorgehen

1. Ordne Eingabe n (Länge der Eingabe) mal untereinander und rotiere dabei jede Zeile um ein Zeichen gegenüber der vorherigen nach rechts
2. Sortiere Zeilen lexikographisch
3. Ausgabe: L-Spalte und Index in der sich die ursprüngliche Eingabe befindet

Beispiel

Eingabe: PIZZA $\Longrightarrow$ Ausgabe: ZPAZI 3

Die Sortierung hat sich verschlechtert!

Erinnerung

Ziel der Burrows-Wheeler Transformation ist eine Umsortierung der Eingabe damit gleiche Zeichen nah beieinander angeordnet sind

Burrows-Wheeler Algorithmus

Warum gute Komprimierung?

Burrows-Wheeler Algorithmus

Warum gute Komprimierung?

- Normalerweise werden ganze Sätze oder Zeichenketten komprimiert

Burrows-Wheeler Algorithmus

Warum gute Komprimierung?

- Normalerweise werden ganze Sätze oder Zeichenketten komprimiert
- Beispiel: Deutsche Sprache
  - Wörter wie "und" werden häufiger verwendet

Burrows-Wheeler Algorithmus

Warum gute Komprimierung?

- Normalerweise werden ganze Sätze oder Zeichenketten komprimiert
- Beispiel: Deutsche Sprache
  - Wörter wie "und" werden häufiger verwendet
- Rotationen die mit "nd" beginnen werden untereinander sortiert
  - In der L-Spalte entstehen Folgen des Buchstaben u

Übersicht

Burrows-Wheeler Algorithmus

Problem: Matrix zu Speicherintensiv

Burrows-Wheeler Algorithmus

Problem: Matrix zu Speicherintensiv

Lösung:

Suffix Array

Ein Suffix Array speichert die Suffixe eines Strings in lexikographischer Reihenfolge ab

Burrows-Wheeler Algorithmus

Problem: Matrix zu Speicherintensiv

Lösung:

Suffix Array

Ein Suffix Array speichert die Suffixe eines Strings in lexikographischer Reihenfolge ab

- Rotationen des Strings statt Suffixe benutzen

Burrows-Wheeler Algorithmus

Problem: Matrix zu Speicherintensiv

Lösung:

Suffix Array

Ein Suffix Array speichert die Suffixe eines Strings in lexikographischer Reihenfolge ab

- Rotationen des Strings statt Suffixe benutzen
- Es existieren Algorithmen die ein Suffix-Array in linearer Laufzeit konstruieren
  - Beispiel: Skew-Algorithmus von Kärkkäinen und Sanders

Übersicht

BZIP2

- Bzip2 wurde von Julian Seward entwickelt und ist ein frei verfügbares Datenkompressionsprogramm
- Bzip2 verwendet den Burrows-Wheeler Algorithmus mit der Huffmankodierung als Entropiekodierer

BZIP2

- Bzip2 wurde von Julian Seward entwickelt und ist ein frei verfügbares Datenkompressionsprogramm
- Bzip2 verwendet den Burrows-Wheeler Algorithmus mit der Huffmankodierung als Entropiekodierer
- Bzip2 ist der Nachfolger von Bzip
  - Bzip verwendete die Arithmetische Kodierung als Entropiekodierer
  - Arithmetische Kodierung unterliegt Patenten weshalb Bzip nicht weiterentwickelt wurde

BZIP2

- Bzip2 wurde von Julian Seward entwickelt und ist ein frei verfügbares Datenkompressionsprogramm
- Bzip2 verwendet den Burrows-Wheeler Algorithmus mit der Huffmankodierung als Entropiekodierer
- Bzip2 ist der Nachfolger von Bzip
  - Bzip verwendete die Arithmetische Kodierung als Entropiekodierer
  - Arithmetische Kodierung unterliegt Patenten weshalb Bzip nicht weiterentwickelt wurde
- Dekodierung deutlich schneller als Kodierung

Vergleich Kompressionsverfahren

Kompressionsraten: progc

gzip: 32,7 %
7z: 31,8 %
bzip2: 31,7 %

Vergleich Kompressionsverfahren

Kompressionsraten: progc

gzip: 32,7 %
7z: 31,8 %
bzip2: 31,7 %

Kompressionsraten: book1

gzip: 39,1 %
7z: 34,0 %
bzip2: 30,3 %

Vergleich Kompressionsverfahren

Kompressionsraten: progc

gzip: 32,7 %
7z: 31,8 %
bzip2: 31,7 %

Kompressionsraten: book1

gzip: 39,1 %
7z: 34,0 %
bzip2: 30,3 %

Kompressionsraten: obj1

gzip: 47,7 %
7z: 44,0 %
bzip2: 50,2 %

Zusammenfassung

- Der Burrows-Wheeler Algorithmus ist ein Stufenbasierter Block-Sortierungsalgorithmus
- Burrows-Wheeler und Move-To-Front Transformation sorgen für Kompressionsfreundlichere Eingabedaten
- Entropiekodierung ist für Kompression zuständig

Zusammenfassung

- Der Burrows-Wheeler Algorithmus ist ein Stufenbasierter Block-Sortierungsalgorithmus
- Burrows-Wheeler und Move-To-Front Transformation sorgen für Kompressionsfreundlichere Eingabedaten
- Entropiekodierung ist für Kompression zuständig
- Bei der Dekompression werden die Stufen des Algorithmus in umgekehrter Reihenfolge abgearbeitet

Zusammenfassung

- Der Burrows-Wheeler Algorithmus ist ein Stufenbasierter Block-Sortierungsalgorithmus
- Burrows-Wheeler und Move-To-Front Transformation sorgen für Kompressionsfreundlichere Eingabedaten
- Entropiekodierung ist für Kompression zuständig
- Bei der Dekompression werden die Stufen des Algorithmus in umgekehrter Reihenfolge abgearbeitet
- Ein Datenkompressionsprogramm das auf dem Burrows-Wheeler Algorithmus aufbaut ist Bzip2