Aufgaben zum Tut am 02.07.2007

Thomas Pajor

9. Juli 2007

Aufgabe 1.

Betrachten Sie die Zeichenkette

$$\alpha := mississippi$$

(a) Wenden Sie eine Burrows-Wheeler-Transformation auf α an.

(b) Wenden Sie eine Move-to-Front Kodierung auf α an. Dabei sei das Alphabet $\mathcal{X} = [i, m, p, s]$.

(c) Dekodieren Sie die Zeichenkette $(\beta, k) = (tttpeeee, 4)$ mit der Burrows-Wheeler-Transformation.

Lösung.

(a) Um eine Burrows-Wheeler-Transformation anwenden zu können, müssen wir zunächst die zyklische Matrix A aufbauen. Dies führt zu

$$A = \begin{pmatrix} m & i & s & s & i & s & s & i & p & p & i \\ i & s & s & i & s & s & i & p & p & i & m \\ s & s & i & s & s & i & p & p & i & m & i \\ s & i & s & s & i & p & p & i & m & i & s \\ i & s & s & i & p & p & i & m & i & s & s \\ s & s & i & p & p & i & m & i & s & s & i \\ s & i & p & p & i & m & i & s & s & i & s \\ i & p & p & i & m & i & s & s & i & s & s \\ p & p & i & m & i & s & s & i & s & s & i \\ p & i & m & i & s & s & i & s & s & i & p \\ i & m & i & s & s & i & s & s & i & p & p \end{pmatrix}$$

Nun werden die Zeilen in nicht-absteigender Reihenfolge sortiert. Dies liefert die Matrix B mit

$$B = \begin{pmatrix} i & m & i & s & s & i & s & s & i & p & p \\ i & p & p & i & m & i & s & s & i & s & s \\ i & s & s & i & p & p & i & m & i & s & s \\ i & s & s & i & s & s & i & p & p & i & m \\ m & i & s & s & i & s & s & i & p & p & i \\ p & i & m & i & s & s & i & s & s & i & p \\ p & p & i & m & i & s & s & i & s & s & i \\ s & i & p & p & i & m & i & s & s & i & s \\ s & i & s & s & i & p & p & i & m & i & s \\ s & s & i & p & p & i & m & i & s & s & i \\ s & s & i & s & s & i & p & p & i & m & i \end{pmatrix}$$

Die letzte Spalte der Matrix ist die transformierte Zeichenkette der BW-Transformation. Zusammen mit dem Zeilenindex, an deren Stelle α steht, liefert die Transformation die Ausgabe

$$(\beta, k) = (pssmipissii, 5).$$

Man sieht an dieser Stelle, dass die Burrows-Wheeler-Transformation nicht besonders gut „gewirkt“ hat[1], deswegen empfiehlt es sich hier durchaus eine Move-to-Front Kodierung anzuwenden *bevor* man die Zeichenkette durch die Lauflängenkodierung jagt.

(b) Wir wenden eine Move-to-Front Kodierung auf die original Zeichenkette

$$\alpha = mississippi$$

an. Dafür benutzen wir zunächst die Tabelle

i	m	p	s
0	1	2	3

Das erste Zeichen α_1 ist ein m, das heißt, wir geben eine 1 aus (Der Index in der Tabelle ist ja schließlich 1), und bewegen das m in de Tabelle an die Spitze. Die neue Tabelle ist

m	i	p	s
0	1	2	3

Das nächste Zeichen ist $\alpha_2 = i$. Dies liefert den Index 1 in der aktuellen Tabelle, und führt uns zur neuen Tabelle wie folgt.

i	m	p	s
0	1	2	3

Für das folgende s ergibt sich ein Index von 3 und die Tabelle

[1] Die volle Wirkung kommt erst bei längeren Zeichenketten gut zur geltung.

s	i	m	p
0	1	2	3

Man sieht, dass alle Zeichen nach hinten geschoben wurden. Für das erneute Auftreten des nächsten s ist die Ausgabe wieder eine 0, und die Tabelle bleibt unverändert. Nun folgt wieder ein i, und wir sehen, dass wir das i an Stelle 1 in der Tabelle haben. Wir geben also eine 1 aus und verändern die Tabelle zu

i	s	m	p
0	1	2	3

Nun folgt wieder ein s; Wir geben also eine 1 aus und führen das Spielchen fort. Man sieht schon, dass relativ häufig geringe Indizes ausgegeben werden. Da unser Ziel ist niedrigen Indizes kurze Codewörter zuzuordnen erhoffen wir uns davon eine insgesamt kleinere Nominalinformation der kodierten Zeichenkette.

Das Gesamtergebnis ergibt schließlich $\beta = 11301101301$.

(c) Wir wollen schlussendlich einmal den Dekodierungsalgorithmus der Burrows-Wheeler Transformation anwenden. Als Eingabe erhalten wir das Tupel $(\beta, k) = (ttte peeee, 4)$[2], und möchten die ursprüngliche Zeichenkette zurückgewinnen.

Bekannt ist, dass β die letzte Spalte der Matrix B ist. Da die Zeilen der Matrix B sortiert sind, können wir die erste Spalte rekonstruieren, indem wir β sortieren. Das heißt, für die erste Spalte ergibt sich

$$S = eeeeepttt.$$

Schreiben wir die Vektoren mal mit ihren Indizes in eine Tabelle so erhalten wir

index	S	β
1	e	t
2	e	t
3	e	t
4	e	e
5	e	p
6	p	e
7	t	e
8	t	e
9	t	e

Nun müssen wir den Dekodierungsalgorithmus anwenden, um die fehlenden Einträge $\alpha_2 \cdots \alpha_{n-1}$ zu rekonstruieren. Unser Startindex ist 4, damit erhalten wir als erstes Ausgabezeichen $S_4 = e$. Das Zeichen S_4 entspricht dem vierten Vorkommen eines es (von oben gezählt). Das vierte Vorkommen eines es im Vektor β ist an der Stelle β_8 zu finden. Das nächste Zeichen für die Ausgabe ist also $S_8 = t$. Dies entspricht dem zweiten Vorkommen eines t in S, und so weiter. Wenn man sich den Ablauf visualisiert ergibt sich das Diagramm aus Abbildung 1.

[2] Das zum Beispiel aus dem Dekoder der Move-to-Front Kodierung stammt

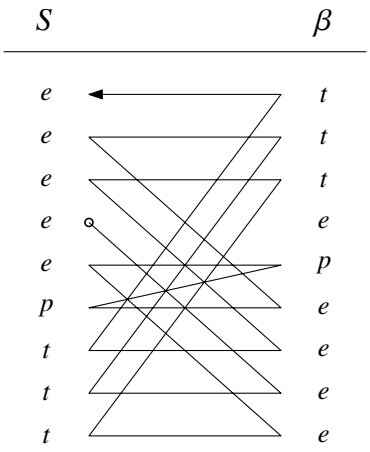

Abbildung 1: Rücktransformation mittels Burrows-Wheeler mit Startindex $i = 4$.

Somit ist die gesuchte Ausgabe

$$\alpha = etepetete.$$

Aufgabe 2.

Gegeben sei folgende Nachricht.

$$\alpha := rokokokostuem$$

(a) Kodieren Sie die Nachricht mittels LZW. Neue Wörterbucheinträge können ab dem Index 256 erzeugt werden.

(b) Dekodieren Sie die aus Aufgabe (a) erhaltene Nachricht mit dem LZW-Verfahren.

Lösung.

(a) Anwendung des Lempel-Ziv-Welch Algorithmus auf die Zeichenkette $\alpha = rokokokostuem$ führt zunächst zu der Tabelle 1. Die Ausgabe ist damit

$$\beta = rok\$257\$259stuem$$

und damit immerhin um drei Zeichen kürzer als die Eingabe. Man sieht hier auch schon den Nachteil der LZW-Kompression: Nämlich, dass wir ziemlich lange nichts gewinnen, bis sich manche Zeichenketten mehrfach wiederholen. Jedoch wie auch bei der Burrows-Wheeler-Transformation zeigt sich erst bei langen Zeichenketten mit häufig wiederholenden Wörtern (wie beispielsweise einem Text) die volle Wirkung des LZW-Verfahrens. Das Wörterbuch, das wir „im Speicher“ aufgebaut haben, braucht nicht mitgeschickt zu werden, da es bei der Dekompression automatisch rekonstruiert werden kann.

Eingabe	Ausgabe	Wörterbuch
r	*r*	256 = *ro*
o	*o*	257 = *ok*
k	*k*	258 = *ko*
o	\$257	259 = *oko*
k		
o	\$259	260 = *okos*
k		
o		
s	*s*	261 = *st*
t	*t*	262 = *tu*
u	*u*	263 = *ue*
e	*e*	264 = *em*
m	*m*	

Tabelle 1: LZW-Kodierung der Zeichenkette *rokokokostuem*

(b) Das Kodieren in LZW ist ziemlich straight-forward. Der interessantere Teil ist sicherlich die Dekodierung und das gleichzeitige Aufbauen des Wörterbuches. Da der Wörterbucheintrag beim Kodieren immer aus der Ausgabe a und dem *nächsten* Zeichen – sagen wir α – besteht, können wir bei der Dekompression die Wörterbucheinträge erst einen Schritt später erzeugen. Denn ist zum Beispiel in unserem Eingabestrom folgende Situation $\cdots a\$423\cdots$, so wissen wir zum Zeitpunkt, zu dem wir das a lesen noch nicht wie das Zeichen, das *nach* dem a kommt. Schließlich hängt das ja von dem Inhalt in \$423 ab.

Eingabe	Ausgabe	Wörterbuch
r	*r*	
o	*o*	256 = *ro*
k	*k*	257 = *ok*
\$257	*ok*	258 = *ko*
\$259		
s		
t		
u		
e		
m		

Tabelle 2: LZW-Dekodierung der Zeichenkette β = *rok*\$257\$259*stuem* bis zum Auftreten des $K\Omega K$-Problems.

Tabelle 2 zeigt den Anfang der Dekodierung bis zum Auftreten des $K\Omega K$-Problems. Wir müssen den Wörterbucheintrag 259 aufsuchen, den es aber noch garnicht gibt. Das heißt beim Erstellen der LZW-Kodierung wurde der Eintrag 259 direkt im folgenden Schritt nach der Erstellung benutzt. Da wir jetzt einen Schritt später das Wörterbuch aufbauen würde

der Eintrag in diesem Schritt erstellt werden. Das letzte Zeichen von \$259 entspricht ja dem Zeichen, was *nach* der Ausgabe des vorhergehenden Codewortes (in unserem Fall also *ok*) steht. Und da wir den Eintrag 259 bei der Kodierung in dem Schritt erstellt haben, als wir *ok* kodiert haben, und es danach gleich benutzt haben, muss dieses Zeichen dem ersten Zeichen von *ok* entsprechen. Der gesuchte Eintrag ist als $259 = oko$. Nun können wir die Tabelle vervollständigen, und erhalten Tabelle 3.

Eingabe	**Ausgabe**	**Wörterbuch**
r	r	
o	o	$256 = ro$
k	k	$257 = ok$
\$257	ok	$258 = ko$
\$259	oko	$259 = oko$
s	s	$260 = oko s$
t	t	$261 = st$
u	u	$262 = tu$
e	e	$263 = ue$
m	m	$264 = em$

Tabelle 3: Komplette Dekodierung der Zeichenkette.

Die Ausgabe ist also wieder

$$\alpha = rokokokostuem.$$