



Multimedia Systems

WS 2009/2010

Exercise Course, January 26, 2010

Dipl.-Inf. Simon Schwantzer

University of Kaiserslautern, Germany
Integrated Communication Systems Lab

Email: schwantzer@informatik.uni-kl.de



Exercise 6.1

1. Encode the following string with the Lempel-Ziv algorithm (LZ77): `ababababab`

Step	Pos	Match	Next	Output
1	1	-	a	<0,0,a>
2	2	-	b	<0,0,b>
3	3	abababa	a	<2,7,b>

→ Finding a match with help of the look ahead buffer enables a very efficient encoding for strings with recurring patterns.



Exercise 6.2

- The following algorithm describes a Lempel-Ziv-Welch (LZW) encoding:
 1. Initialize the directory with all symbols of the used alphabet.
 2. Set $pos = 0$.
 3. Encode the string s with a directory index, whereas x is the character following s in the input.
 4. Add sx to the directory, whereas x is the character following s in the input.
 5. Set pos to the position of x ; continue at 3 or end.

Exercise 6.2

- Encoding: abababa
 - Created directory

Index	String
1	a
2	b
3	ab
4	ba
5	aba

- Output: 1, 2, 3, 5

Step	Directory	pos	s	x	Output
1.	1: a 2: b	n/a	n/a	n/a	n/a
2.		0	n/a	n/a	n/a
3.		0	a	n/a	1
4.	3: ab	0	a	b	1
5.		1	a	b	1
3.		1	b	b	1, 2
4.	4: ba	1	b	a	1, 2
5.		2	b	a	1, 2
3.		2	ab	a	1, 2, 3
4.	5: aba	2	ab	a	1, 2, 3
5.		4	ab	a	1, 2, 3
3.		4	aba	a	1, 2, 3, 5
4.+5.		4	aba	n/a	1, 2, 3, 5

Exercise 6.2

- Decoding algorithm (incomplete):
 1. Initialize the directory with all symbols of the used alphabet.
 2. Set i to next index to decode or stop.
 3. Add $D(i)$ (the directory entry at position i) to the output. Set $x =$ first character of $D(i)$.
 4. If s is not empty: Append sx to the end of the directory.
 5. Set $s = D(i)$. Go to step 2.
- Note:
 - s is the directory entry of the last cycle.
 - x is the first character of the directory entry of the current cycle.
 - sx is equivalent in encoding and decoding.

Exercise 6.2

- Decoding
 - Initial directory

Index	String
1	a
2	b

- Encoded string
1, 2, 3, 5

Step	Directory	i	s	x	Output
1.	1: a 2: b	n/a	n/a	n/a	n/a
2.		1	n/a	n/a	n/a
3.		1	n/a	a	a
5.		1	a	a	a
2.		2	a	a	a
3.		2	a	b	ab
4.	3: ab	2	a	b	ab
5.		2	b	b	ab
2.		3	b	b	ab
3.		3	b	a	abab
4.	4: ba	3	b	a	abab
5.		3	ab	a	abab
2.		5	ab	a	abab
3.		5	ab	?	D(5)!?



Exercise 6.2

- Decoding algorithm (with exception handling):
 1. Initialize the directory with all symbols of the used alphabet.
 2. Set i to next index to decode or stop.
 3. If
 - $i \leq \text{size of } D$:
 - Add $D(i)$ to the output.
 - Set $x = \text{first character of } D(i)$.
 - $i > \text{size of } D$:
 - Set $x = \text{first character of } s$.
 - Add sx to the output.
 4. If s is not empty: Append sx to the end of the directory.
 5. Set $s = D(i)$. Go to step 2.

Exercise 6.2

- Decoding
 - Initial directory

Index	String
1	a
2	b

- Encoded string
(Indices)
1, 2, 3, 5

Step	Directory	i	s	x	Output
1.	1: a 2: b	n/a	n/a	n/a	n/a
2.		1	n/a	n/a	n/a
3.		1	n/a	a	a
5.		1	a	a	a
2.		2	a	a	a
3.		2	a	b	ab
4.	3: ab	2	a	b	ab
5.		2	b	b	ab
2.		3	b	b	ab
3.		3	b	a	abab
4.	4: ba	3	b	a	abab
5.		3	ab	a	abab
2.		5	ab	a	abab
3.		5	ab	a	abababa
4.	5: aba	5	ab	a	abababa
5.		5	aba	a	abababa