Algorithmen und Datenstrukturen II

SoSe 2010, **3. Aufgabenblatt**, Abgabe 26.05.2010

Gegeben seien die Zeichenketten S_1 = `HASEN_ABER_HABEN_ALLE_ALLELE`
und S_2 = `ANANASSAHNE_AN_NASE`
für Aufgaben 7, 8, und 9.

Aufgabe 7

14 Punkte

a) Zählen Sie separat in S_1 und S_2 die Häufigkeiten der Zeichen und geben Sie diese in je einer Tabelle für S_1 und S_2 an. Versichern Sie Sich, dass Ihre Zählung stimmt, damit Sie die beiden folgenden Teilaufgaben auch richtig lösen können. (2 Punkte)

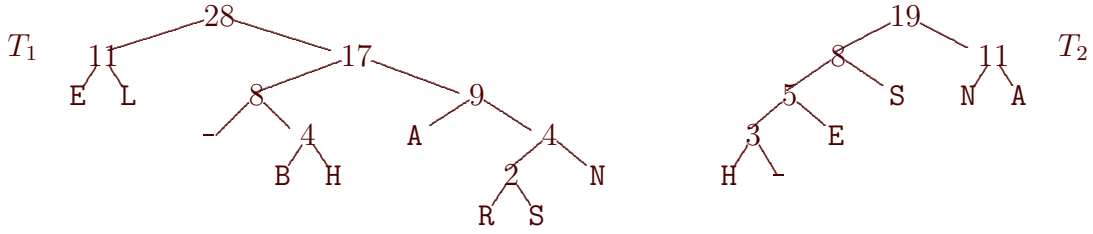
S_1 :

_	A	B	E	H	L	N	R	S
4	5	2	6	2	5	2	1	1

S_2 :

_	A	E	H	N	S
2	6	2	1	5	3

b) Konstruieren Sie anhand der in S_1 und S_2 ermittelten Häufigkeiten die entsprechenden Huffman-Codierungs-Bäume T_1 und T_2. Gibt es mehr als einen Baum mit minimaler Häufigkeit, so wird jeweils derjenige ausgewählt, der das am frühesten im Alphabet erscheinende Zeichen enthält. Das Leerzeichen _ kommt alphabetisch vor `A`. (6 Punkte)



c) Kodieren Sie die Zeichenkette S_1 mit T_1; und kodieren Sie die Zeichenkette S_2 sowohl mit T_1 als auch mit T_2. (6 Punkte)

Codes von T_1: _=100, A=110, B=1010, E=00, H=1011, L=01, N=1111, R=11100, S=11101
S_1 kodiert mit T_1:
1011 110 11101 00 1111 100 110 1010 00 11100 ...
S_2 kodiert mit T_1:
110 1111 110 1111 110 11101 11101 110 ...

Codes von T_2: _=0001, A=11, E=001, H=0000, N=10, S=01
S_2 kodiert mit T_2:
11 10 11 10 11 01 01 11 0000 10 001 0001 11 10 0001 10 11 01 001

Aufgabe 8

8 Punkte

Wenden Sie den Algorithmus LZ77 (Sliding Window Lempel-Ziv ohne Optimierungen, laut Vorlesung) auf S_1 und auf S_2 an. Der Buffer fasse 4 Zeichen. Geben Sie die erzeugte Ausgabe bei Längen 5 und 15 des Dictionary jeweils für die Zeichenketten S_1 und S_2 an. Sie müssen also insgesamt vier Läufe des LZ77 durchführen. (8 Punkte)

S_1, buffer=4, dict=5

```
01  |HASE|N_ABER_HABEN_ALLE_ALLELE     (0,0,H)
02  H|ASEN|_                           (0,0,A)
03  HA|SEN_|A                          (0,0,S)
04  HAS|EN_A|B                         (0,0,E)
05  HASE|N_AB|E                        (0,0,N)
06  HASEN|_ABE|R                       (0,0,_)
07   ASEN_|ABER|_                      (5,1,B)
08     EN_AB|ER_H|H                    (5,1,R)
09       _ABER|_HAB|E                  (5,1,H)
10         BER_H|ABEN|_                (0,0,A)
11          ER_HA|BEN_|A               (0,0,B)
12           R_HAB|EN_A|L              (0,0,E)
13            _HABE|N_AL|L             (0,0,N)
14             HABEN|_ALL|E            (0,0,_)
15              ABEN_|ALLE|_           (5,1,L)
16                EN_AL|LE_A|L         (1,1,E)
17                  _ALLE|_ALL|E       (5,4,E)
18                        _ALLE|LE|    (2,2,eof)
```

S_1, buffer=4, dict=15

```
01  |HASE|N_ABER_HABEN_ALLE_ALLELE     (0,0,H)
02  H|ASEN|_                           (0,0,A)
03  HA|SEN_|A                          (0,0,S)
04  HAS|EN_A|B                         (0,0,E)
05  HASE|N_AB|E                        (0,0,N)
06  HASEN|_ABE|R                       (0,0,_)
07  HASEN_|ABER|_                      (5,1,B)
08  HASEN_AB|ER_HA|B                   (5,1,R)
09  HASEN_ABER|_HAB|E                  (5,1,H)
10  HASEN_ABER_H|ABEN|_                (6,3,N)
11   ASEN_ABER_HABEN|_ALL|E            (11,2,L)
12      N_ABER_HABEN_AL|LE_A|L         (1,1,E)
13        ABER_HABEN_ALLE|_ALL|E       (5,4,E)
14           R_HABEN_ALLE_ALLE|LE|     (2,2,eof)
```

S_2, buffer=4, dict=5

```
01  |ANAN|ASSAHNE_AN_NASE          (0,0,A)
02  A|NANA|S                      (0,0,N)
03  AN|ANAS|S                     (2,3,S)
04   NANAS|SAHN|E                 (1,1,A)
05     NASSA|HNE_|A                (0,0,H)
06      ASSAH|NE_A|N               (0,0,N)
07       SSAHN|E_AN|_              (0,0,E)
08        SAHNE|_AN_|N             (0,0,_)
09         AHNE_|AN_N|A            (5,1,N)
10           NE_AN|_NAS|E          (3,1,N)
11             _AN_N|ASE|          (4,1,S)
12               N_NAS|E|          (0,0,E)
```

S_2, buffer=4, dict=15

```
01  |ANAN|ASSAHNE_AN_NASE          (0,0,A)
02  A|NANA|S                      (0,0,N)
03  AN|ANAS|S                     (2,3,S)
04  ANANAS|SAHN|E                 (1,1,A)
05  ANANASSA|HNE_|A               (0,0,H)
06  ANANASSAH|NE_A|N              (6,1,E)
07  ANANASSAHNE|_AN_|N            (0,0,_)
08  ANANASSAHNE_|AN_N|A           (10,2,_)
09  ANANASSAHNE_AN_|NASE|         (12,3,E)
```

Aufgabe 9

8 Punkte

Wenden Sie die Burrows-Wheeler-Transformation auf S_1 und S_2 an. Geben Sie als Zwischenergebnis die Sortierung der Teilstrings an, entweder in Form der Matrix mit allen rotierten Versionen der Zeichenketten nach alphabetischer Sortierung (vgl. Vorlesung) oder (platzsparender), durch Angabe des Sortierranges bei jedem Zeichen in den Original-Zeichenketten S_1 und S_2. Geben Sie als Endergebnis die permutierten Zeichenketten und die Zeilennummer des Originalblocks an. Die Zeilennummerierung beginne bei 0. (8 Punkte)

S_1:

unsortiert			sortiert		
Index	Anfang	Ende	Index	Anfang	Ende
0	HASEN	E	5	_	N
1	ASEN_	H	16	_	N
2	SEN_A	A	21	_	E
3	EN_A	S	10	_	R
4	N_ABE	E	12	A	H
5	_ABER	N	6	A	_
6	ABER	_	17	A	_
7	BER_	A	22	A	_
8	ER_HA	B	1	A	H
9	R_HAB	E	13	B	A
10	_HABE	R	7	B	A
11	HABEN	_	20	E	L
12	ABEN_	H	27	E	L
13	BEN_	A	25	E	L
14	EN_AL	B	3	E	S
15	N_ALL	E	14	E	B
16	_ALLE	N	8	E	B
17	ALLE	_	11	H	_
18	LLE_	A	0	H	E
19	LE_A	L	19	L	L
20	E_ALL	L	26	L	E
21	_ALLEL	E	24	L	L
22	ALLEL	_	18	L	A
23	LLELE	A	23	L	A
24	LELEH	L	4	N	E
25	ELEHA	L	15	N	E
26	LEHASS	E	9	R	E
27	EHASE	L	2	S	A

Tabelle “spaltenweise” erzeugen, output: Letzte Spalte (codierter Text) und Index=18.

S_2:

unsortiert			sortiert		
Index	Anfang	Ende	Index	Anfang	Ende
0	ANAN	E	11	_A	E
1	NAN	A	14	_N	N
2	ANAS	N	7	AHN	S
3	NASS	A	12	AN_	_
4	ASS	N	0	ANAN	E
5	SSA	A	2	ANAS	N
6	SAH	S	16	ASE	N
7	AHN	S	4	ASS	N
8	HNE	A	10	E_	N
9	NE_	H	18	EA	S
10	E_A	N	8	H	A
11	_AN	E	13	N_	A
12	AN_	_	1	NAN	A
13	N_N	A	15	NASE	_
14	_NA	N	3	NASS	A
15	NASE	_	9	NE	H
16	ASE	N	6	SA	S
17	SEA	A	17	SE	A
18	EAN	S	5	SS	A

Kodierter Text: Letzte Spalte, Index=4