EXERCISE 6: COMPRESSION

Multimedia Databases SS 23

Prof. Dr. Harald Kosch/ Prof. Dr. Mario Döller

Tutor: Kanishka Ghosh Dastidar, Alaa Alhamzeh

Task 1: LZW Encoding

- Lempel–Ziv–Welch (LZW) is a universal, lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch.
- Published by Welch in 1984 as an improved implementation of the LZ78 algorithm published by Lempel and Ziv.
- Used in the GIF image format.
- A large English text file can typically be compressed via LZW to about half its original size.

Task 1: LZW Encoding

1. The Prefix is empty and the dictionary is initialized with all characters
2. Read the next character z from the input stream
3. Is "‘ Prefix + z ’" in the dictionary?

   Yes: Prefix = Prefix + z

   No:

   Output Code **for** Prefix

   put Prefix + z **in the dictionary**

   Prefix = z
4. Has the end of the input stream been reached?

   No: go to step 2

   Yes: If the Prefix is not empty, output the corresponding code

Task 1: LZW Encoding

z	P	P+z	P+z in Dictionary?			
			Yes	No		
			New P	Output	Dictionary	New P
					1:m,2:i,3:s,4:p	
M		m	M			
I	m	mi		1	5:mi	i
S	i	is		2	6:is	s
S	s	ss		3	7:ss	s
I	s	si		3	8:si	i
S	i	is	is			
S	is	iss		6	9:iss	s
i	s	si	si			
P	si	sip		8	10:sip	p
P	p	pp		4	11:pp	p
i	p	pi		4	12:pi	i
	i			2		

Task 1: LZW Decoding

Dictionary initialised with all occurring characters
Code = first code from the input stream (always one character)
Output the entry for code and
Store code in OldCode
Code = next code in the input stream
Prefix = character of OldCode
Code in dictionary?
Yes:
- Output the character of Code
- Character = first character of Code
- Enter Prefix+Character in dictionary

No:
- Character = first character of OldCode
- Enter prefix+character in dictionary AND output it

As long as characters still exist, go to 4.

Task 1: LZW Decoding

C	oldCode	Dict?	Präfix =Character(oldCode)	Character(Dict/!Dict) =Character(C/oC)[0]	Dictionary 1:m,2:i,3:s,4:p	Output(Dict/!Dict) Character(C)/Prefix+z
1	1					m
2	1	y	m	i	5:mi	i
3	2	y	i	s	6:is	s
3	3	y	s	s	7:ss	s
6	3	y	s	i	8:si	is
8	6	y	is	s	9:iss	si
4	8	y	si	p	10:sip	p
4	4	y	p	p	11:pp	p
2	4	y	p	i	12:pi	i

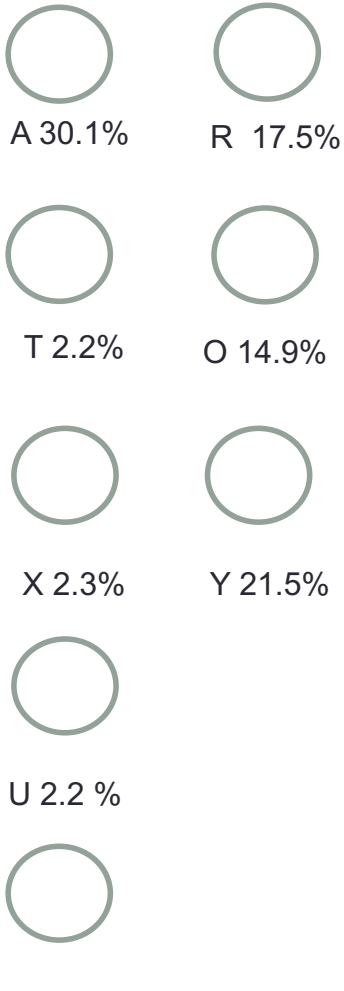
Task 2: Huffman-Code

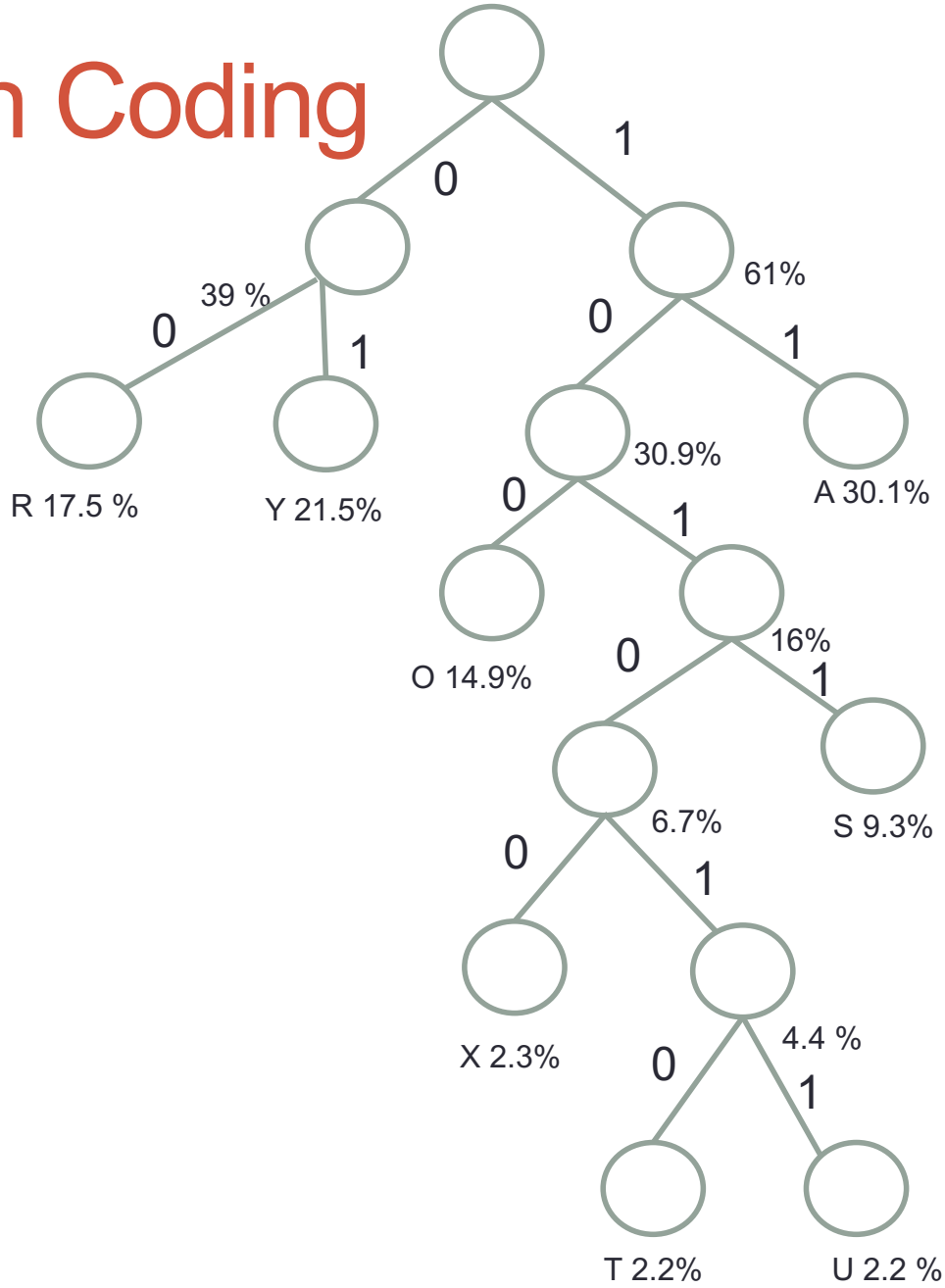
- **Generate the Huffman Code for the following set of characters**

A	R	Y	O	S	T	X	U
30.1%	17.5%	21.5%	14.9%	9.3%	2.2%	2.3%	2.2%

- Create a leaf node for each symbol and add it to the priority queue.
- While there is more than one node in the queue:
  Remove the two nodes of highest priority (lowest probability) from the queue
  Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.
  Add the new node to the queue.
- The remaining node is the root node and the tree is complete.

Task 2: Huffman Coding





Task 2: Huffman Coding

A	R	Y	O	S	T	X	U
30.1%	17.5%	21.5%	14.9%	9.3%	2.2%	2.3%	2.2%

A	**11**
Y	**01**
R	**00**
O	**100**
S	**1011**
X	**10100**
T	**101010**
U	**101011**