2.a

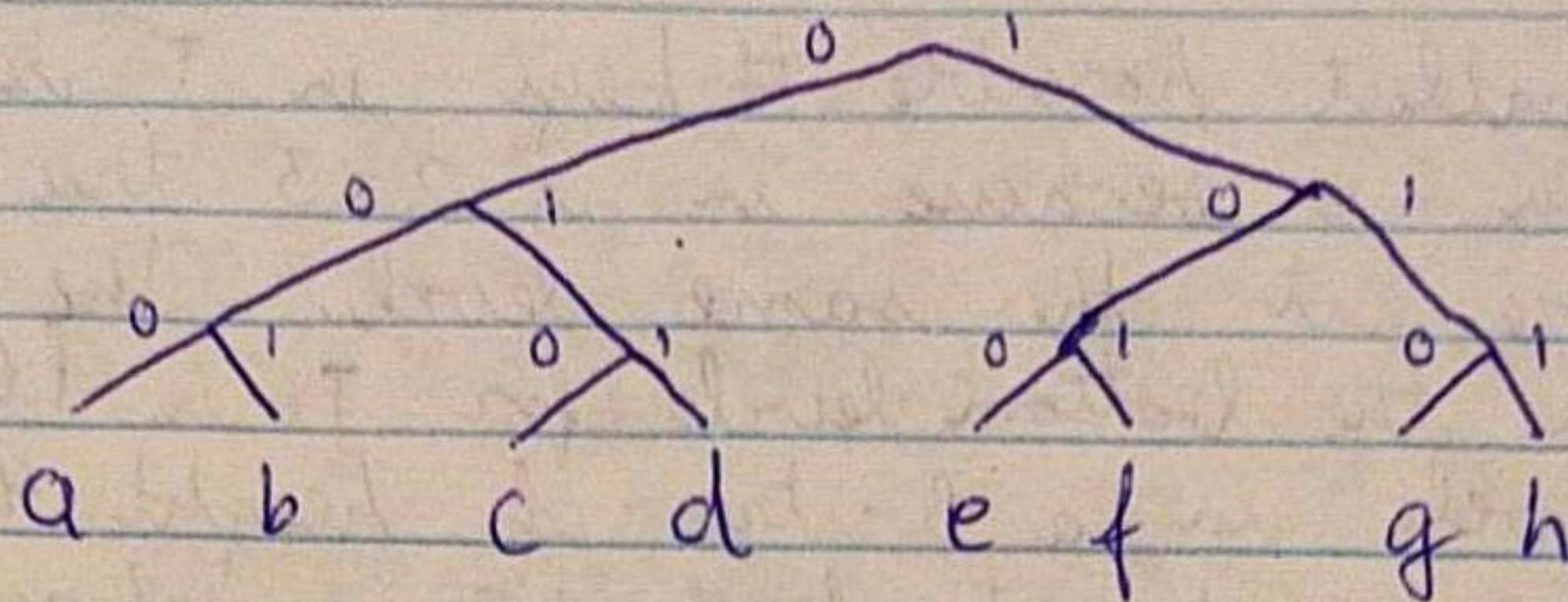

The trie structure with leaves a, b, c, d, e, f, g, h.

b   The trie doesn't decode z uniquely because combination of characters could have the same count but be added at different times. Eg. 11010010 can be decoded as

11 010 10 & 1101 010

g t e   r t

Three diff. encodings of z:

i   110 11 01 010 010 01 11 10 1101 010

a   g   f   t   t f   g   e   r   t

ii   1101 10 10 10 010 01 11 10 11 010 10

r   e   e   e   t f   g   e   g   t   e

iii   1101 10 10 10 010 01 11 10 1101 010

r   e   e   e   t f   g   e   r   t

betty_bought_a_bit_of_better_butter

w:	b	e	t	t	y	_	b	o	u	g	h	t	_	a		_	_b	i	t	t_	o	f		_	_b	e	et	t	e	r		_	_b	u	t	tt	e	er
k:	e	t	t	y	_	b	o	u	g	h	t	_	a		_	b	i	t		_	o	f		_	b	e	t	t	e	r		_	b	u	t	t	e	r

| o: | **98** | **101** | **116** | **116** | **121** | **32** | **98** | **111** | **117** | **103** | **104** | **116** | **32** | **97** | | **133** | **105** | | **139** | **111** | **102** | | **133** | | **129** | **116** | **101** | **114** | | **133** | **117** | | **130** | | **150** |
| | b | e | t | t | y | _ | b | o | u | g | h | t | _ | a | | _b | i | | t_ | o | f | | _b | | et | t | e | r | | _b | u | | tt | | er |

| i: | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | | 142 | 143 | | 144 | 145 | 146 | | 147 | | 148 | 149 | 150 | 151 | | 152 | 153 | | 154 |
| | be | et | tt | ty | y_ | _b | bo | ou | ug | gh | ht | t_ | _a | a_ | | _bi | it | | t_o | of | f_ | | _be | | ett | te | er | r_ | | _bu | ut | | tte |

Σs:

32	_	103	g	114	r
97	a	104	h		...
98	b	105	i	116	t
	...		...	117	u
101	e	111	o		...
102	f		...	121	y

betty_bought_a_bit_of_better_butter$ 36

i	Cyclic shifts	j	Sorted	C
0	betty_bought_a_bit_of_better_butter$	35	$betty_bought_a_bit_of_better_butter	r
1	etty_bought_a_bit_of_better_butter$b	12	_a_bit_of_better_butter$betty_bought	t
2	tty_bought_a_bit_of_better_butter$be	21	_better_butter$betty_bought_a_bit_of	f
3	ty_bought_a_bit_of_better_butter$bet	14	_bit_of_better_butter$betty_bought_a	a
4	y_bought_a_bit_of_better_butter$bett	5	_bought_a_bit_of_better_butter$betty	y
5	_bought_a_bit_of_better_butter$betty	28	_butter$betty_bought_a_bit_of_better	r
6	bought_a_bit_of_better_butter$betty_	18	_of_better_butter$betty_bought_a_bit	t
7	ought_a_bit_of_better_butter$betty_b	13	a_bit_of_better_butter$betty_bought_	_
8	ught_a_bit_of_better_butter$betty_bo	22	better_butter$betty_bought_a_bit_of_	_

#	Rotation	#	Sorted rotation	L
9	ght_a_bit_of_better_butter$betty_bou	0	betty_bought_a_bit_of_better_butter$	$
10	ht_a_bit_of_better_butter$betty_boug	15	bit_of_better_butter$betty_bought_a_	_
11	t_a_bit_of_better_butter$betty_bough	6	bought_a_bit_of_better_butter$betty_	_
12	_a_bit_of_better_butter$betty_bought	29	butter$betty_bought_a_bit_of_better_	_
13	a_bit_of_better_butter$betty_bought_	26	er_butter$betty_bought_a_bit_of_bett	t
14	_bit_of_better_butter$betty_bought_a	33	er$betty_bought_a_bit_of_better_butt	t
15	bit_of_better_butter$betty_bought_a_	23	etter_butter$betty_bought_a_bit_of_b	b
16	it_of_better_butter$betty_bought_a_b	1	etty_bought_a_bit_of_better_butter$b	b
17	t_of_better_butter$betty_bought_a_bi	20	f_better_butter$betty_bought_a_bit_o	o
18	_of_better_butter$betty_bought_a_bit	9	ght_a_bit_of_better_butter$betty_bou	u
19	of_better_butter$betty_bought_a_bit_	10	ht_a_bit_of_better_butter$betty_boug	g
20	f_better_butter$betty_bought_a_bit_o	16	it_of_better_butter$betty_bought_a_b	b
21	_better_butter$betty_bought_a_bit_of	19	of_better_butter$betty_bought_a_bit_	_
22	better_butter$betty_bought_a_bit_of_	7	ought_a_bit_of_better_butter$betty_b	b
23	etter_butter$betty_bought_a_bit_of_b	27	r_butter$betty_bought_a_bit_of_bette	e
24	tter_butter$betty_bought_a_bit_of_be	34	r$betty_bought_a_bit_of_better_butte	e
25	ter_butter$betty_bought_a_bit_of_bet	11	t_a_bit_of_better_butter$betty_bough	h
26	er_butter$betty_bought_a_bit_of_bett	17	t_of_better_butter$betty_bought_a_bi	i
27	r_butter$betty_bought_a_bit_of_bette	25	ter_butter$betty_bought_a_bit_of_bet	t
28	_butter$betty_bought_a_bit_of_better	32	ter$betty_bought_a_bit_of_better_but	t
29	butter$betty_bought_a_bit_of_better_	24	tter_butter$betty_bought_a_bit_of_be	e
30	utter$betty_bought_a_bit_of_better_b	31	tter$betty_bought_a_bit_of_better_bu	u
31	tter$betty_bought_a_bit_of_better_bu	2	tty_bought_a_bit_of_better_butter$be	e
32	ter$betty_bought_a_bit_of_better_but	3	ty_bought_a_bit_of_better_butter$bet	t
33	er$betty_bought_a_bit_of_better_butt	8	ught_a_bit_of_better_butter$betty_bo	o
34	r$betty_bought_a_bit_of_better_butte	30	utter$betty_bought_a_bit_of_better_b	b
35	$betty_bought_a_bit_of_better_butter	4	y_bought_a_bit_of_better_butter$bett	t

C = rtfayrt__$___ttbbougb_beehitteuetobt

```
w:  r  t  f  a  y  r  rt  _  _  $  _  __  _  t  t  b  b  o  u  g  b  _  b  e  e  h  i  t  tt  e  u  e  t  o  b  t
k:  t  f  a  y  r        _  _  $  _  _   _  t  t  b  b  o  u  g  b  _  b  e  e  h  i  t  t   e  u  e  t  o  b  t
```

o:	114	116	102	97	121		128	32	32	3		134	32	116	116	98	98	111	117	103	98	32	98	101	101	104	105		139	101	117	101	116	111	98	116
	r	t	f	a	y		rt	_	_	$		__	_	t	t	b	b	o	u	g	b	_	b	e	e	h	i		tt	e	u	e	t	o	b	

i:	128	129	130	131	132		133	134	135	136		137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152		153	154	155	156	157	158	159
	rt	tf	fa	ay	yr		rt_	__	_$	$_		___	_t	tt	tb	bb	bo	ou	ug	gb	b_	_b	be	ee	eh	hi	it		tte	eu	ue	et	to	ob	bt

Σs:

3	$	103	g	114	r
32	_	104	h		...
97	a	105	i	116	t
98	b	...		117	u
	...	111	o		...
101	e	...		121	y
102	f				

2e

T = abacadaeafagaha

LZW_encode(T)			=		97	98	97	99	97	100	97	101	97	101	97	102	97	103	97	104	97
C	=	BWT_encode(T)	=	ah$bcdefgaaaaaaa																	
LCW_encode(C)			=		97	104	3	98	99	100	102	102	103	97	137	138	97
LCW_table(C) (part of)			=									aa	aaa	

BWT prior to LZW yields a better compression ratio because
BWT exposes correlations / patterns in T which makes it more compressible

2f

i) It is not guaranteed.
   If we have two words 'a' and 'I', their encodings would be 0 and 1.
   However there is no compression achieved

ii) The best case would be when the word 'shells' is repeated n times
    This is because shells has the most number of letters and the best compressed string (11) (at par with 'she's and 'sells').