AVL Trees, Heaps, Heapsort and Huffman Coding

Tutorial 8

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Question 1

a. Construct a simple binary search tree by inserting the following elements:

18,9,6,33,27,2,1,4,8,3

b. Construct an AVL tree by inserting the following elements:

18,9,6,33,27,2,1,4,8,3

c. Deletion?

Question 1

18,9,6,33,27,2,1,4,8,3

To Ponder

1. At each insertion in the AVL tree, we need the balance factor of all the nodes present in the tree to check for balancedness and perform some operations if the balance is off by more than 1. But how is still the cost of insertion O(log(n))?

To Ponder

1. At each insertion in the AVL tree, we need the balance factor of all the nodes present in the tree to check for balancedness and perform some operations if the balance is off by more than 1. But how is still the cost of insertion O(log(n))?

2. Is there a case where the cost of any operation in an AVL tree would be at least the cost of any operation if the input was stored in a simple binary search tree?

Question 2

a. Construct a heap by inserting the following elements

18,9,6,33,27,2,1,4,8,3

Question 2

a. Construct a heap by inserting the following elements

18,9,6,33,27,2,1,4,8,3

b. Do the following arrays represent a min heap?

A = [2, 3, 6, 8, 10, 15, 18 ,20, 25]
B = [2, 3, 4, 10, 7, 5, 6, 13]
C = [2, 3, 4, 13, 6, 5, 7, 10, 12]

To Ponder

1. Heap vs binary search tree. What should I use?

To Ponder

1. Heap vs binary search tree. What should I use?
    - BST is ordered while heap is not
    - Insertion and deletion times?

To Ponder

1. Heap vs binary search tree. What should I use?

2. Sorting using Heapsort seems to be better than the other sorting algorithms. Then why not use just heapsort in all the applications?

Question 3

Encode the word ‘racoonnookkeeper’ using Huffman coding.

Size of ‘racoonnookkeeper’ is 16.

Question 3

Encode the word ‘racoonnookkeeper’ using Huffman coding.

Size of ‘racoonnookkeeper’ is 16.

Frequency Table:

R	2
A	1
C	1
O	4
N	2
K	2
E	3
P	1

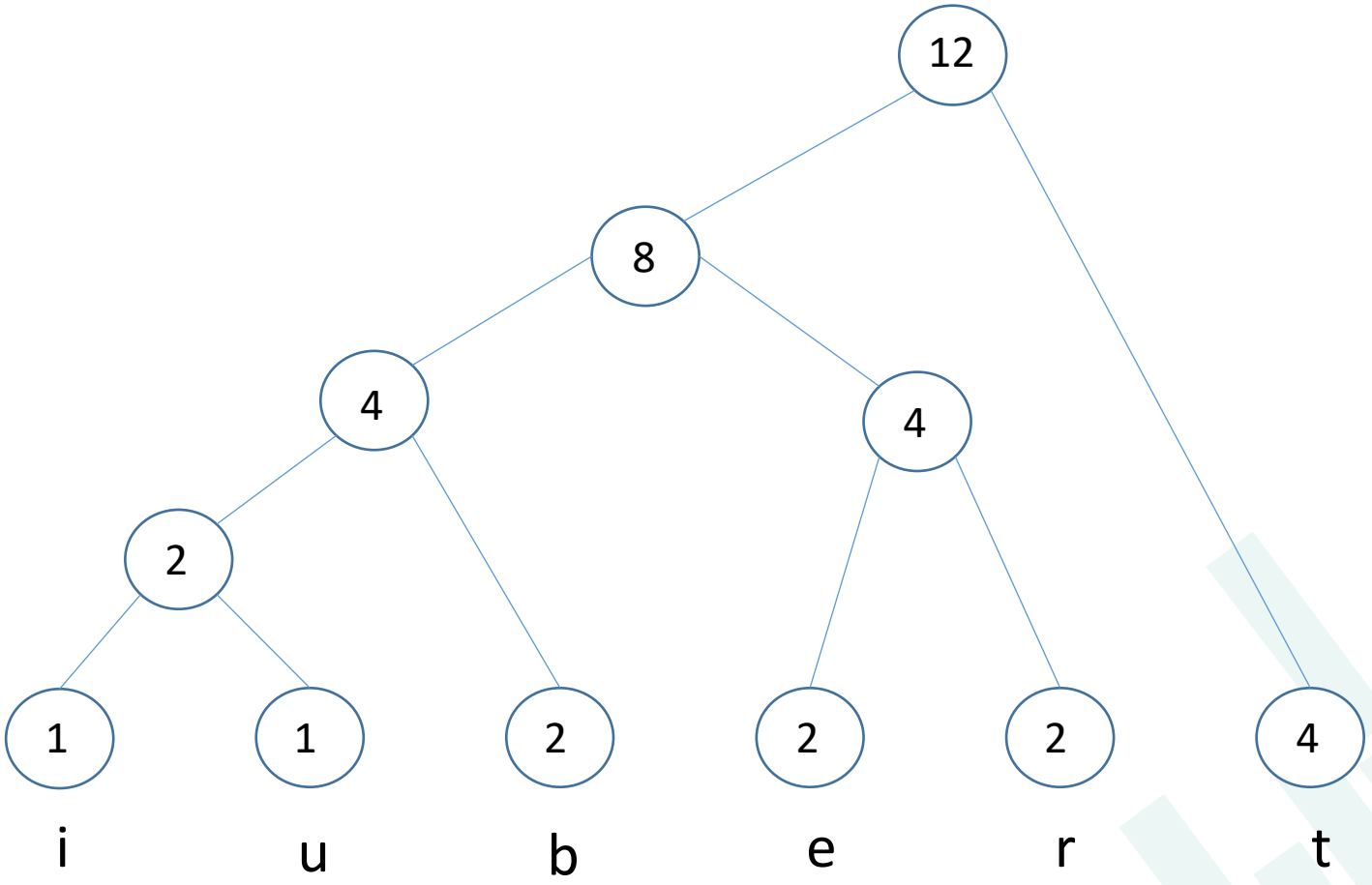
Question 3

R	A	C	O	N	K	E	P
2	1	1	4	2	2	3	1

Question 3

Decode the following Huffman code.

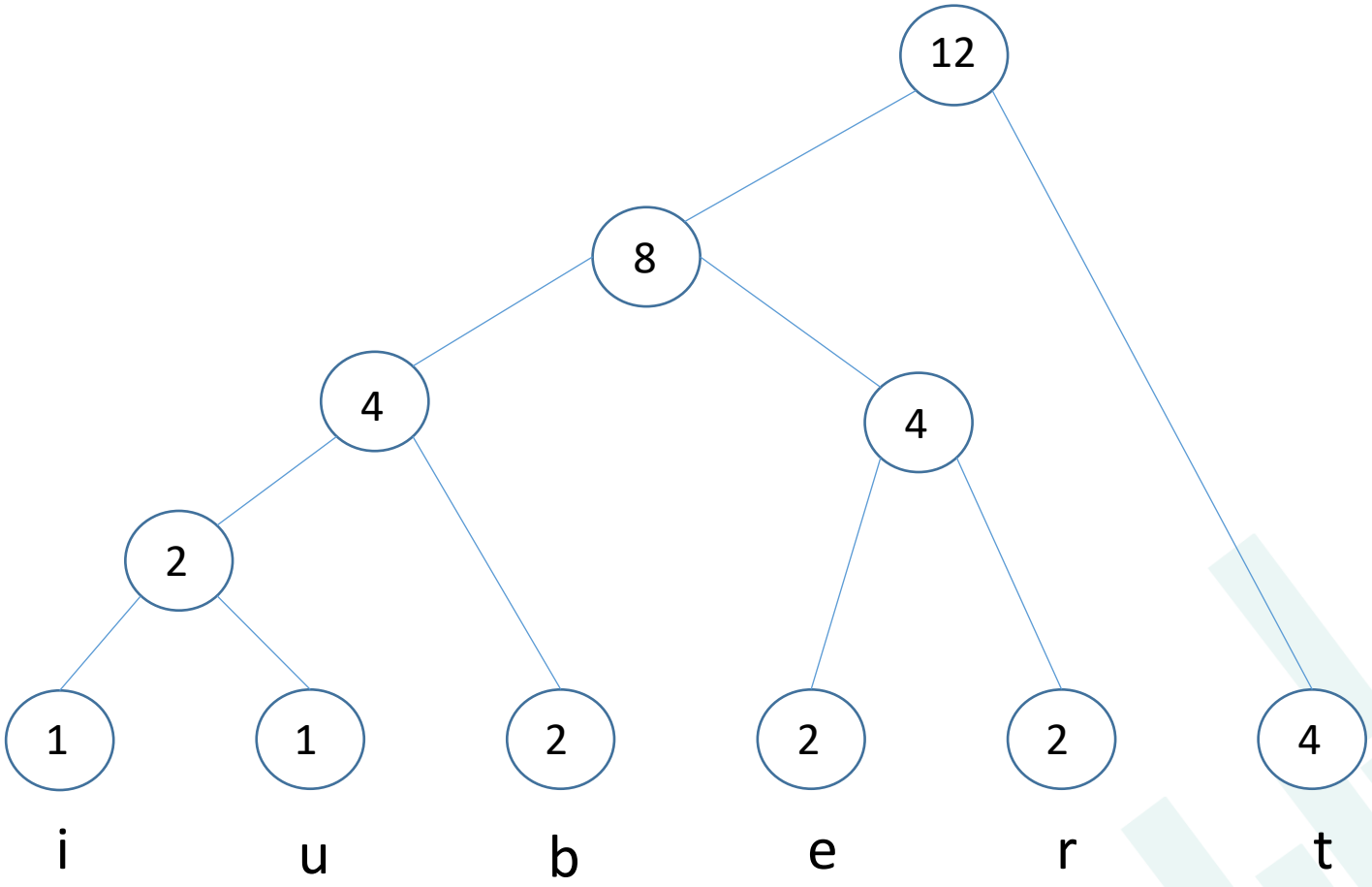
Code = 0010 0001 1010 0110 0100 0111 0100 11



Question 3

001000011010011001000111010011

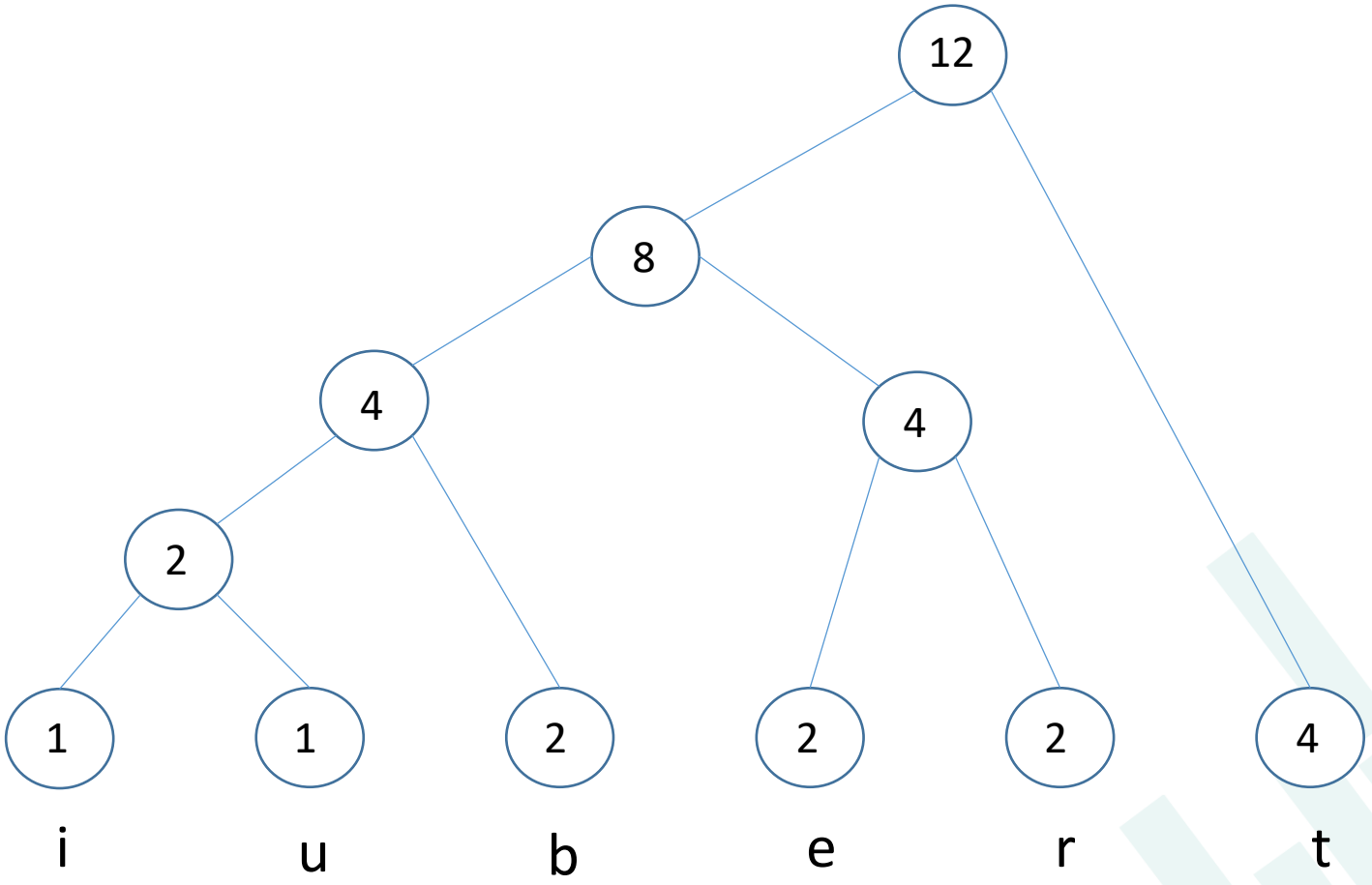
b



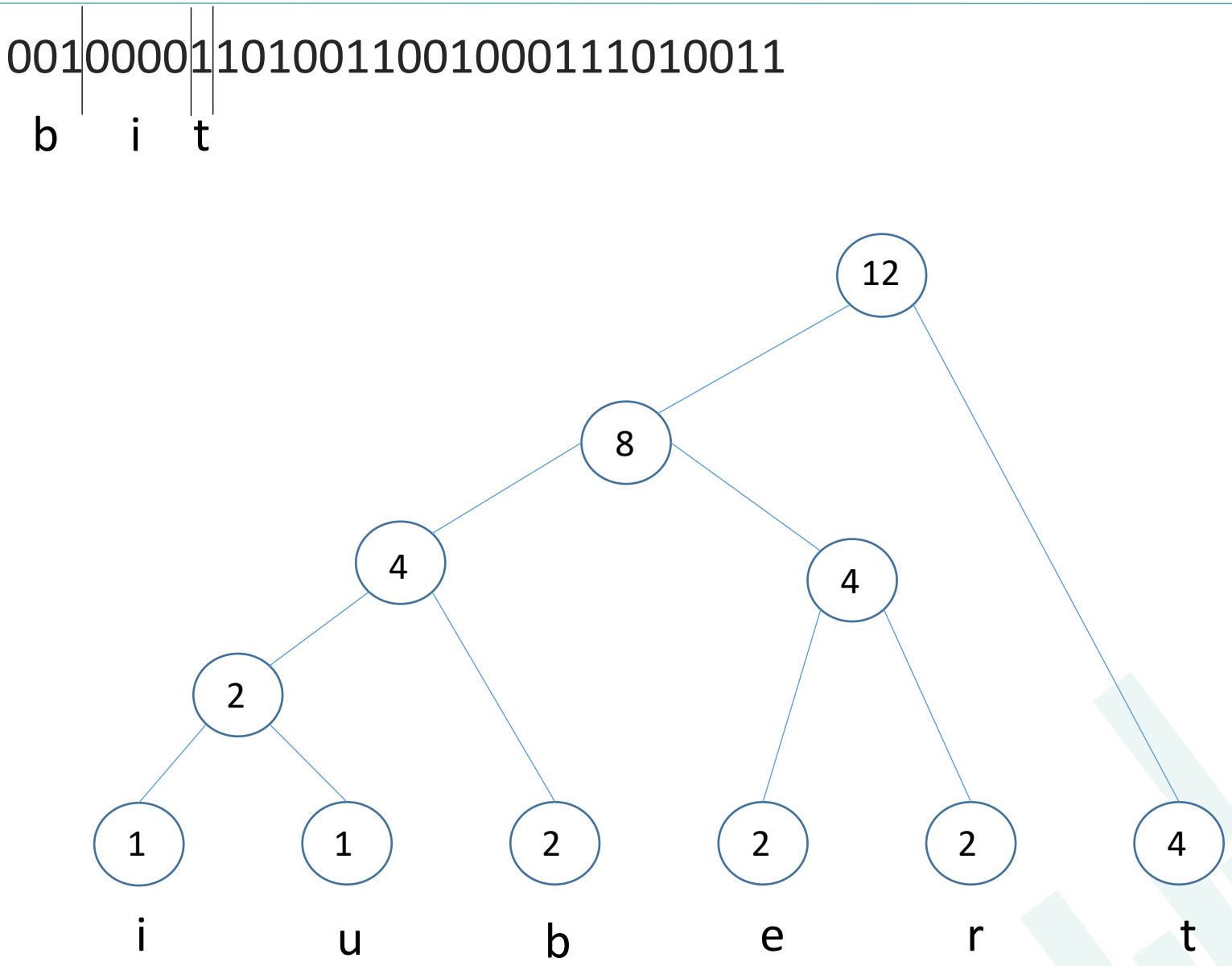
Question 3

001|0000|1101001100100011101001 1

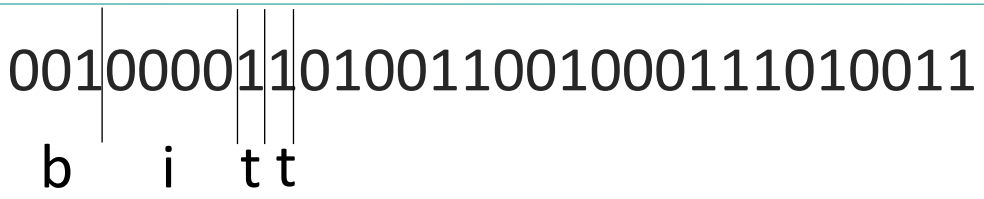
b i

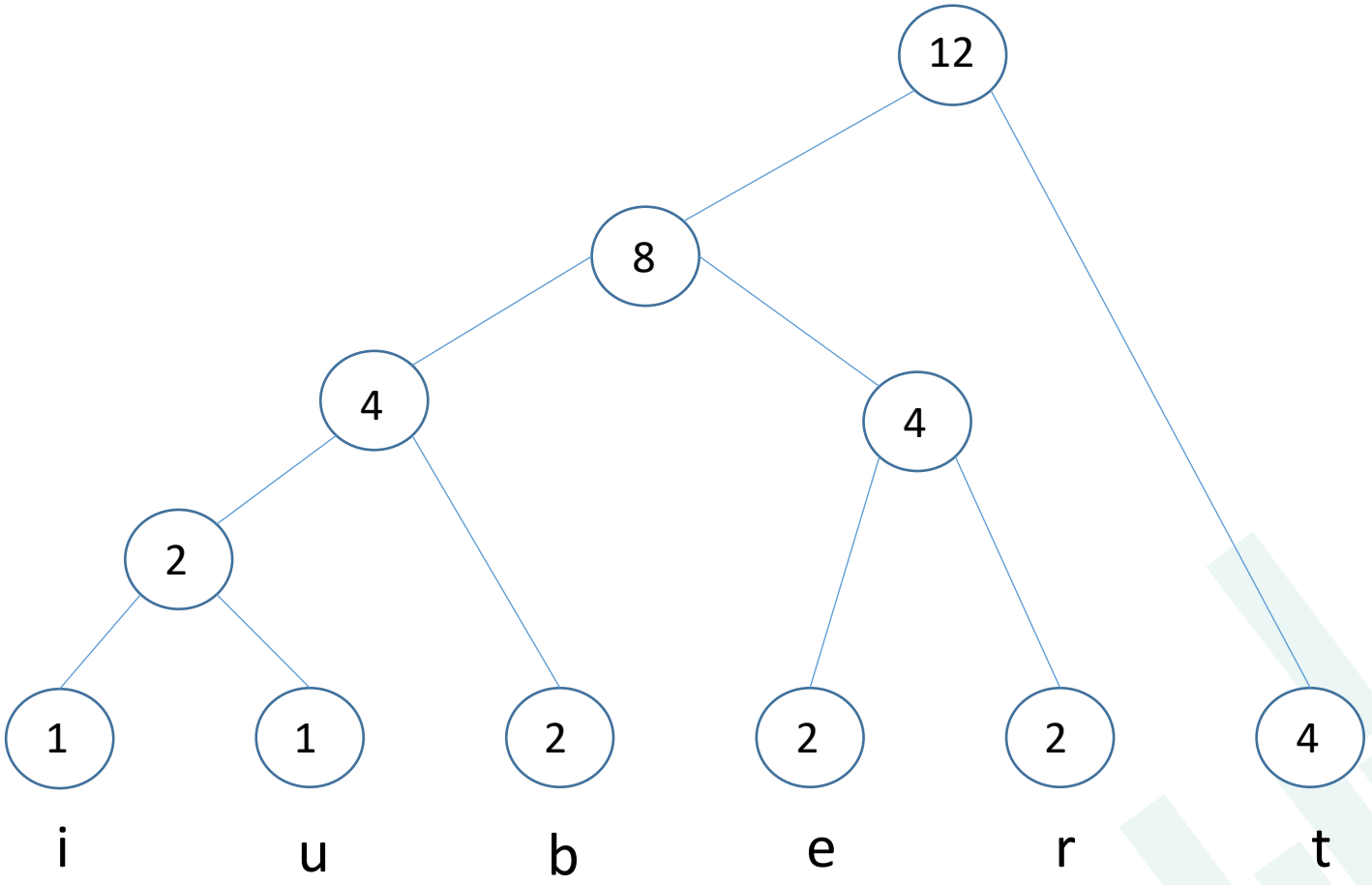


Question 3


00100001101001100100011101 0011
b
i
t
12
8
4
4
2
1
1
2
2
2
4
i
u
b
e
r
t

Question 3

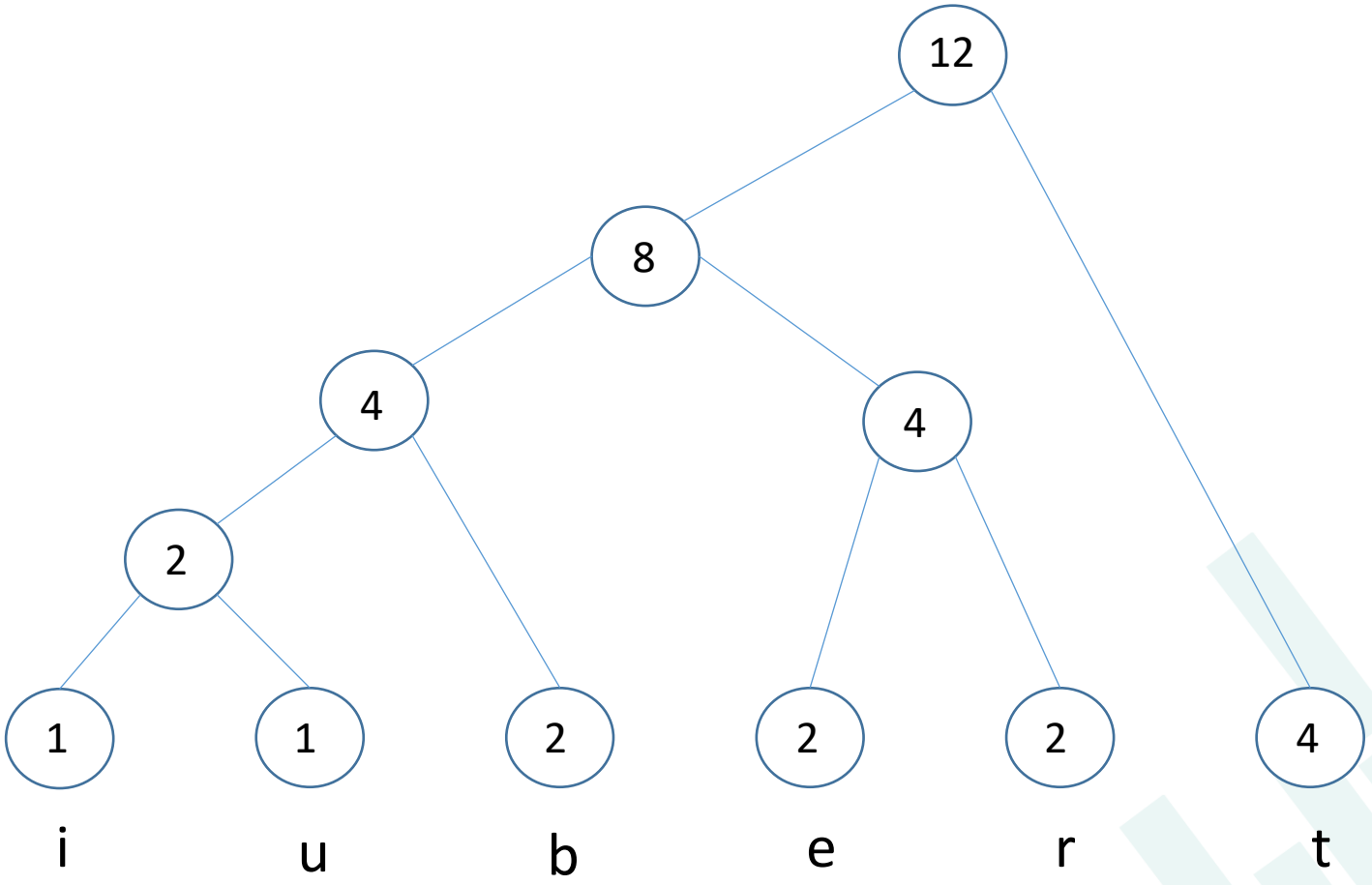

00100001101001100100011101001 1
b i t t


12
8
4
4
2
1
1
2
2
2
4
i
u
b
e
r
t

Question 3

00100001101001100100011101001 1

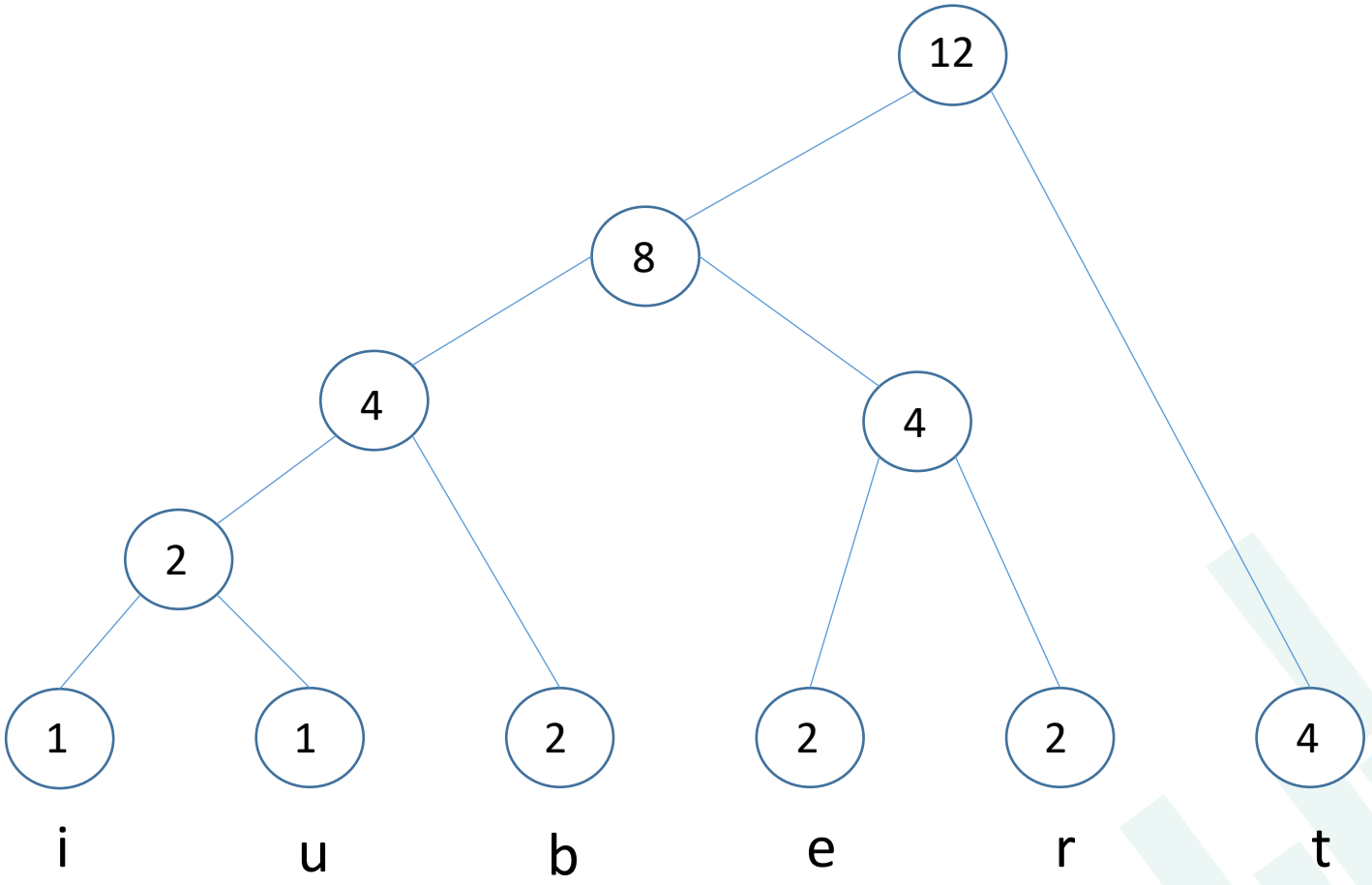
b i t t e



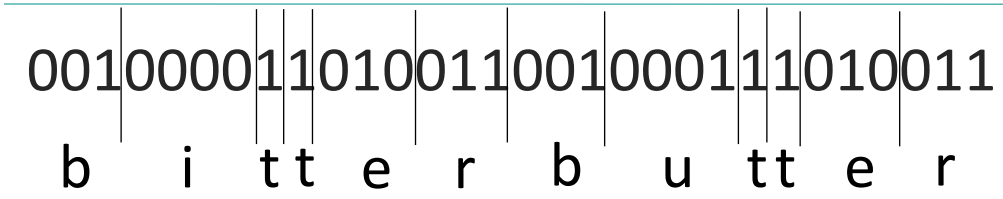
Question 3

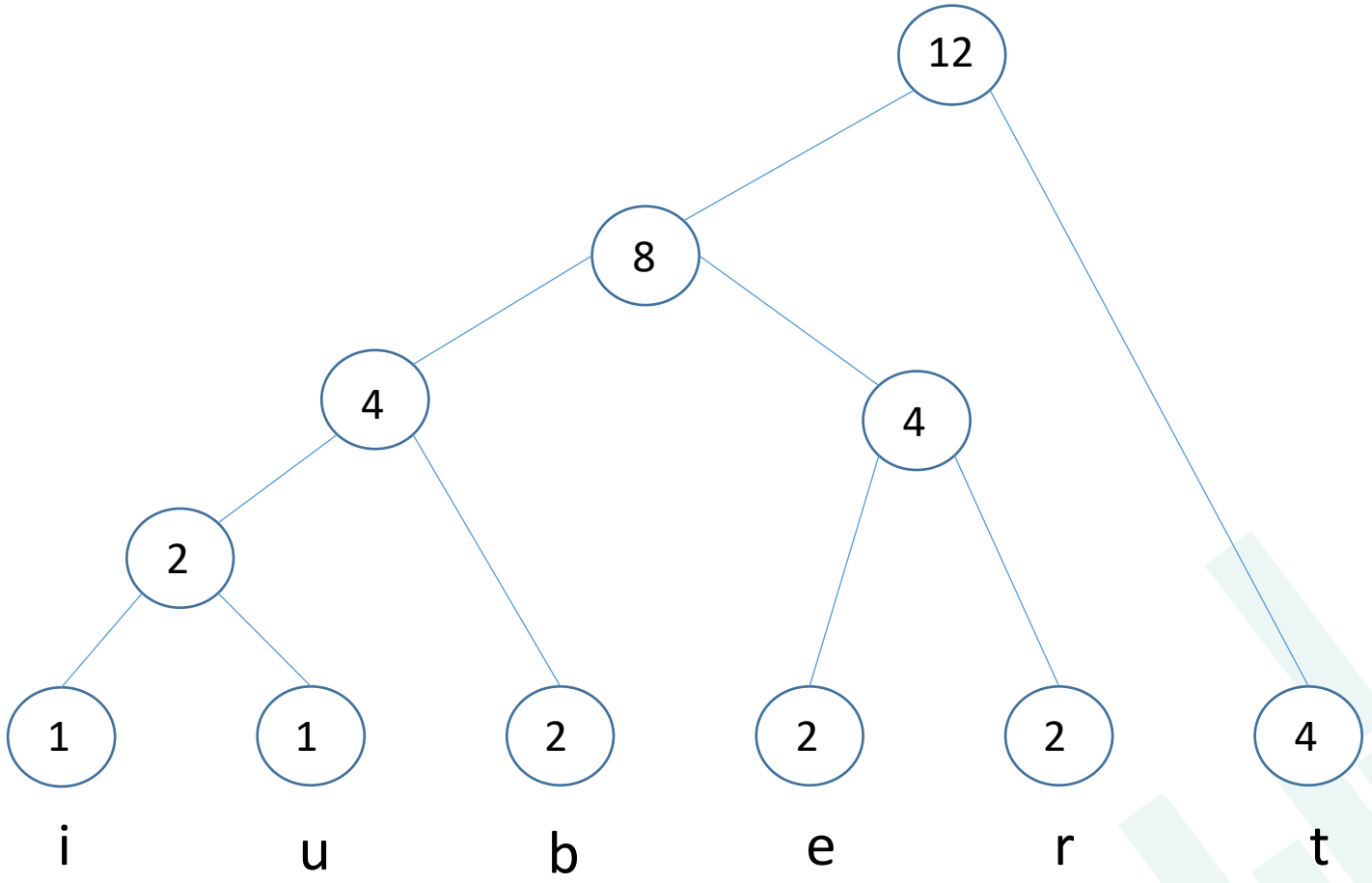
0010000111010011001000111010011

b i t t e .....



Question 3


00100001101001100100011101011
b i t t e r b u t t e r


12
8
4
4
2
1
1
2
2
2
4
i
u
b
e
r
t

To Ponder

1. Is the Huffman tree unique for any sequence of letters?

To Ponder

1. Is the Huffman tree unique for any sequence of letters?

2. In which case will the Huffman tree be unique?

To Ponder

1. Is the Huffman tree unique for any sequence of letters?

2. In which cases will the Huffman tree be unique?

Is the Huffman tree unique for a word with the following frequency table?

R	A	C	O	N	K	E	P
2	9	3	7	2	5	3	4

To Ponder

1. Is the Huffman tree unique for any sequence of letters?

2. In which case will the Huffman tree be unique?

3. Does the Huffman coding always enable compression? Or does the Huffman code of a string always have smaller size than binary encoding of that string?