UNIVERSITY INSTITUTE OF COMPUTING

MASTER OF COMPUTER APPLICATIONS

DESIGN AND ANALYSIS OF ALGORITHMS

24CAT-611

UNIT-2

DISCOVER . LEARN . EMPOWER

DESIGN AND ANALYSIS OF ALGORITHMS

Course Outcome

CO Number	Title	Level
CO3	Apply and analyze important algorithmic design paradigms and their applications	Understand
CO4	Implement the major graph algorithms to model engineering problems	Understand

- **Divide and Conquer**: General method, Binary search, Advantages and disadvantages of divide and conquer, Decrease and conquer approach: Topological sort

Topics to be covered

- Dijkstra’s Algorithm
- Huffman Trees and Codes
- Heaps and Heap Sort

Huffman Coding

- Huffman Coding is a famous Greedy Algorithm.
- It is used for the lossless compression of data.
- It uses variable length encoding.
- It assigns variable length code to all the characters.
- The code length of a character depends on how frequently it occurs in the given text.
- The character which occurs most frequently gets the smallest code.
- The character which occurs least frequently gets the largest code.
- It is also known as **Huffman Encoding**.

Prefix Rule

- Huffman Coding implements a rule known as a prefix rule.
- This is to prevent the ambiguities while decoding.
- It ensures that the code assigned to any character is not a prefix of the code assigned to any other character.

Huffman Coding

- Huffman Coding is a technique of compressing data to reduce its size without losing any of the details. It was first developed by David Huffman.
- Huffman Coding is generally useful to compress the data in which there are frequently occurring characters.
- **Huffman coding** assigns codes to characters such that the length of the code depends on the relative frequency or weight of the corresponding character. Huffman codes are of variable-length, and prefix-free (no code is prefix of any other). Any prefix-free binary code can be visualized as a binary tree with the encoded characters stored at the leaves.

Huffman coding tree

- **Huffman coding tree** or **Huffman tree** is a full binary tree in which each leaf of the tree corresponds to a letter in the given alphabet.
- Define the weighted path length of a leaf to be its weight times its depth. The Huffman tree is the binary tree with minimum external path weight, i.e., the one with the minimum sum of weighted path lengths for the given set of leaves. So the goal is to build a tree with the minimum external path weight.

How Huffman Coding works?

- Suppose the string below is to be sent over a network.

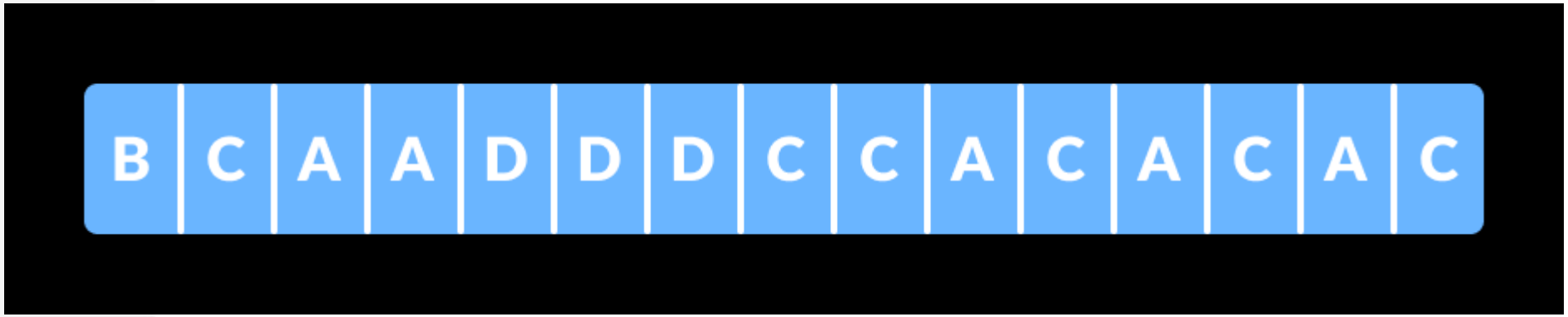


Fig 1: Initial string

How Huffman Coding works?

Each character occupies 8 bits. There are a total of 15 characters in the above string.

Thus, a total of 8 * 15 = 120 bits are required to send this string.

Using the Huffman Coding technique, we can compress the string to a smaller size.

Huffman coding first creates a tree using the frequencies of the character and then generates code for each character.

How Huffman Coding works?

Huffman coding is done with the help of the following steps.

1. Calculate the frequency of each character in the string.

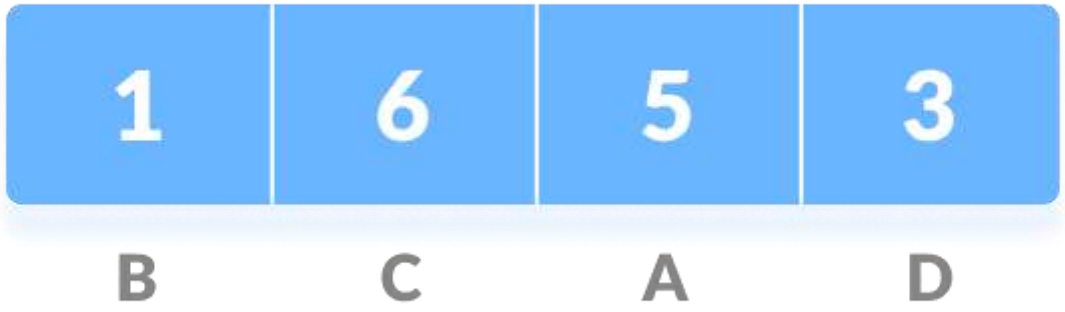


Fig 2: Frequency of string

How Huffman Coding works?

2. Sort the characters in increasing order of the frequency. These are stored in a priority queue Q.

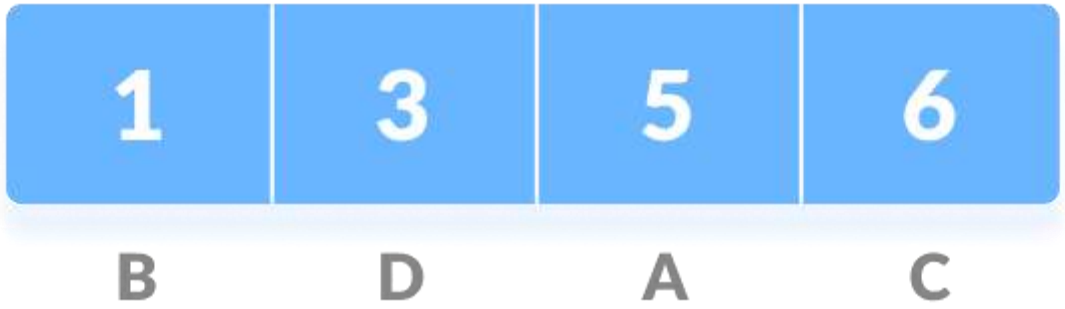


Fig 3: Characters sorted according to the frequency

How Huffman Coding works?

3. Make each unique character as a leaf node.

4. Create an empty node z. Assign the minimum frequency to the left child of z and assign the second minimum frequency to the right child of z. Set the value of the z as the sum of the above two minimum frequencies.

How Huffman Coding works?

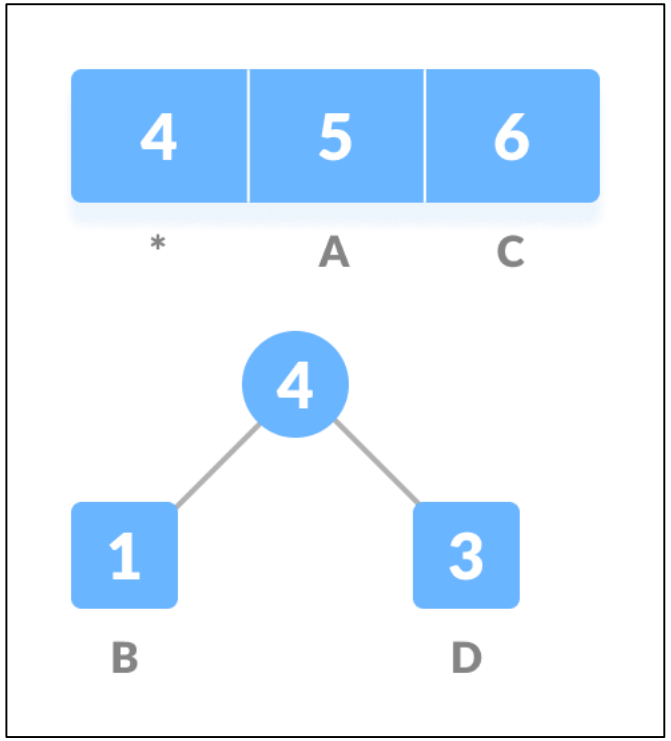


Fig 4: Getting the sum of the least numbers

Contd…

5. Remove these two minimum frequencies from Q and add the sum into the list of frequencies (* denote the internal nodes in the figure above).
6. Insert node z into the tree.
7. Repeat steps 3 to 5 for all the characters.

Contd…

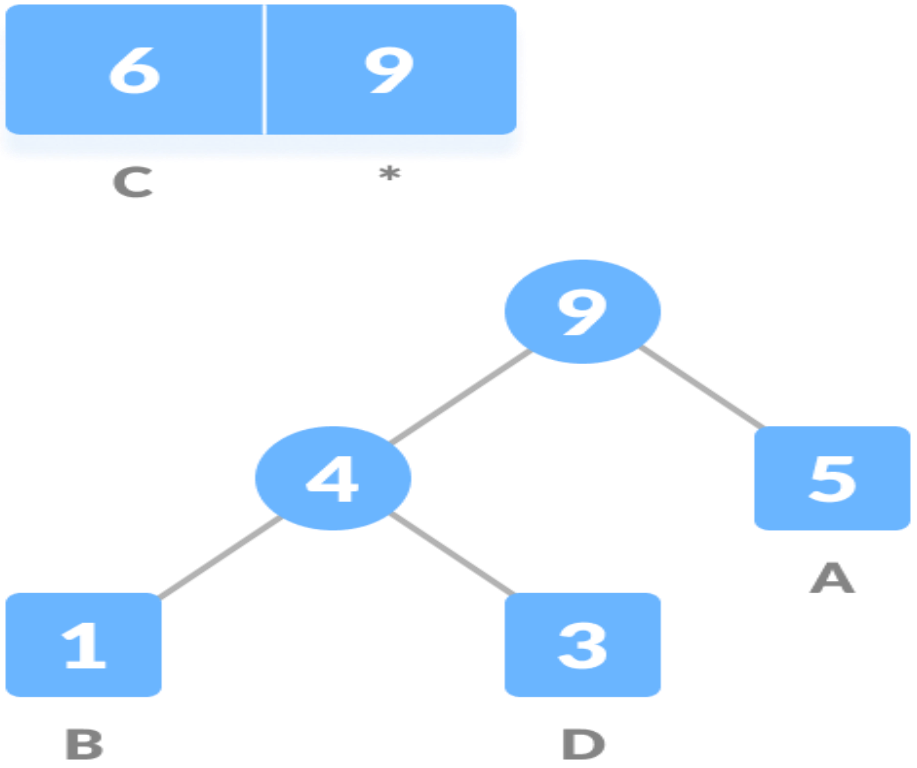


Fig 5: Repeat steps 3 to 5 for all the characters.

Contd…

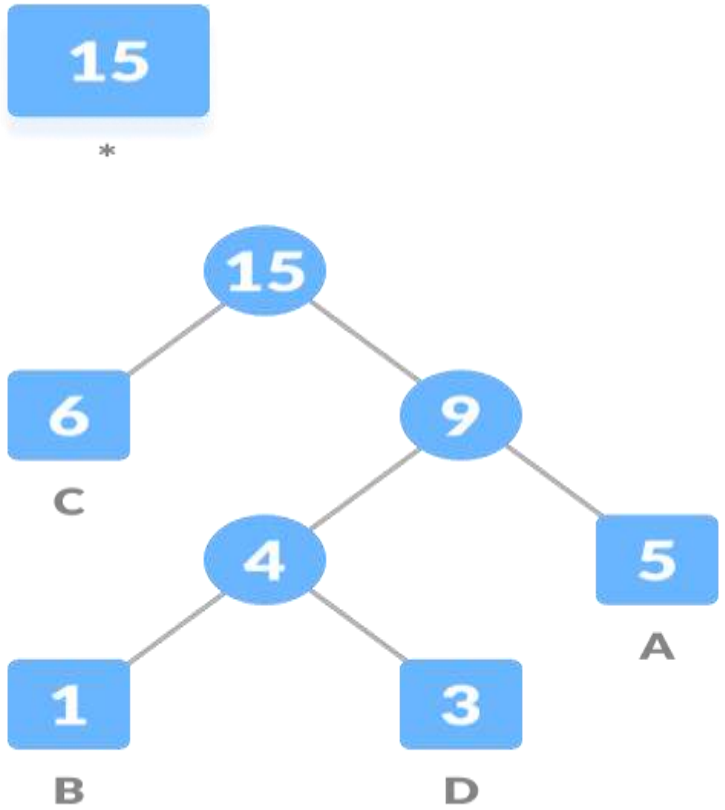


Fig 6: Repeat steps 3 to 5 for all the characters.

Contd…

8. For each non-leaf node, assign 0 to the left edge and 1 to the right edge.

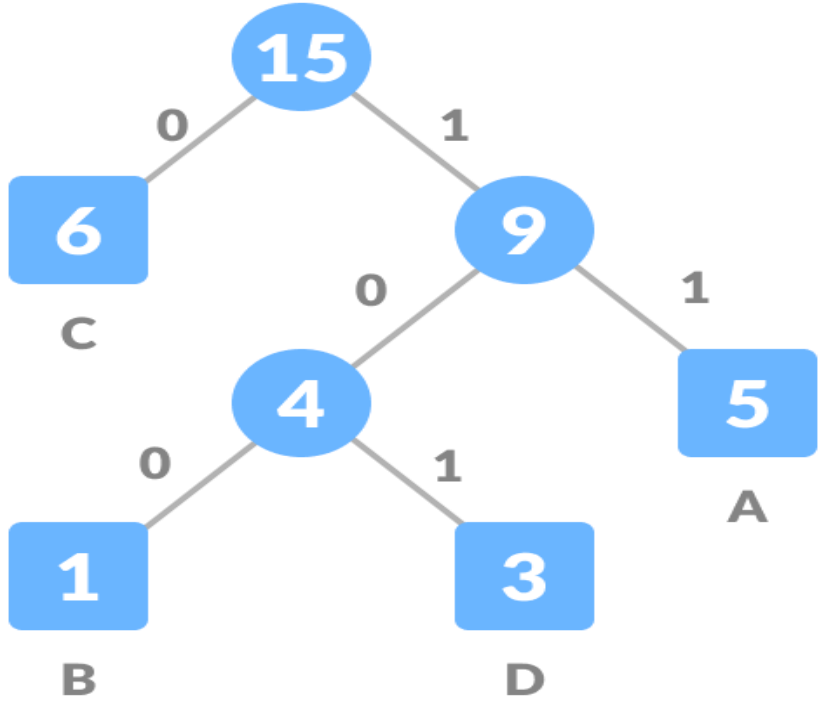


Fig 7: Assign 0 to the left edge and 1 to the right edge

Contd…

- For sending the above string over a network, we have to send the tree as well as the above compressed-code. The total size is given by the table below.

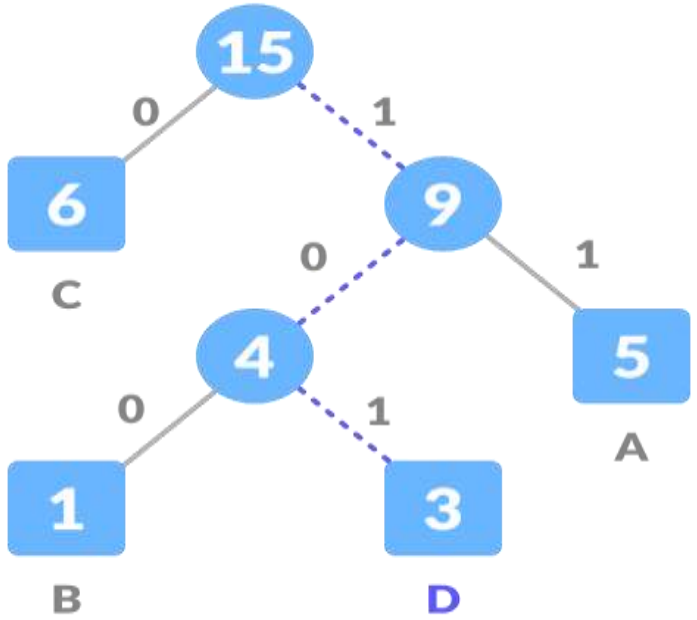
Character	Frequency	Code	Size
A	5	11	5*2 = 10
B	1	100	1*3 = 3
C	6	0	6*1 = 6
D	3	101	3*3 = 9
4 * 8 = 32 bits	15 bits		28 bits

Contd…

Without encoding, the total size of the string was 120 bits. After encoding the size is reduced to 32 + 15 + 28 = 75

Decoding the code

For decoding the code, we can take the code and traverse through the tree to find the character.
Let 101 is to be decoded, we can traverse from the root as in the figure below.



2. Example

Letter frequency table

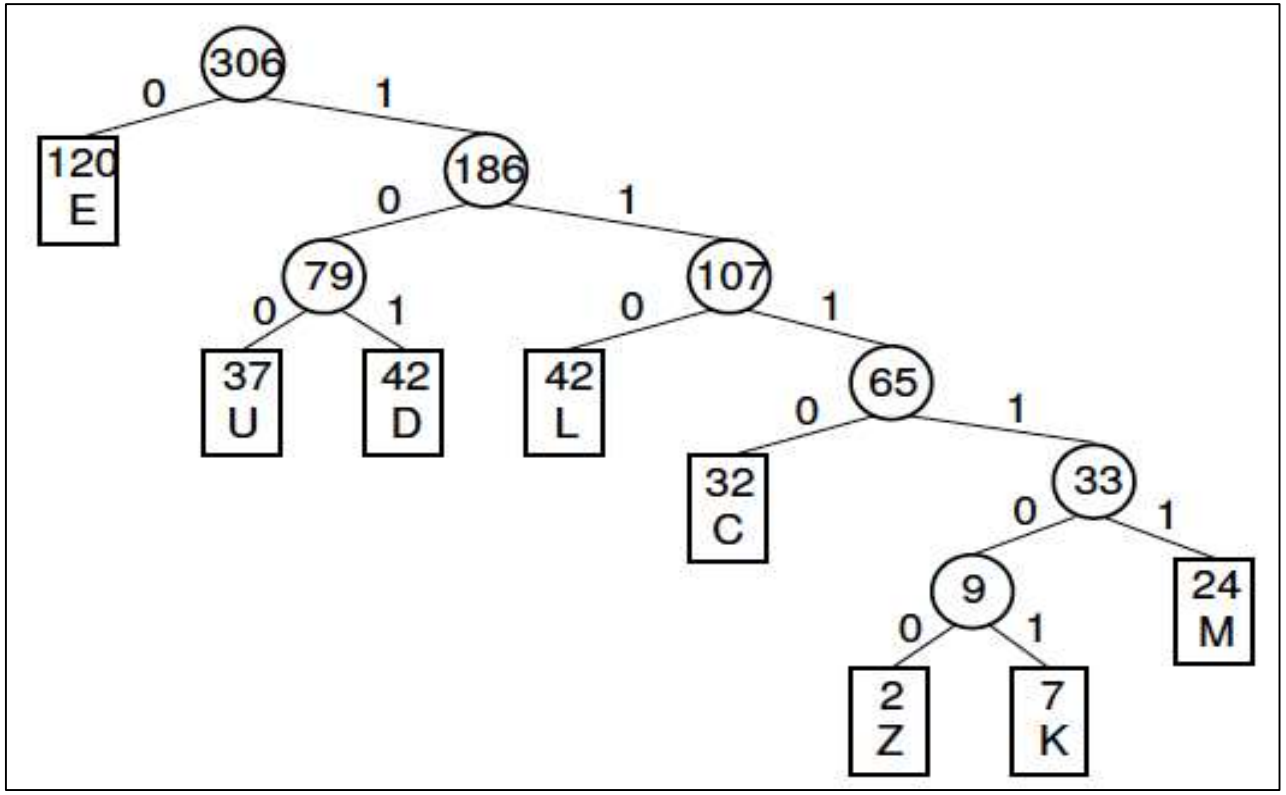
Letter	Z	K	M	C	U	D	L	E
Frequency	2	7	24	32	37	42	42	120

Contd..

Huffman code

Letter	Freq	Code	Bits
E	120	0	1
D	42	101	3
L	42	110	3
U	37	100	3
C	32	1110	4
M	24	11111	5
K	7	111101	6
Z	2	111100	6

The Huffman tree



Contd..

- Three problems:
- Problem 1: Huffman tree building
- Problem 2: Encoding
- Problem 3: Decoding
- **Problem 2: Encoding**
- Encoding a string can be done by replacing each letter in the string with its binary code (the Huffman code).
- Examples:
  DEED 10100101 (8 bits)
  MUCK 1111110011110111101 (18 bits)
  Problem 3: Decoding
- Decoding an encoded string can be done by looking at the bits in the coded string from left to right until a letter decoded.
  10100101 -> DEED

References

[1] https://www.programiz.com/dsa/huffman-coding

[2] http://homes.sice.indiana.edu/yye/lab/teaching/spring2014-C343/huffman.php#:~:text=Huffman%20coding%20tree%20or%20Huffman,its%20weight%20times%20its%20depth.

Books:

1. Introduction to Algorithms by Coreman, Leiserson, Rivest, Stein.
2. Fundamentals of Algorithms by Ellis Horwitz, Sartaj Sahni, Sanguthevar Rajasekaran

THANK YOU

For query:
uttamae7426@cumail.in