DATE : / /

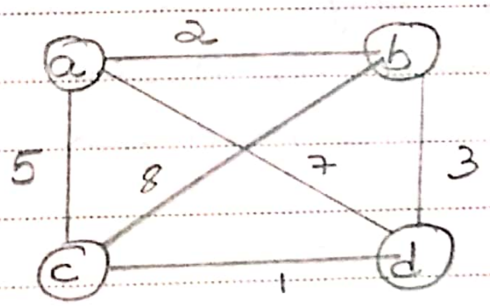
* Traveling Salesman Problem *

Problem: The problem asks to find the shortest tour through a given set of n cities that visits each city exactly once before returning to the city where it started.

Its the problem of finding the shortest Hamiltonian Circuit of the graph.

Technique: Exhaustive Search (Brute Force)

Example:



Tour	Length	
a → b → c → d → a	2 + 8 + 1 + 7 = 18	
a → b → d → c → a	2 + 3 + 1 + 5 = 11	optimal
a → c → b → d → a	5 + 8 + 3 + 7 = 23	
a → c → d → b → a	5 + 1 + 3 + 2 = 11	optimal
a → d → b → c → a	7 + 3 + 8 + 5 = 23	
a → d → c → b → a	7 + 1 + 8 + 2 = 18	

Observations :

- Pairs of tours might only differ by direction

- Approach is practical only for smaller values of n
  ~~Total permutations needed will be n!~~

- We can get all the tours by generating all the permutations of $n-1$ intermediate cities.
  We could cut the number of vertex permutations by half. (Eg: choose only permutations where v_1 precedes v_2).
  This will reduce the number of permutations to $\frac{(n-1)!}{2}$.

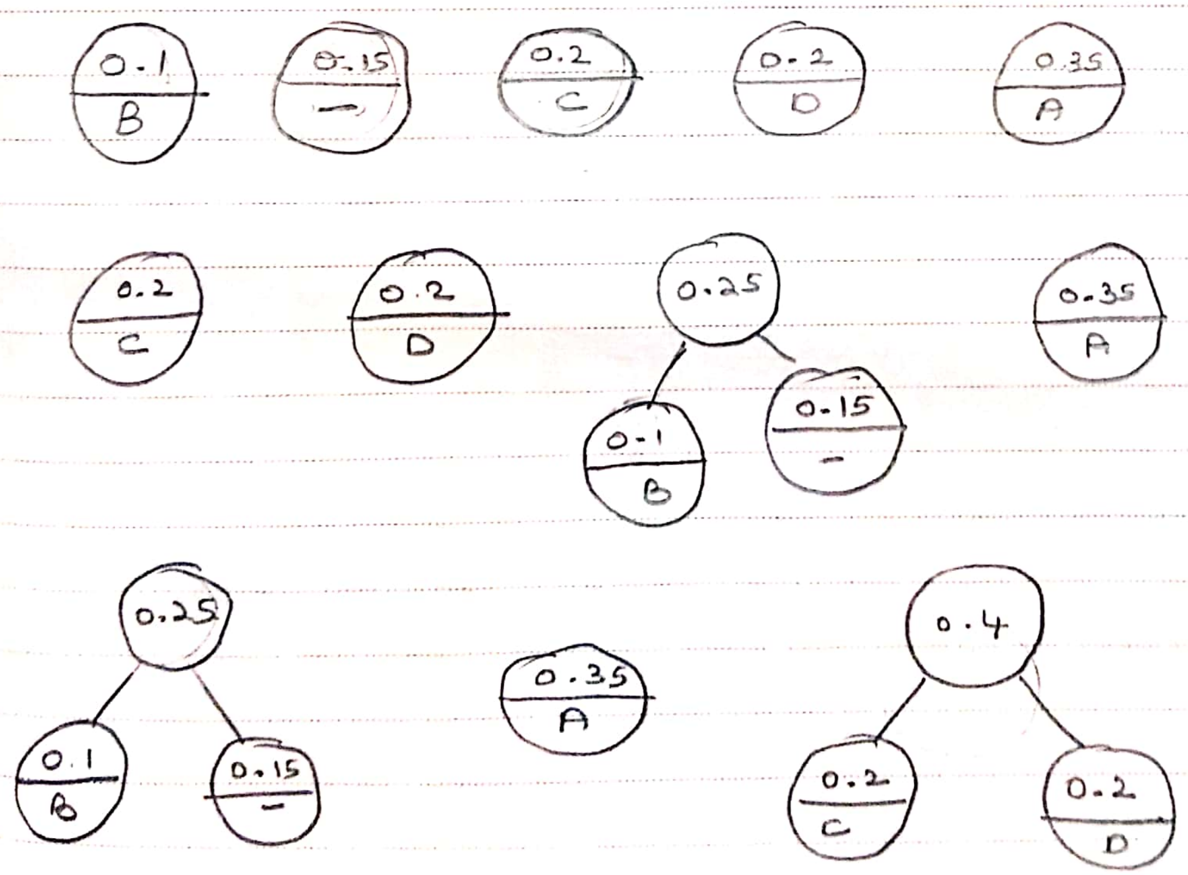
* Huffman Trees *

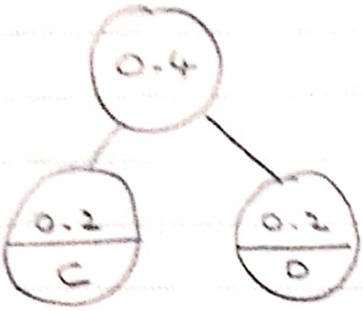
<u>Motivation</u>: Encode a text that comprises characters from some n-character alphabet by assigning to each of the text's characters some sequence of bits called the codeword.

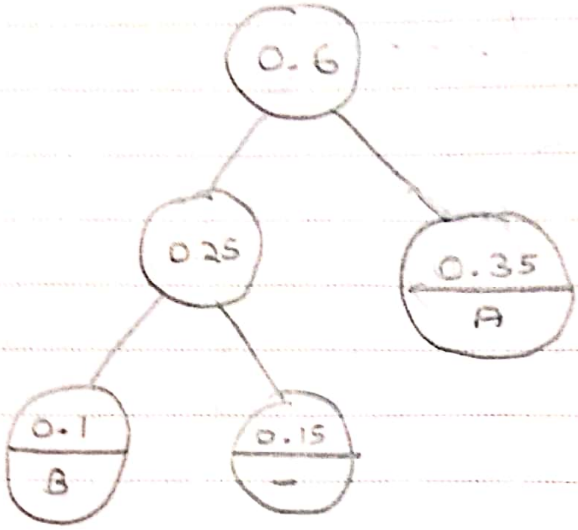
<u>Example</u>: Consider five-character alphabet with following occurrence probabilities:

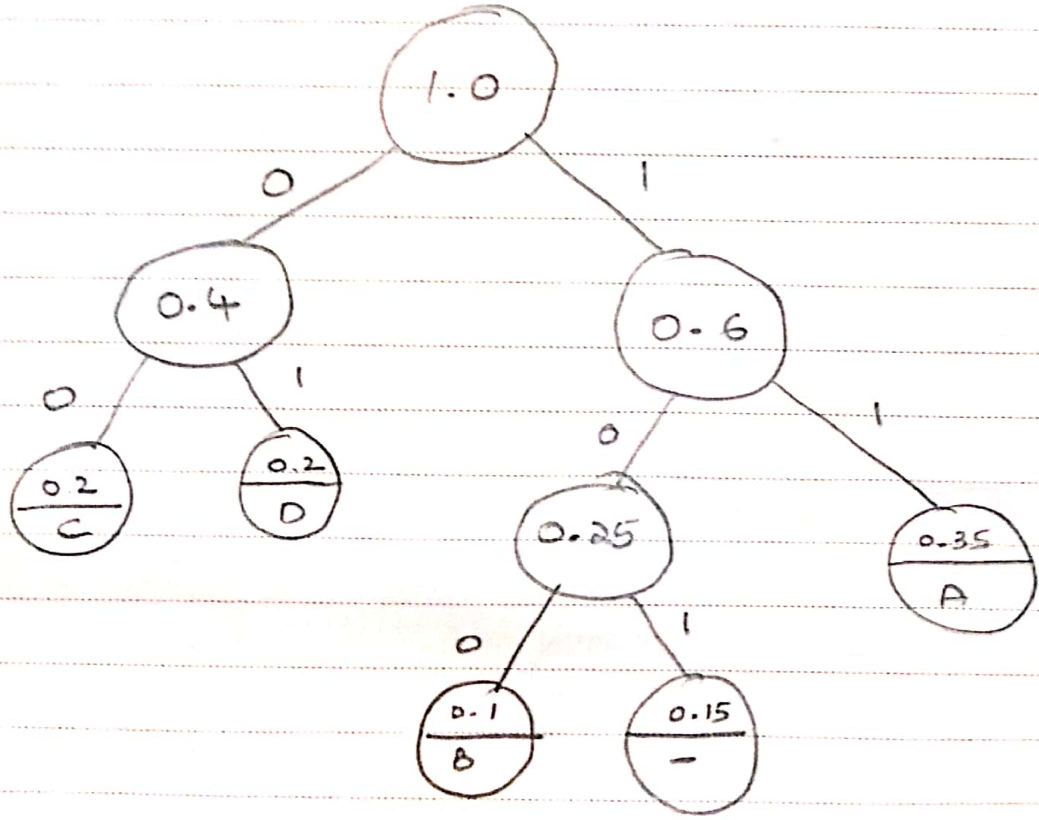
character	A	B	C	D	_
probability	0.35	0.1	0.2	0.2	0.15

We construct Huffman Coding Tree as:









character	A	B	C	D	–
probability	0.35	0.1	0.2	0.2	0.15
Codeword	11	100	00	01	101

PAGE NO. :
DATE : / /

Notes :

- Constructs a tree that assigns shorter bit strings to high-frequency characters & longer ones to low-frequency characters.

- prefix free or prefix codes - no codeword is a prefix of a character codeword of another character.

- Fixed length v/s variable length encoding

- For the example, the expected number of bits per character in this code is,

$$= 2 * 0.35 + 3 * 0.1 + 2 * 0.2 + 2 * 0.2 + 3 * 0.15$$

$$= 2.25.$$

- Compression ratio :

$$= \frac{3 - 2.25}{3} * 100 = 25\%.$$

↳ Fixed length would have used 3.

Huffman uses 25% less memory than its fixed length encoding.

- Experiments show that Huffman codes have compression ratio typically falling between 20% & 80%.

Exercise:

1. Construct a Huffman tree for the following data & obtain its Huffman code:

character	A	B	C	D	E	_
probability	0.5	0.35	0.5	0.1	0.4	0.2

Encode the text DAD_BE using the obtained code

Decode the text whose encoding is 1100110110

What is the achieved compression ratio?