* Chapter 07 *

-Prakash Hegade

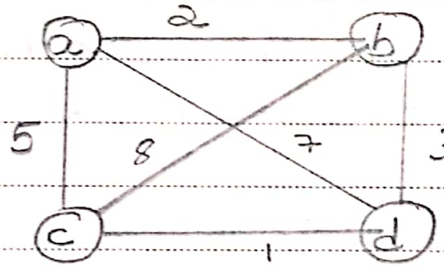
* Traveling Salesman Problem *

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

Its the problem of finding the shortest <u>Hamiltonian Circuit</u> of the graph.

<u>Technique</u>: Exhaustive Search (Brute Force)

<u>Example</u>:



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 $\frac{(n-1)!}{2}$.

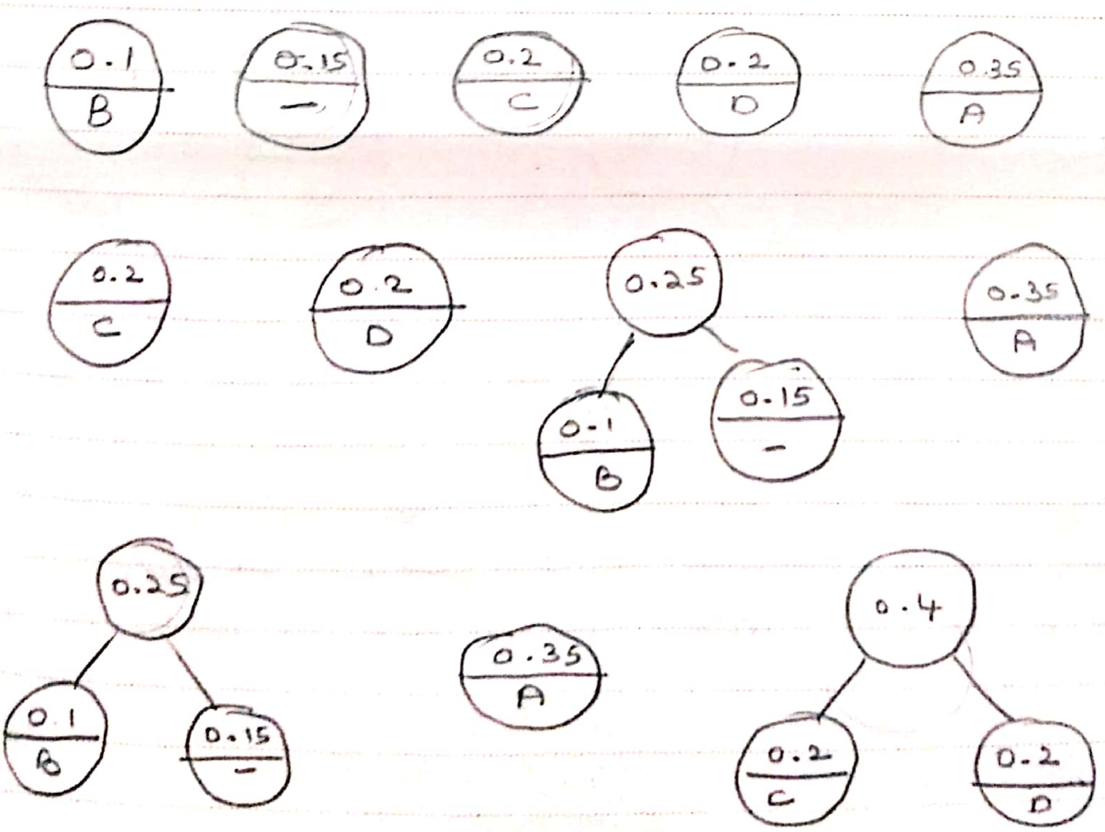
Huffman Trees

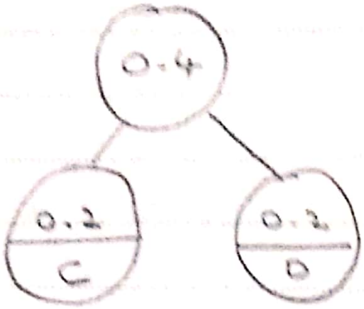
Motivation: Encode a text that comprises characters from some n-character alphabet by assigning to each of the text's characters some sequence of bits called the codeword.

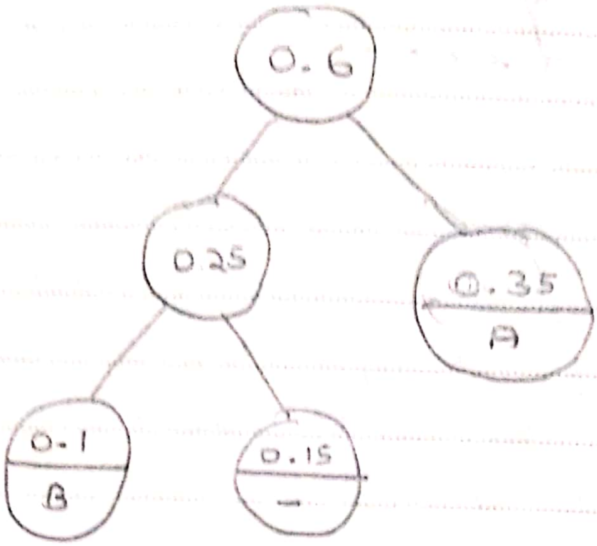
Example: Consider five-character alphabet with following occurrence probabilities:

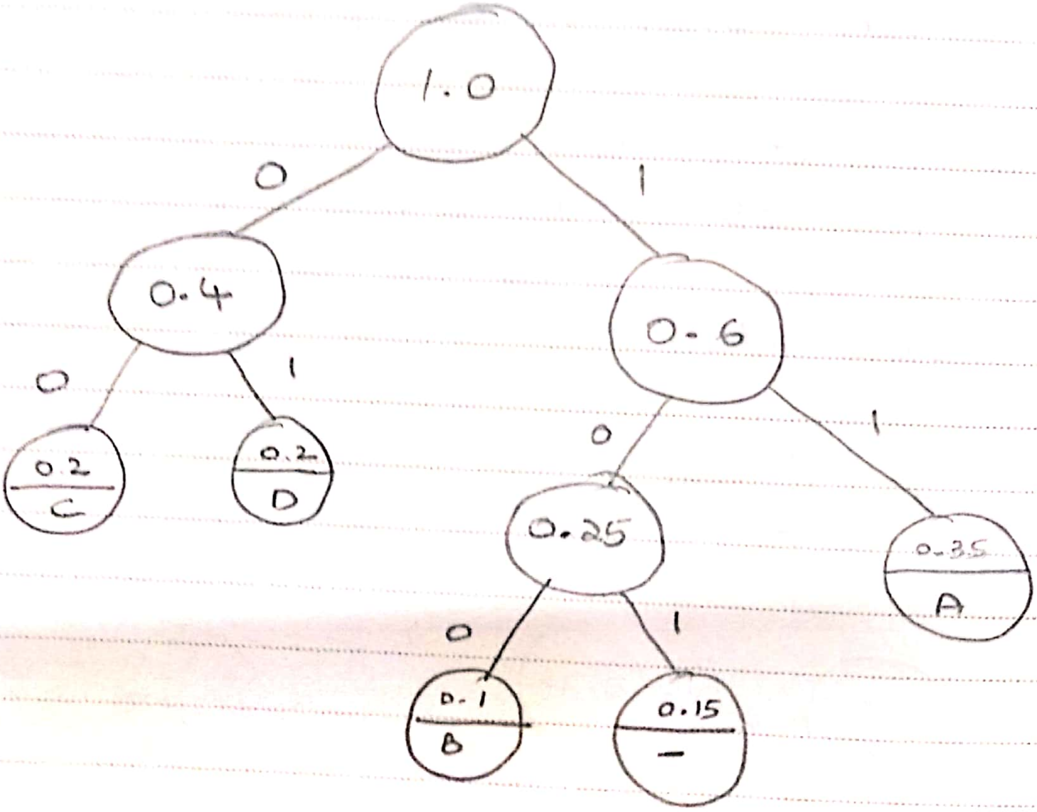
character	A	B	C	D	_
probability	0.35	0.1	0.2	0.2	0.15

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?

Technique: Greedy algorithm.

* The knapsack Problem *

Given items, weights & values, find the maximum value of a subset of the n given items that fit into the knapsack of capacity W, and an optimal subset itself.

Table:

V

	0	$j-w_i$	j	W
0	0	0	0	0
w_i, v_i $i-1$	0	$V[i-1][j-w_i]$	$V[i-1][j]$	
i	0		$V[i,j]$	
n	0			goal

Recurrence Relation:

$V[0,j] = 0$ for $j \geq 0$ & $V[i,0] = 0$ for $i \geq 0$

$$V[i,j] = \begin{cases} \max\{V[i-1,j],\ v_i + V[i-1, j-w_i]\} & \text{if } j - w_i \geq 0 \\ V[i-1,j] & \text{if } j - w_i < 0 \end{cases}$$

Technique: Dynamic Programming.

Example:

$W = 5$

item	weight	value
1	2	12
2	1	10
3	3	20
4	2	15

	i	0	1	2	3	4	5
	0	0	0	0	0	0	0
$w_1 = 2, v_1 = 12$	1	0	0	12	12	12	12
$w_2 = 1, v_2 = 10$	2	0	10	12	22	22	22
$w_3 = 3, v_3 = 20$	3	0	10	12	22	30	32
$w_4 = 2, v_4 = 15$	4	0	10	15	25	30	$\boxed{37}$

To find the items included:

i	0	1	2	3	4	5
0	$\boxed{0}$	0	0	0	0	0
1	0	0	$\boxed{12}$	12	12	12
2	0	10	12	$\boxed{22}$	22	22
3	0	10	12	22	30	32
4	0	10	15	25	30	$\boxed{37}$

Items included are: 1, 2 and 4.

PAGE NO. :
DATE : / /

Exercise :

$W = 10$

item	weight	value
1	7	42
2	3	12
3	4	40
4	5	25

* Fake Coin Problem *

Identify the fake coin among n-identically looking coins.

<u>Technique</u>: Decrease and Conquer (decrease by 1)

<u>Idea</u>: Divide n coins into ~~n/2~~ two piles of n/2 each, leaving one extra coin apart if n is odd, and put the two piles on the scale.

If the piles weigh the same, the coin put aside may be fake. (Assume fake coin weighs less) Otherwise, we can proceed ahead in the same manner with the lighter pile, which must be the one with fake coin.

Recurrence relation for the number of weighings W(n) needed by this algorithm in the worst case:

$$W(n) = \begin{cases} 0 & n = 1 \\ W(n/2) + 1 & \text{otherwise} \end{cases}$$

PAGE NO. :
DATE : / /

Problem is similar to worst case of binary search.

$W(n) = \log_2 n$.

Assignment

Discuss the problem if it is solved by dividing the coins into three piles of about $n/3$ each.

What will be the effect on efficiency?

Note:

$W(n) = W(n/2) + 1 \rightarrow (1)$

is solved likewise the ones solved in previous classes.

* Strassen's Matrix Multiplication *

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$

$$= \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

$$m_1 = (a_{00} + a_{11}) * (b_{00} + b_{11})$$
$$m_2 = (a_{10} + a_{11}) * b_{00}$$
$$m_3 = a_{00} * (b_{01} - b_{11})$$
$$m_4 = a_{11} * (b_{10} - b_{00})$$
$$m_5 = (a_{00} + a_{01}) * b_{11}$$
$$m_6 = (a_{10} - a_{00}) * (b_{00} + b_{01})$$
$$m_7 = (a_{01} - a_{11}) * (b_{10} + b_{11})$$

For a $2*2$ matrices multiplication:

Brute Force: 8 multiplications & 4 additions
Strassen's : 7 multiplications & 18 add/sub

Recurrence relation for number of Multiplications:

$$M(n) = \begin{cases} 7M(n/2) & \text{for } n > 1 \\ 1 & n = 1 \end{cases}$$

PAGE NO.
DATE / /

$M(n) = 7M(n/2)$

assume $n = 2^k$

$M(2^k) = 7M(2^{k-1}) \rightarrow (1)$

put $k = k-1$ in (1)

$M(2^{k-1}) = 7M(2^{k-2}) \rightarrow (2)$

put (2) in (1)

$$M(2^k) = 7^2 M(2^{k-2})$$
$$= 7^3 M(2^{k-3})$$
$$= \dots$$
$$= 7^k M(2^{k-k})$$
$$= 7^k$$

$$\begin{cases} n = 2^k \\ \log_2 n = k \end{cases}$$

$$= 7^{\log_2 n}$$
$$= n^{\log_2 7}$$
$$= n^{2.807}$$

This is smaller than n^3 as required by Brute Force.

If we consider additions,

$$A(n) = \begin{cases} 7A(n/2) + 18(n/2)^2 & n > 1 \\ 0 & n = 1 \end{cases}$$

Example:

$$A = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix} \qquad B = \begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix}$$

$m_1 = 6 * 8 = 48$

$m_2 = 12 * 6 = 72$

$m_3 = 1 * 6 = 6$

$m_4 = 5 * -2 = -10$

$m_5 = 4 * 2 = 8$

$m_6 = 6 * 14 = \cancel{60}\ 84$

$m_7 = -2 * 6 = -12$

$$C = \begin{bmatrix} 48-10-8-12 & 6+8 \\ 72-10 & 48+6-72+\cancel{60}\,84 \end{bmatrix}$$

$$= \begin{bmatrix} 18 & 14 \\ 62 & 66 \end{bmatrix}$$

Note:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$