Unit - IV.

Huffman Coding. (Introduced by Huffman - 1952).

- A measure to reduce / removing coding redundancy.
- Most popular coding redundancy technique.
- Variable length code.

Example

Image size 10 x 10.

Frequency

a1	- 10
a2	- 40
a3	- 6
a4	- 10
a5	- 4.
a6	- 30

Arrange / ordering.

a2 = 40
a6 - 30
a1 - 10
a4 - 10
a3 - 6
a5 - 4.

Probability of a2 $40/100 = 0.4$

$\{a1, a2, a3, a4, a5, a6\} = \{0.1, 0.4, 0.06, 0.1, 0.04, 0.3\}$

Huffman Source Reduction.

Original Source		Source Reduction.			
Symbol	Probability	1	2	3	4.
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_5	0.06	0.1			
a_6	0.04				

Step 1:

- Create a Series of Source reductions by ~~ordering~~ ordering the probabilities of Symbol.
- CSymbol - intensities of image)
- First Source reduction is formed by combining 0.06 and 0.04

  0.06+0.04 = 0.1.
- In the Source reduction, I column, write this 0.1 instead of 0.06 & 6.04.
- Second reduction (0.1 + 0.1 = 0.2) - 0.2 is written as a 3rd number in II column
- In column IV we get only 2 values so we can Stop the reduction.

Step 2 : Huffman Code Assignment Procedure.

- In this Step, each reduced Source is coded.
- It starts from the Smallest Source obtained in the last Step and goes back to the original Source.
- The minimal lengths binary codes are used are 0 and 1.

Huffman Code Assignment Procedure:

Original Source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	010	0.1	010	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

- The reduced symbols 0.6 and 0.4 in the last column are assigned 0 and 1.
- Since 0.6 was generated by combining (0.3 + 0.3).
So symbols in the reduced source to its part 0 and 1 are appended which produces codes 00 and 01.
- 0.3 was generated by 0.2 + 0.1 -
Codes - 010 and 011 codes produced.
- This operation is repeated for each produced source until the original source is reached.
- The average length of this code is

$$L_{avg} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + 0.04(5)$$

$$= 2.2 \text{ bits/pixels}$$

Advantages:
- It creates an optimal code for a set of symbols and probability.
- Implementation is very simple.

Drawbacks:
- It is difficult to implement when a large no. of symbols has to be coded.