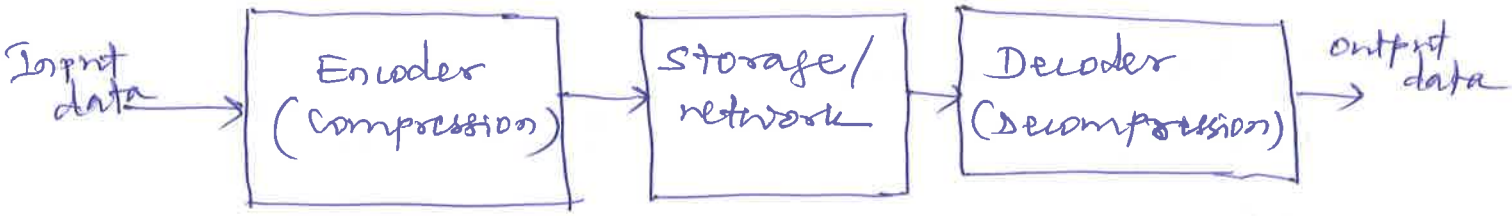
Unit 2 (Part 2)

Lossless Compression Algorithms

- Lossless – compression & decompression introduces no loss in information.
- Lossy – compression/decompression introduces loss in information that is tolerable.
- Information for compression/decompression – - image, video, audio

General data compression scheme



$$\text{Compression ratio} = \frac{\text{Total no. of bits reqd. to represent data before compression}}{\text{Total no. of bits reqd. to rep. data after compression}}$$

$$= \frac{B_0}{B_1} \geq 1$$

BASICS OF INFORMATION THEORY

Shannon entropy theory for a source of information

$$S = \{s_1, s_2, \ldots s_n\},$$

$$\eta = H(S) = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{n} p_i \log_2 p_i$$

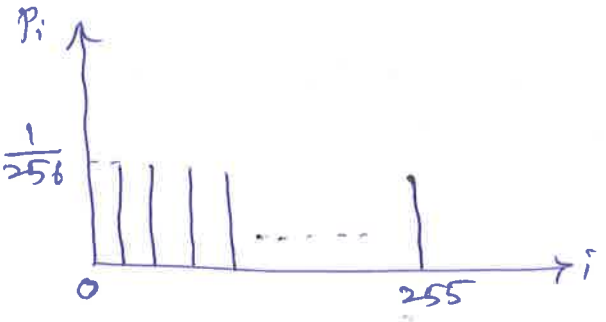
p_i = the probability that the symbol s_i in S is present.

$\log_2 \frac{1}{p_i}$ = amount of infmn. = no. of bits required to encode s_i

if probability of $s_i = 1/32$ in information $p_i = 1/32$, then amount of information is reptd as 5 bits since $\log_2 \frac{1}{p_i} = \log_2 32 = 5$ bits.

Entropy of 2 gray level images

Image 1 histogram



Gray level pixel values

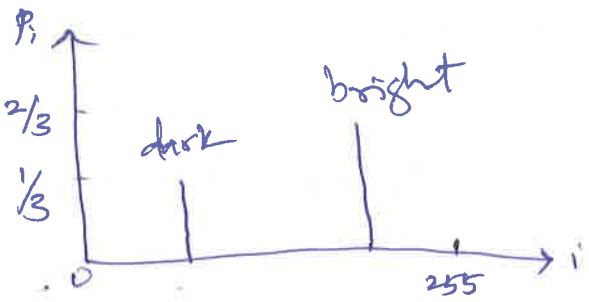
$\forall i;\ p_i = \frac{1}{256}$

$$\eta = \sum_{i=1}^{256} p_i \log_2 \frac{1}{p_i}$$

$$= \sum_{i=1}^{256} \frac{1}{256} \log_2 256 = 8$$

ie min. no. of bits reqd. to represent each gray intensity is 8.
No compression possible!

Image 2 histogram



$p_1 = 1/3 \qquad p_2 = 2/3$

$$\eta = \sum_{i=1}^{2} p_i \log_2 \frac{p_i}{p_i} =$$

$$= \frac{1}{3} \log_2 3 + \frac{2}{3} \log_2 3/2$$

$$= 0.92.$$

Each pixel is represented by less than 1 bit.

Run length coding (RLC)

- If information content contains of each symbol groups have same values, it is possible to code the groups rather than individuals. It achieves better compression.

Variable Length Coding (VLC)

Shannon Fano & Huffman coding

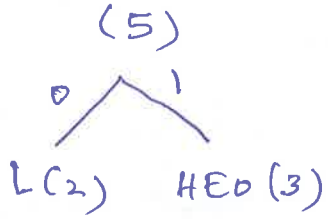
Shannon Fano Coding: word HELLO
(Top down approach)

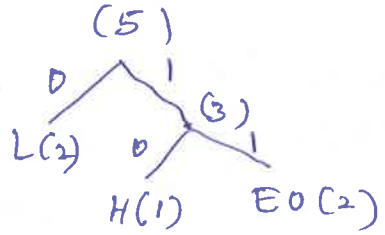
Symbol	H	E	L	O
Count	1	1	2	1

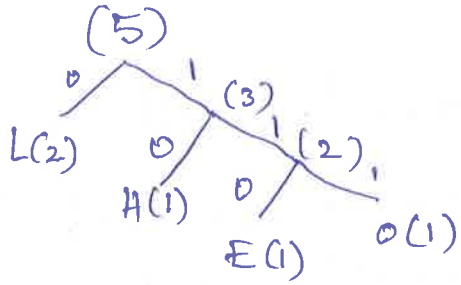
Shannon Fano algorithm

1. Sort symbols according to frequency of occurance.
2. Recursively divide the symbols into 2 parts, each with same number of counts, until the parts contain only one count(symbol).

Binary tree: Assign bits 0 to left branches
1 to right branches







Symbol	Count	$\log_2 \frac{1}{P_i}$	code	No. of bits used
L	2	1.32	0	2
H	1	2.32	10	2
E	1	2.32	110	3
O	1	2.32	111	3
			10 bits	10

Entropy

$$\eta = P_L \log_2 \frac{1}{P_L} + P_H \log_2 \frac{1}{P_H} + P_E \log_2 \frac{1}{P_E} + P_O \log_2 \frac{1}{P_O}$$

$$= 0.4 \log_2 \frac{1}{0.4} + 0.2 \log_2 \frac{1}{0.2} + 0.2 \log_2 \frac{1}{0.2} + 0.2 \log_2 \frac{1}{0.2}$$

$$= 1.92 \text{ bits/symbol.} \quad \hookrightarrow 2 \text{ bits/symbol.}$$

Huffman Coding – bottom up approach

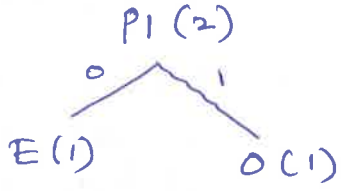
Algorithm

1. Initialization – put all symbols on the list sorted according their frequency counts
2. Repeat
   a. From the list, pick 2 symbols with lowest frequency counts. Form a Huffman subtree that has these two symbols as child nodes & create a parent node for them.
   b. Assign the sum of the children's freq. counts the parent & ~~insert~~ insert it into the list, such that the order is maintained.
   c. Delete the children from the list.
3. Assign codeword for each leaf based on the path from the root.

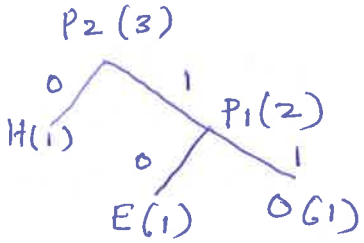
Word: HELLO

after initialization: L H E O

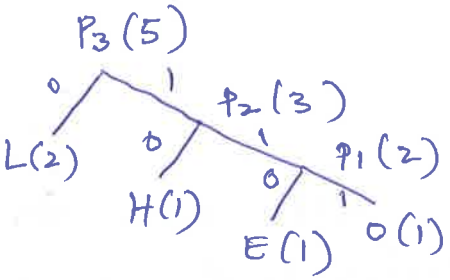
after iteration (a): L P1 H



after iteration (b): L P2



after iteration (c): P3



symbol	code
L	0
H	10
E	110
O	111

symbol	count	$\log_2 \frac{1}{p_i}$	code	No. of bits
L	2	1.32	00	4
H	1	2.32	01	2
E	1	2.32	10	2
O	1	2.32	11	2
				10

Bits/symbol = 10/5 = 2

Properties of Huffman coding

- Unique prefix property
  - No huffman code is prefix to any other huffman code
- Optimality
  - 2 least freqnent symbols will have same length of huffman codes, differing only in the last bits.
  - symbols that occur more frequently have shorter codes.
    if $p_i \geq p_j$ then $l_i \leq l_j$

Adaptive Huffman coding

Encoder

```
initial_code();
while not EOF
{
    get(c);
    encode(c);
    update_tree(c);
}
```

Decoder

```
initial_code();
while not EOF
{
    decode(c);
    output(c);
    update_tree(c);
}
```

Explanation

Initial_code

assigns symbols with some initially agreed upon codes without prior knowledge of frequency counts.

update_tree

1. increments frequency counts of symbols
2. updates configuration of the tree.

The tree should maintain sibling property.

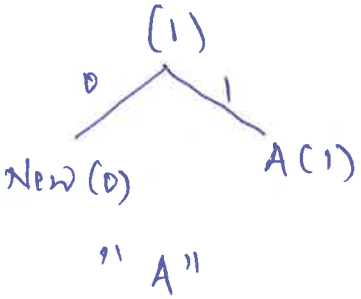
- All nodes (internal & leaf) are arranged in the order of increasing counts. Nodes are numbered in order from left to right, bottom to top.
- If sibling property is about to be violated, a swap procedure is invoked.
- Swap: A farthest node with count N is swapped with the node whose count has just been increased to N+1.

  If node with count is not a leaf node the entire subtree will go with it during the swap.

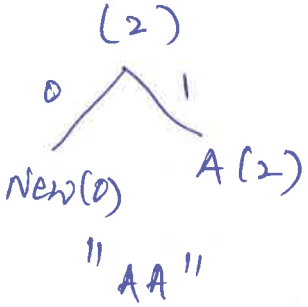
Example: Adaptive Huffman coding

String: A A D C C D D

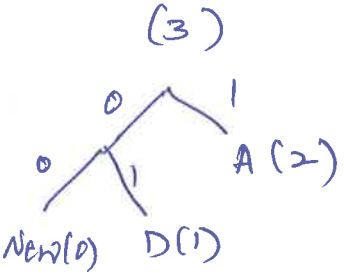
To begin with, a special symbol NEW: (0) is introduced. Its count (0) – always same.



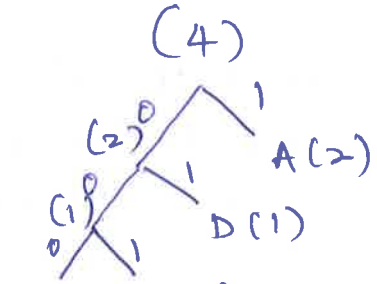
"A"



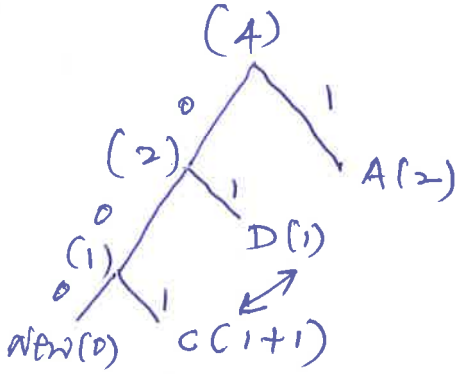
"AA"



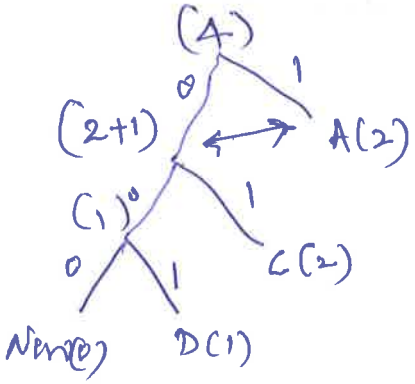
"AAD"



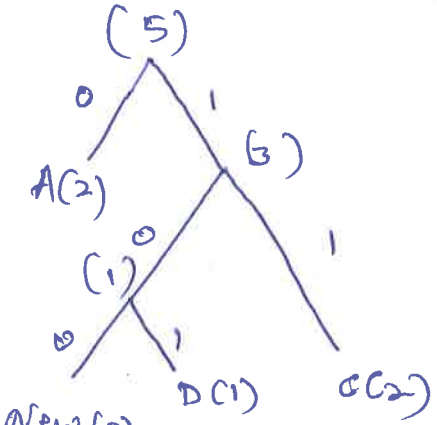
"AADC"



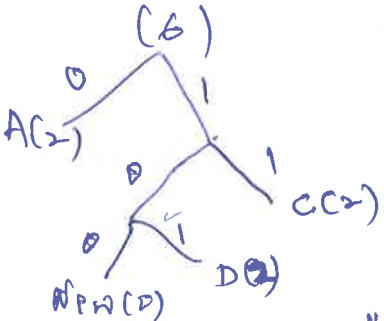
"AADCC" – (1)



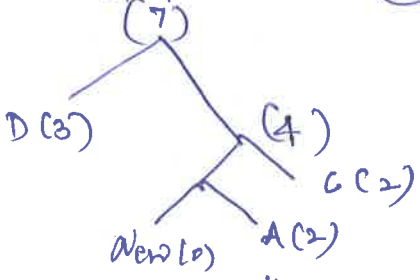
"AADCC" – (2)



"AADCC" – (3)



"AADCCD"



"AADCCDD"

Dictionary Based coding

LZW Compression Algo

```
Begin
  s = next input character;
  while not EOF
    {
      c = next input character;
      if s+c exists in the dictionary
       s = s+c
      else
        {
          output the code for s;
          add string s+c to dictionary
                          with a new code;
          s = c;
        }
    }
  output the code for s;
End
```

LZW Decompression

```
Begin
  s = NIL;
  while not EOF
    {
      k = next input code;
      entry = dictionary entry for k;
      output entry;
      if (s != NIL)
       add string s+entry[0] to dictionary
                          with a new code;
      s = entry;
    }
End
```

LZW

Compress string : ABABBABCABABBA ↑ 14 characters

Dictionary

code	string
1	A
2	B
3	C

S	C	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

output codes = {1, 2, 4, 5, 2, 3, 4, 6, 1}

Instead of 14 characters only 9 codes need to be sent. If we assume that character codes are transmitted as byte, then

compression ratio = $\frac{14}{9} = 1.56$

Example LZW decompression ABABBABCABABBA

Input codes: 1 2 4 5 2 3 4 6 1

S	k	entry/output
NIL	1	A
A	2	B
B	4	AB
AB	5	BA
BA	2	B
B	3	C
C	4	AB
AB	6	ABB
ABB	1	A
A	EOF	

code	string
1	A
2	B
3	C
4	AB
5	BA
6	ABB
7	BAB
8	BC
9	CA
10	ABA
11	ABBA

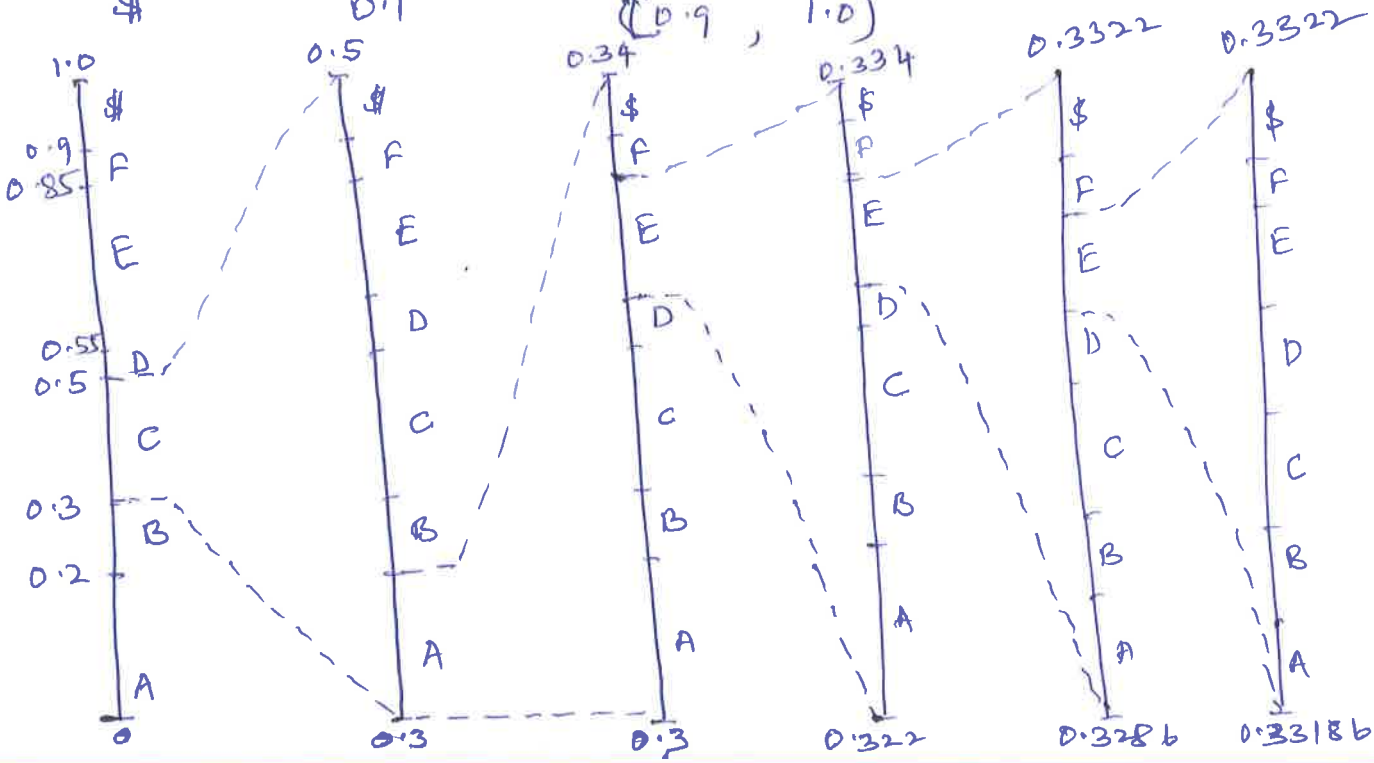
Decompressed output = ABABBABCABABBA

Arithmetic coding

Algorithm

```
Begin
    low = 0.0; high = 1.0; range = 1.0;
    while (symbol != terminator)
        {
            get (symbol);
            low = low + range * range_low (symbol);
            high = low + range * range_high (symbol);
            range = high - low;
        }
    output a code so that low ≤ code < high;
End
```

Symbol	Probability	Range
A	0.2	[0, 0.2)
B	0.1	[0.2, 0.3)
C	0.2	[0.3, 0.5)
D	0.05	[0.5, 0.55)
E	0.3	[0.55, 0.85)
F	0.05	[0.85, 0.9)
$	0.1	[0.9, 1.0)



Code for CAEE$

Symbol	low	high	range
	0	1.0	1.0
C	0.3	0.5	0.2
A	0.3	0.34	0.04
E	0.322	0.334	0.012
E	0.3286	0.3322	0.0036
$	0.33184	0.3322	0.00036

Initially low = 0, high = 1, range = 1.0

after first symbol C,

Range_low (c) = 0.3, range_high (c) = 0.5

so, low = 0 + 1.0 × 0.3 = 0.3

high = 0 + 1.0 × 0.5 = 0.5

finally,

range = $P_C \times P_A \times P_E \times P_E \times P_\$ = 0.2 * 0.2 \times 0.3 \times 0.3 \times 0.1 = 0.00036$

Lossy Compression

Distortion Measures

It is a mathematical measure that specifies how close an approximation to its original based on some distortion criteria.

Numerical distortion, MSE σ^2

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2$$

x_n - input data sequence, y_n - reconstructed output data sequence

N - Length of data sequence

If we consider the size of the relative error with signal, we use SNR, & ~~SQNR~~ PSNR

$$SNR = 10\log_{10}\frac{\sigma_x^2}{\sigma_d^2}$$

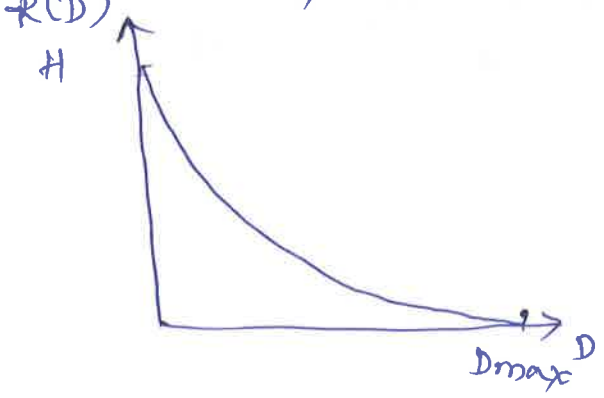
$$PSNR = 10\log_{10}\frac{x_{peak}^2}{\sigma_d^2}$$

Rate Distortion theory

Rate is the average number of bits required to represent each source symbol.

$R(D)$ = lowest rate in which the source data can be encoded while keeping the distortion bounded above by D.

When $D = 0$, we have lossless compression



Transform coding

Discrete cosine Transform (DCT)

2D DCT

Given a function $f(i,j)$ over 2 variables i, j, 2D DCT transforms it into a function $F(u,v)$ with u, v running over the ranges same as i and j.

$$F(u,v) = \frac{2\,C(u)\,C(v)}{\sqrt{MN}} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \cos\frac{(2i+1)u\pi}{2M} \cos\frac{(2j+1)v\pi}{2N} f(i,j)$$

where $i, u = 0, 1, \ldots M-1$

$j, v = 0, 1, \ldots N-1$

Constants $C(u)$ and $C(v)$ are defined by

$$C(\xi) = \begin{cases} \sqrt{2}/2 & \text{if } \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

2D DCT (used in JPEG compression)

$$F(u,v) = \frac{c(u)\,c(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} f(i,j)$$

where $i, j, u, v = 0, 1, \ldots 7$

2D IDCT

$$f(i,j) = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{c(u)\,c(v)}{4} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} F(u,v)$$

1D DCT

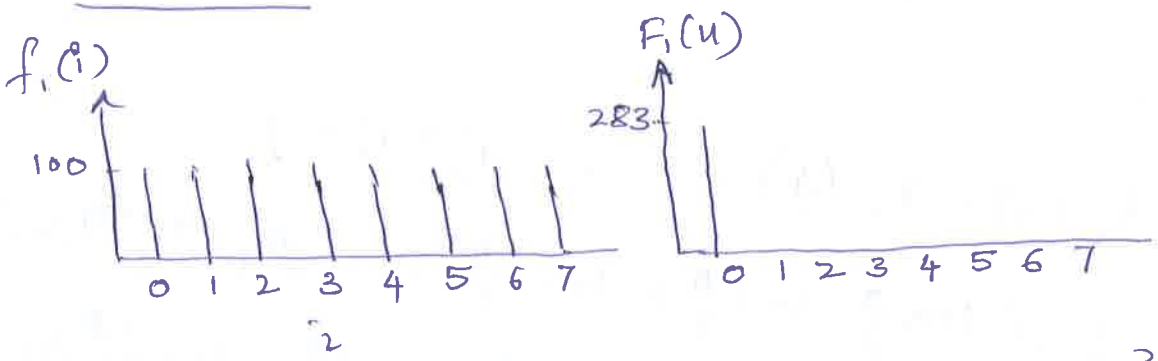
$$F(u) = \frac{c(u)}{2} \sum_{i=0}^{7} \cos \frac{(2i+1)u\pi}{16} f(i)$$

$i, u = 0, 1, \cdots 7$

1D IDCT

$$f(i) = \sum_{u=0}^{7} \frac{c(u)}{2} \cos \frac{(2i+1)u\pi}{16} F(u)$$

Example



$$f_1(i) = \{100, 100, 100, 100, 100, 100, 100, 100\}$$

$$F_1(u) = \frac{c(u)}{2} \sum_{i=0}^{7} \cos \frac{(2i+1)u\pi}{16} f(i)$$

$i = 0, 1, \cdots 7 \quad u = 0, 1, 2, \cdots 7, \quad c(0) = \frac{\sqrt{2}}{2}$

$$F_1(0) = \frac{\sqrt{2}}{2} (\cos 0 \times 100 + \cos(0) \times 100 + \cos(0) \times 100 + \cos 0 \times 100 + \cos 0 \times 100 + \cos 0 \times 100 + \cos 0 \times 100 + \cos 0 \times 100)$$

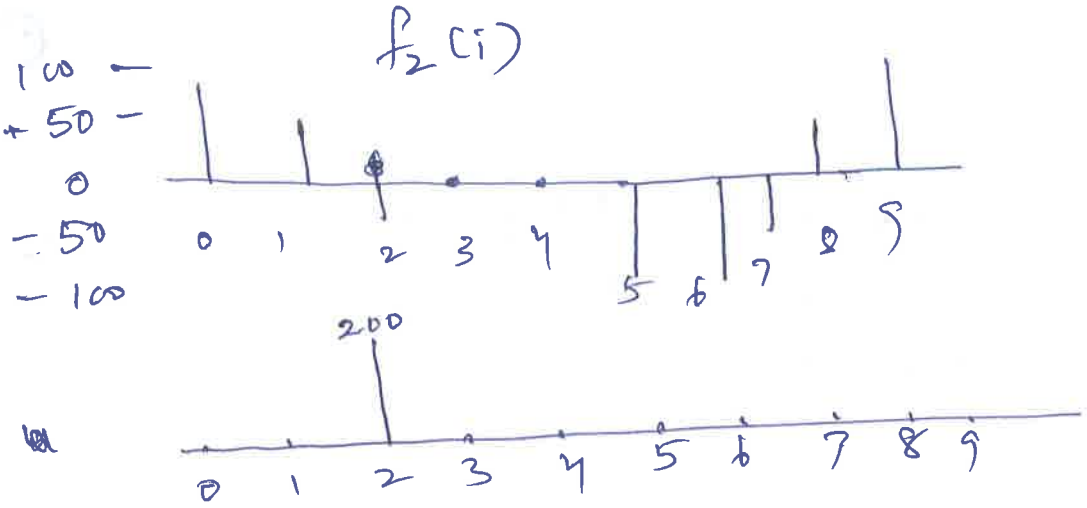
$$= \frac{\sqrt{2}}{2.2} (100+100+100+100+100+100+100+100)$$

$$\hookrightarrow 283$$

$$F_1(1) = F_1(2) = F_1(3) = F_1(4) = F_1(5) = F_1(6) = F_1(7) = 0$$

~~Scratch~~ $c_1(u) = 1$

$$F_1(1) = \frac{1}{2} \left(\cos \frac{\pi}{16} \cdot 100 + \cos \frac{3\pi}{16} \cdot 100 + \cos \frac{5\pi}{16} \cdot 100 + \cos \frac{7\pi}{16} \cdot 100 + \cos \frac{9\pi}{16} \cdot 100 + \cos \frac{11\pi}{16} \cdot 100 + \cos \frac{13\pi}{16} \cdot 100 + \cos \frac{15\pi}{16} \cdot 100\right) = 0$$



$u=0$

$$F_2(0) = \frac{\sqrt{2}}{2\cdot 2}\cdot 1\cdot\left(100\cos\frac{\pi}{8} + 100\cos\frac{3\pi}{8} + 100\cos\frac{5\pi}{8} + 100\cos\frac{7\pi}{8}\right.$$
$$\left. + 100\cos\frac{9\pi}{8} + 100\cos\frac{11\pi}{8} + 100\cos\frac{13\pi}{8} + 100\cos\frac{15\pi}{8}\right)$$

$$= 0$$

$$F_2(1) = F_2(3) = F_2(4) = \cdots = F_2(7) = 0$$

$$F_2(2) = \frac{1}{2}\cdot\left(\cos\frac{\pi}{8}\cdot\cos\frac{\pi}{8} + \cos\frac{3\pi}{8}\cdot\cos\frac{3\pi}{8} + \frac{\cos 5\pi}{8}\cdot\cos\frac{5\pi}{8}\right.$$
$$+ \cos\frac{2\pi}{8}\cos\frac{7\pi}{8} + \cos\frac{9\pi}{8}\cos\frac{9\pi}{8} + \cos\frac{11\pi}{8}\cos\frac{11\pi}{8}$$
$$\left. + \cos\frac{13\pi}{8}\cos\frac{13\pi}{8} + \cos\frac{15\pi}{8}\cos\frac{15\pi}{8}\right)100$$

$$= \frac{1}{2}(1+1+1+1)100 = 200$$