Image Coding

Image Coding (Compression) - techniques used to reduce the amount of data required to represent an image as a digital string of bits

Image Coding

Image Coding (Compression) – techniques used to reduce the amount of data required to represent an image as a digital string of bits

Useful to code images for

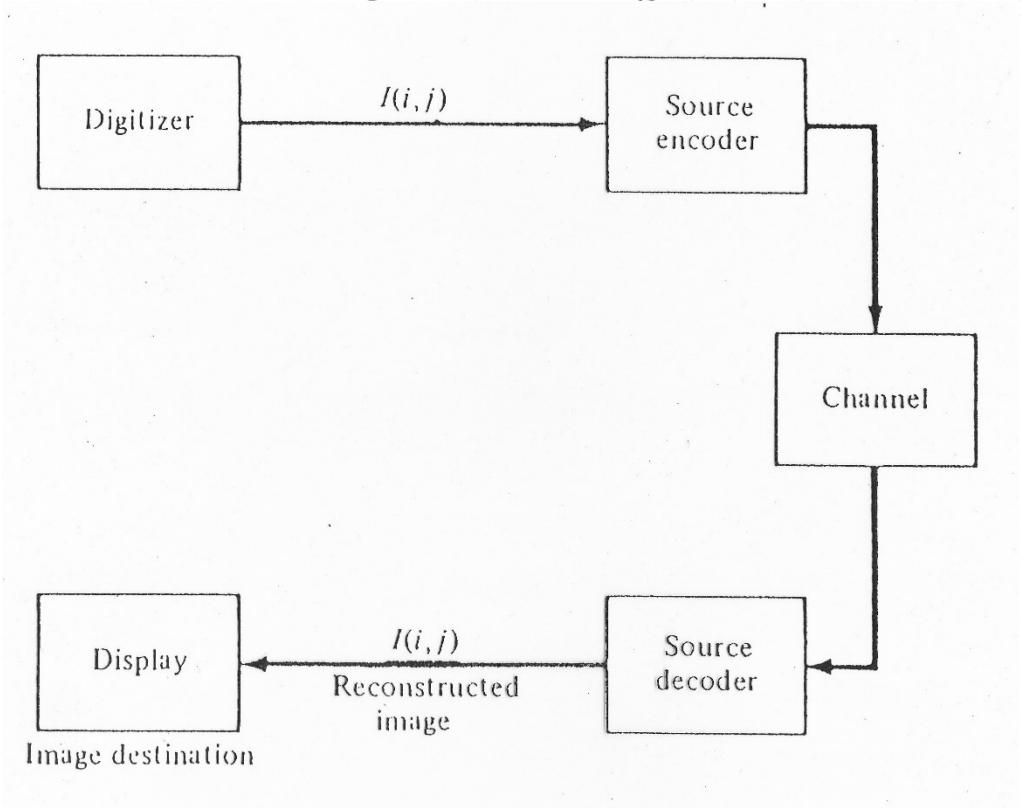
i) digital storage ii) digital transmission

Image Coding

Image Coding (Compression) – techniques used to reduce the amount of data required to represent an image as a digital string of bits

Useful to code images for

i) digital storage ii) digital transmission



Lossless techniques – coded images can be reconstructed exactly

Lossless techniques – coded images can be reconstructed exactly

Lossy techniques – coded images cannot be reconstructed exactly

Lossless techniques – coded images can be reconstructed exactly

Lossy techniques – coded images cannot be reconstructed exactly

Goal is to reduce the amount of data as much as possible while preserving the fidelity of the data as much as possible.

Lossless techniques – coded images can be reconstructed exactly

Lossy techniques – coded images cannot be reconstructed exactly

Goal is to reduce the amount of data as much as possible while preserving the fidelity of the data as much as possible.

Significant compression is possible because images have a great deal of structure (redundancy). Usually adjacent pixel values are highly correlated.

Lossless techniques – coded images can be reconstructed exactly

Lossy techniques – coded images cannot be reconstructed exactly

Goal is to reduce the amount of data as much as possible while preserving the fidelity of the data as much as possible.

Significant compression is possible because images have a great deal of structure (redundancy). Usually adjacent pixel values are highly correlated.

One simple compression technique is to subsample the image by taking every nth pixel. (lossy)

Lossless techniques – coded images can be reconstructed exactly

Lossy techniques – coded images cannot be reconstructed exactly

Goal is to reduce the amount of data as much as possible while preserving the fidelity of the data as much as possible.

Significant compression is possible because images have a great deal of structure (redundancy). Usually adjacent pixel values are highly correlated.

One simple compression technique is to subsample the image by taking every nth pixel. (lossy)

For image sequences, we can take every nth frame. (lossy)

Run-Length Encoding exploits the fact that most rows in binary images contain long strings of identical pixel values.

Run-Length Encoding exploits the fact that most rows in binary images contain long strings of identical pixel values.

(Ex) 000001111111111000

Can code as 5 9 3 if we assume the row starts with 0

Run-Length Encoding exploits the fact that most rows in binary images contain long strings of identical pixel values.

(Ex) 000001111111111000

Can code as 5 9 3 if we assume the row starts with 0

Run-Length Encoding works well if most of the strings are long.

```
00000000000000000000000000011111111111111000000000     28 14  9
00000000000000000000000000111111111111111110000000     26 18  7
00000000000000000000000001111111111111111111110000     23 24  4
00000000000000000000000011111111111111111111111000     22 26  3
00000000000000000000001111111111111111111111111110     20 30  1
00000000000000000000111111100000000000000000001111111  19  7 18 7
00000000000000000001111100000000000000000000011111     19  5 22 5
00000000000000000011100000000000000000000000000111     19  3 26 3
00000000000000000011100000000000000000000000000111     19  3 26 3
00000000000000000011100000000000000000000000000111     19  3 26 3
00000000000000000011100000000000000000000000000111     19  3 26 3
00000000000000000011110000000000000000000000001110     20  4 23 3 1
00000000000000000001110000000000000000000000111000     22  3 20 3 3
01111111111111111111111111111111111111111111111111111  1  50
01111111111111111111111111111111111111111111111111111  1  50
01111111111111111111111111111111111111111111111111111  1  50
01111111111111111111111111111111111111111111111111111  1  50
01111111111111111111111111111111111111111111111111111  1  50
01100000000000000000000000000000000000000000000000011  1   2 46 2
```

00000000000000000000000000011111111111111000000000	28 14 9
00000000000000000000000000111111111111111110000000	26 18 7
00000000000000000000000001111111111111111111110000	23 24 4
00000000000000000000000011111111111111111111111000	22 26 3
00000000000000000000001111111111111111111111111110	20 30 1
00000000000000000000111111100000000000000000001111111	19 7 18 7
00000000000000000001111100000000000000000000011111	19 5 22 5
00000000000000000001110000000000000000000000000111	19 3 26 3
00000000000000000001110000000000000000000000000111	19 3 26 3
00000000000000000001110000000000000000000000000111	19 3 26 3
00000000000000000001110000000000000000000000000111	19 3 26 3
00000000000000000001111000000000000000000000001110	20 4 23 3 1
00000000000000000000111000000000000000000000111000	22 3 20 3 3
01111111111111111111111111111111111111111111111111	1 50
01111111111111111111111111111111111111111111111111	1 50
01111111111111111111111111111111111111111111111111	1 50
01111111111111111111111111111111111111111111111111	1 50
01111111111111111111111111111111111111111111111111	1 50
01100000000000000000000000000000000000000000000011	1 2 46 2

Image has 19x51 = 969 bits

0000000000000000000000000001111111111111000000000	28 14 9
0000000000000000000000000011111111111111110000000	26 18 7
0000000000000000000000001111111111111111111110000	23 24 4
0000000000000000000000011111111111111111111111000	22 26 3
0000000000000000000001111111111111111111111111110	20 30 1
0000000000000000000011111110000000000000000001111111	19 7 18 7
0000000000000000000011111000000000000000000000011111	19 5 22 5
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011110000000000000000000000001110	20 4 23 3 1
0000000000000000000001110000000000000000000000111000	22 3 20 3 3
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0110000000000000000000000000000000000000000000000011	1 2 46 2

Image has 19x51 = 969 bits

Run-Length Encoding has 63 counts.

0000000000000000000000000001111111111111000000000	28 14 9
0000000000000000000000000011111111111111110000000	26 18 7
0000000000000000000000001111111111111111111110000	23 24 4
0000000000000000000000011111111111111111111111000	22 26 3
0000000000000000000001111111111111111111111111110	20 30 1
0000000000000000000011111110000000000000000001111111	19 7 18 7
0000000000000000000011111000000000000000000000011111	19 5 22 5
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011110000000000000000000000001110	20 4 23 3 1
0000000000000000000001110000000000000000000000111000	22 3 20 3 3
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0110000000000000000000000000000000000000000000000011	1 2 46 2

Image has 19x51 = 969 bits

Run-Length Encoding has 63 counts.

Need 6 bits/count

0000000000000000000000000001111111111111000000000	28 14 9
0000000000000000000000000011111111111111110000000	26 18 7
0000000000000000000000001111111111111111111110000	23 24 4
0000000000000000000000011111111111111111111111000	22 26 3
0000000000000000000001111111111111111111111111110	20 30 1
0000000000000000000011111110000000000000000001111111	19 7 18 7
0000000000000000000011111000000000000000000000011111	19 5 22 5
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011110000000000000000000000001110	20 4 23 3 1
0000000000000000000001110000000000000000000000111000	22 3 20 3 3
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0110000000000000000000000000000000000000000000000011	1 2 46 2

Image has 19x51 = 969 bits

Run-Length Encoding has 63 counts.

Need 6 bits/count

Run-Length Encoding requires 63x6 = 378 bits

0000000000000000000000000001111111111111000000000	28 14 9
0000000000000000000000000011111111111111110000000	26 18 7
0000000000000000000000001111111111111111111110000	23 24 4
0000000000000000000000011111111111111111111111000	22 26 3
0000000000000000000001111111111111111111111111110	20 30 1
0000000000000000000011111110000000000000000001111111	19 7 18 7
0000000000000000000011111000000000000000000000011111	19 5 22 5
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011100000000000000000000000000111	19 3 26 3
0000000000000000000011110000000000000000000000001110	20 4 23 3 1
0000000000000000000000111000000000000000000000111000	22 3 20 3 3
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0111111111111111111111111111111111111111111111111111	1 50
0110000000000000000000000000000000000000000000000011	1 2 46 2

Image has 19x51 = 969 bits

Run-Length Encoding has 63 counts.

Need 6 bits/count

Run-Length Encoding requires 63x6 = 378 bits

Lossless

Variable-Length Encoding

When representing gray level images, instead of using a fixed number, e.g. 8, of bits per pixel, use only a few bits for pixel values that appear often and more bits for pixel values that appear rarely.

Variable-Length Encoding

When representing gray level images, instead of using a fixed number, e.g. 8, of bits per pixel, use only a few bits for pixel values that appear often and more bits for pixel values that appear rarely.

(Ex) Suppose pixel values are 0 to 26.
This requires 5 bits per pixel.

Variable-Length Encoding

When representing gray level images, instead of using a fixed number, e.g. 8, of bits per pixel, use only a few bits for pixel values that appear often and more bits for pixel values that appear rarely.

(Ex) Suppose pixel values are 0 to 26.
This requires 5 bits per pixel.

11 pixels

1	2	18	1	3	1	4	1	2	18	1
A	B	R	A	C	A	D	A	B	R	A

Variable-Length Encoding

When representing gray level images, instead of using a fixed number, e.g. 8, of bits per pixel, use only a few bits for pixel values that appear often and more bits for pixel values that appear rarely.

(Ex) Suppose pixel values are 0 to 26.
This requires 5 bits per pixel.

11 pixels

1	2	18	1	3	1	4	1	2	18	1
A	B	R	A	C	A	D	A	B	R	A

standard binary code 00001|00010|10010|00001|00011|...
55 bits

1	2	18	1	3	1	4	1	2	18	1
A	B	R	A	C	A	D	A	B	R	A

Pixel Value	#Appearances
1	5
2	2
18	2
3	1
4	1

1	2	18	1	3	1	4	1	2	18	1
A	B	R	A	C	A	D	A	B	R	A

Pixel Value	#Appearances
1	5
2	2
18	2
3	1
4	1

Approach – assign shortest codes to most common pixel values

Pixel Value	Code
1	0
2	1
18	01
3	10
4	11

1	2	18	1	3	1	4	1	2	18	1
A	B	R	A	C	A	D	A	B	R	A

Pixel Value	Code
1	0
2	1
18	01
3	10
4	11

1	2	18	1	3	1	4	1	2	18	1
A	B	R	A	C	A	D	A	B	R	A

Pixels would get encoded as

0 1 0 1 0 1 0 0 1 1 0 1 0 1 0 (15 bits)

Pixel Value	Code
1	0
2	1
18	01
3	10
4	11

1	2	18	1	3	1	4	1	2	18	1
A	B	R	A	C	A	D	A	B	R	A

Pixel Value	Code
1	0
2	1
18	01
3	10
4	11

Pixels would get encoded as

0 1 0 1 0 1 0 0 1 1 0 1 0 1 0 (15 bits)

But this doesn't work. The string could be decoded as

18 18 18 1 18 2 18 18 1

1	2	18	1	3	1	4	1	2	18	1
A	B	R	A	C	A	D	A	B	R	A

Pixel Value	Code
1	0
2	1
18	01
3	10
4	11

Pixels would get encoded as

0 1 0 1 0 1 0 0 1 1 0 1 0 1 0 (15 bits)

But this doesn't work. The string could be decoded as

18 18 18 1 18 2 18 18 1

We need to design the code so that no pixel code is the prefix of another.

1	2	18	1	3	1	4	1	2	18	1
A	B	R	A	C	A	D	A	B	R	A

Pixel Value	Code
1	0
2	1
18	01
3	10
4	11

Pixels would get encoded as

0 1 0 1 0 1 0 0 1 1 0 1 0 1 0 (15 bits)

But this doesn't work. The string could be decoded as

18 18 18 1 18 2 18 18 1

We need to design the code so that no pixel code is the prefix of another.

(Ex)

Pixel Value	Code
1	11
2	00
18	011
3	010
4	10

String would be encoded as

11 00 011 11 010 11 10 11 00 011 11 (25 bits)

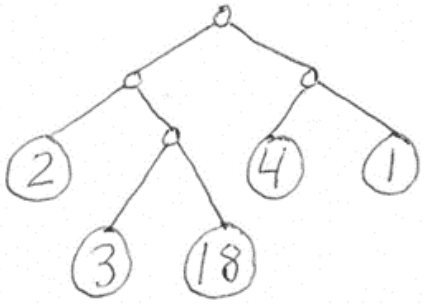
Now decoding is unique

String would be encoded as

11 00 011 11 010 11 10 11 00 011 11 (25 bits)

Now decoding is unique

Can represent this code using a trie



0 = go left
1 = go right

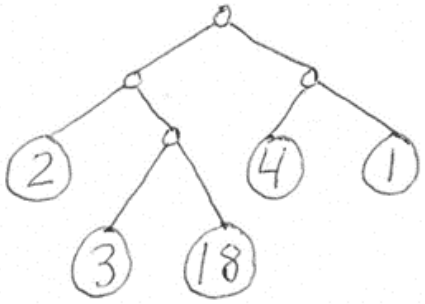
2 = 00
3 = 010
18 = 011
4 = 10
1 = 11

String would be encoded as

11 00 011 11 010 11 10 11 00 011 11 (25 bits)

Now decoding is unique

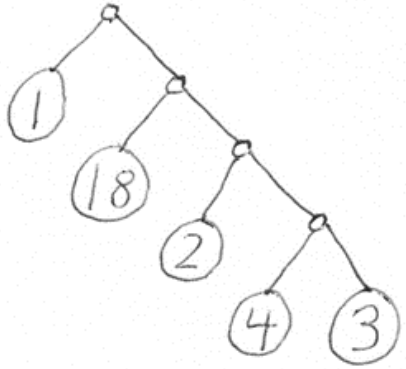
Can represent this code using a trie



0 = go left
1 = go right

2 = 00
3 = 010
18 = 011
4 = 10
1 = 11

Another possibility



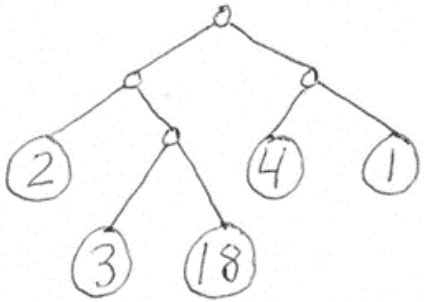
1 = 0
18 = 10
2 = 110
4 = 1110
3 = 1111

String would be encoded as

11 00 011 11 010 11 10 11 00 011 11 (25 bits)

Now decoding is unique

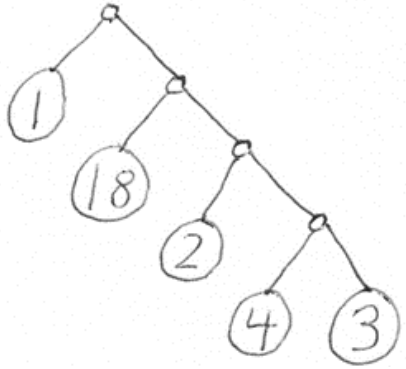
Can represent this code using a trie



0 = go left
1 = go right

2 = 00
3 = 010
18 = 011
4 = 10
1 = 11

Another possibility



1 = 0
18 = 10
2 = 110
4 = 1110
3 = 1111

Pixels would get encoded as

0 110 10 0 1111 0 1110 0 110 10 0 (23 bits)

How do we determine the best code (trie)?

How do we determine the best code (trie)?

Frequently used values should be near the top, rarely used values should be near the bottom.

How do we determine the best code (trie)?

Frequently used values should be near the top, rarely used values should be near the bottom.

Huffman Coding

(Ex) "A SIMPLE STRING TO BE ENCODED USING A MINIMAL NUMBER OF BITS"

How do we determine the best code (trie)?

Frequently used values should be near the top, rarely used values should be near the bottom.

Huffman Coding

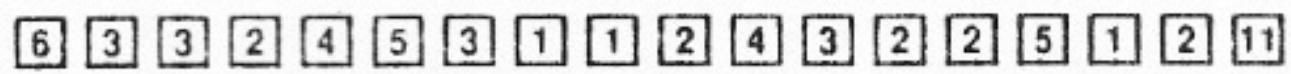
(Ex) "A SIMPLE STRING TO BE ENCODED USING A MINIMAL NUMBER OF BITS"

First build a histogram.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
count[k]	11	3	3	1	2	5	1	2	0	6	0	0	2	4	5	3	1	0	2	4	3	2	0	0	0	0	0

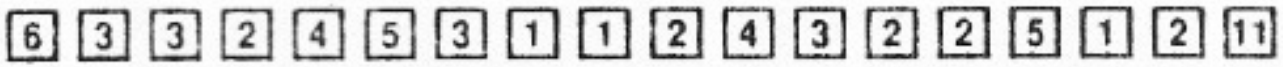
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
count[k]	11	3	3	1	2	5	1	2	0	6	0	0	2	4	5	3	1	0	2	4	3	2	0	0	0	0	0

Create a node for each nonzero count

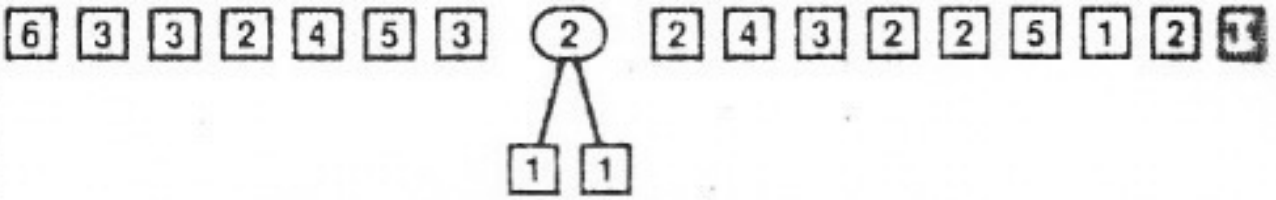


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
count[k]	11	3	3	1	2	5	1	2	0	6	0	0	2	4	5	3	1	0	2	4	3	2	0	0	0	0	0

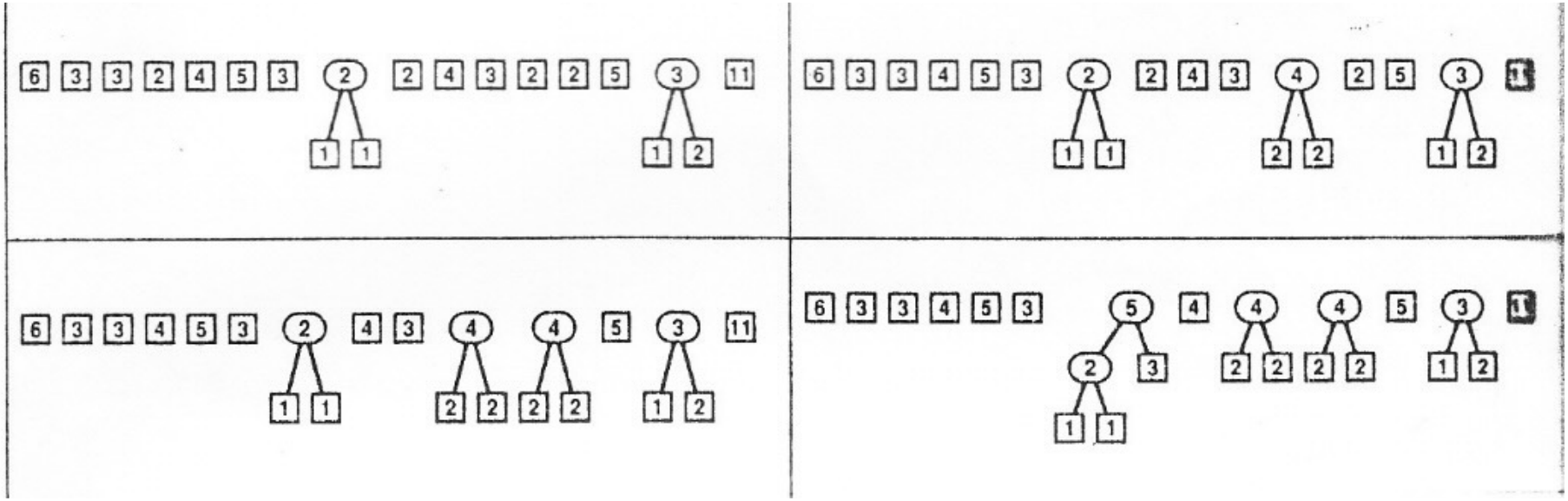
Create a node for each nonzero count



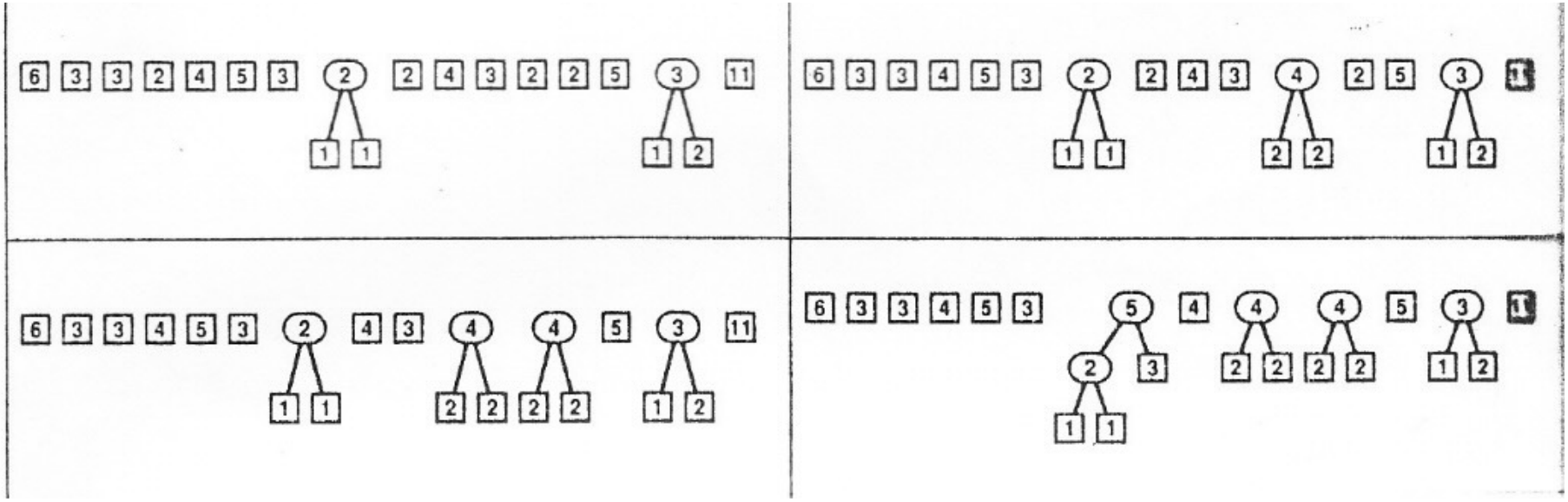
Find the 2 top level nodes with the smallest counts and create a new node with these 2 nodes as children and with a count the sum of the counts of the children.

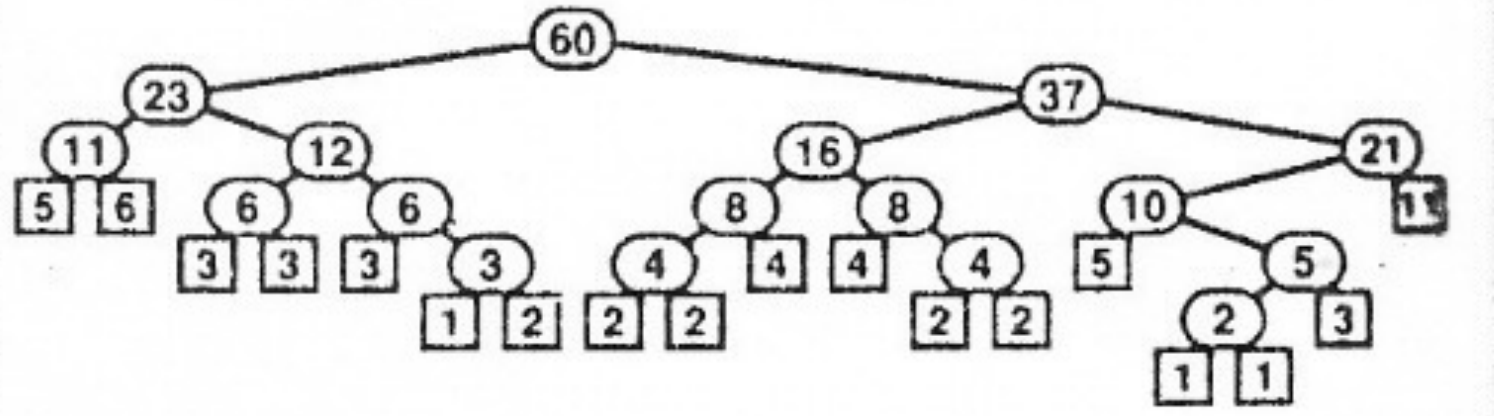


Repeat until we get a single trie.

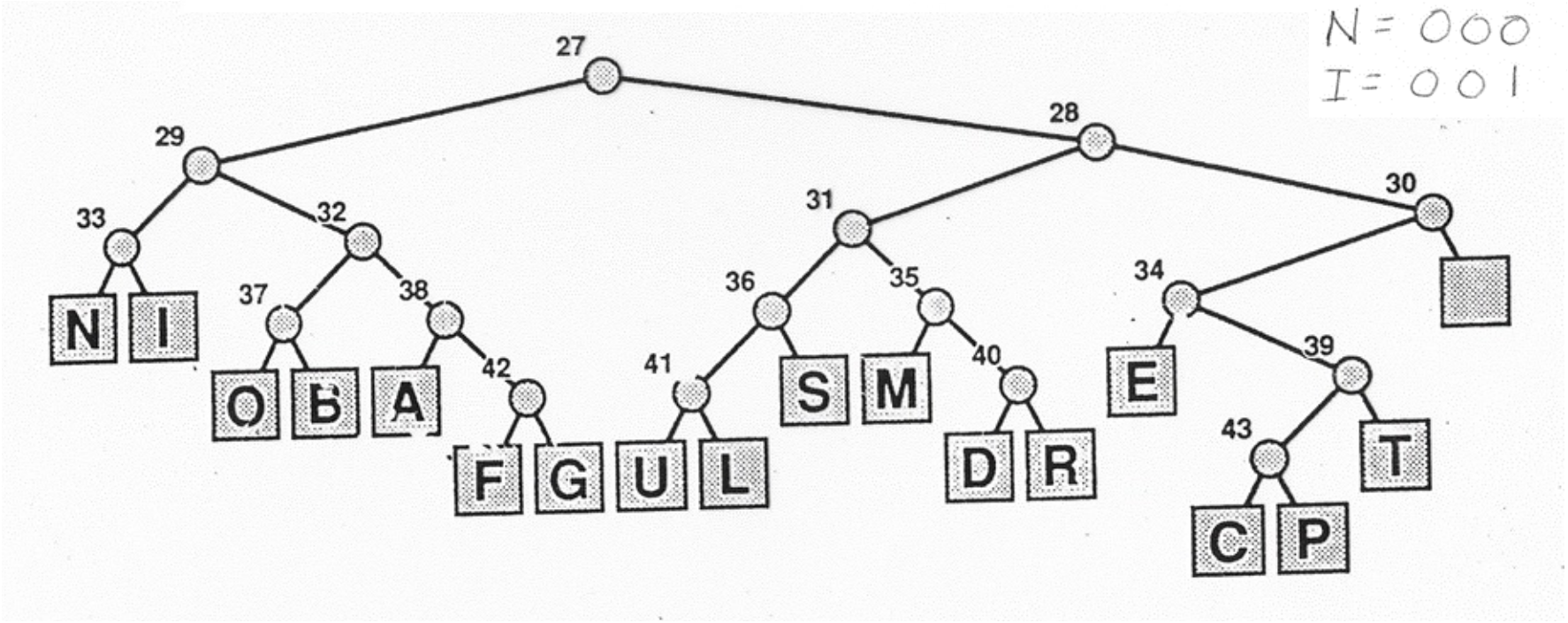


Repeat until we get a single trie.

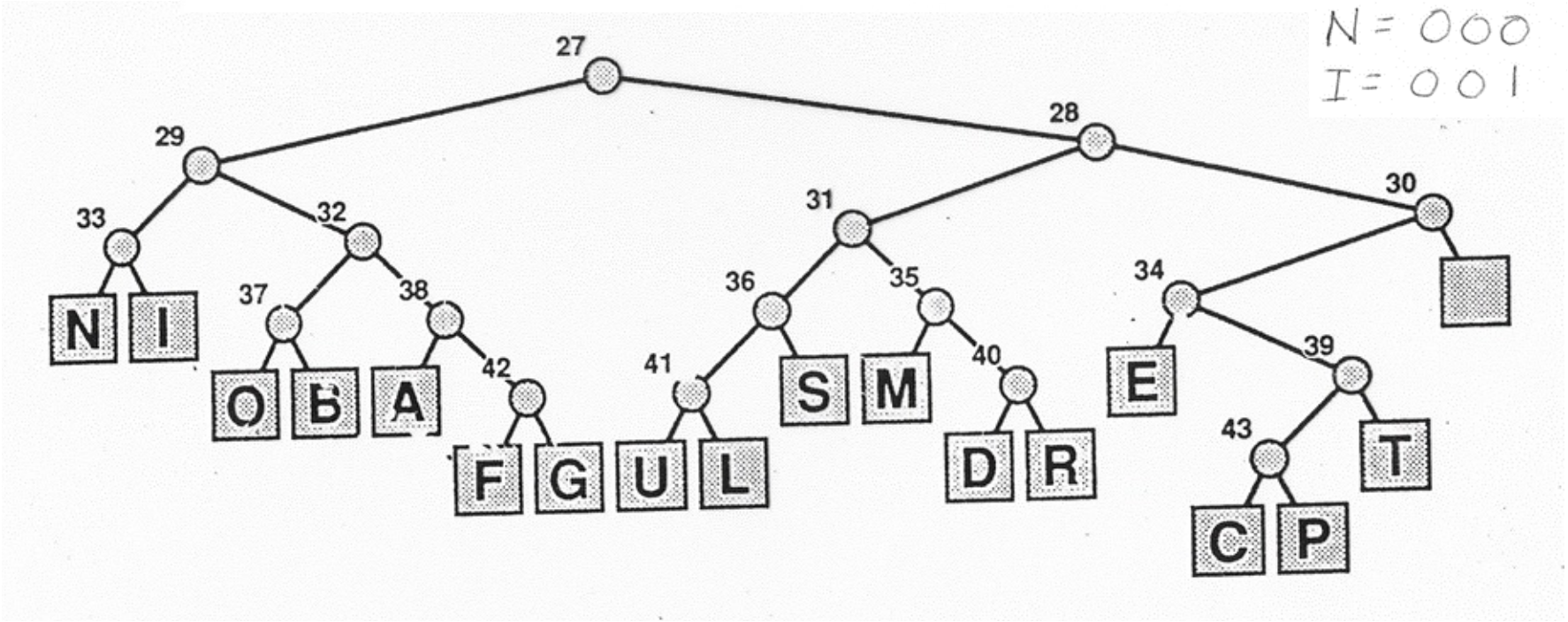




Replace the frequencies at the bottom nodes with the associated letters and view as an encoding trie.



Replace the frequencies at the bottom nodes with the associated letters and view as an encoding trie.



The resolting trie gives the Huffman code which uses the minimum number of bits.