ECI750 Multimedia Data Compression

Lecture 5

Huffman Coding – 2

Dr. Shadan Khattak

Department of Electrical Engineering

COMSATS Institute of Information Technology - Abbottabad

Huffman Coding

- In this lecture, we will study
  - How can we generate Huffman codes when the probability model of the source is unknown (Adaptive Huffman Codes)?
  - What are other related approaches?
  - How can Huffman coding be used for image compression, audio compression, and text compression?

Huffman Coding

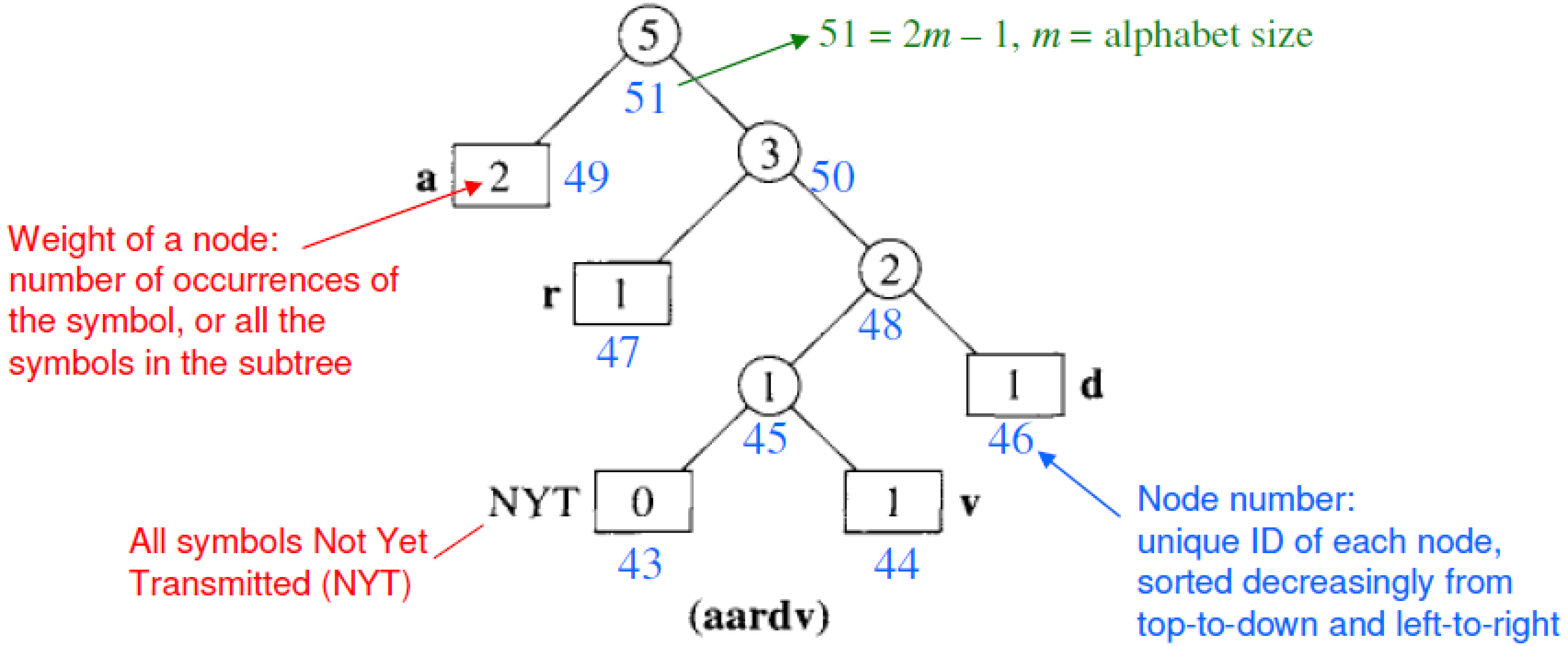
- <u>**Adaptive Huffman Codes**</u>
  - Huffman coding requires knowledge of the probabilities of the source sequence.
  - If this knowledge is not available, Huffman coding becomes a two-pass procedure:
    - Statistics are collected in the first pass
    - Source is encoded in the second pass
  - Faller [1] and Gallagher [2] developed adaptive algorithms to construct the Huffman code based on the statistics of the symbols already encountered.
    - These were later improved by Knuth [3] and Vitter [4].

Huffman Coding

- **<u>Adaptive Huffman Codes</u>**
  - In adaptive Huffman Codes, we add two new parameters to the binary tree:
    - ***Weight:***
      - The weight of each external node is the number of times the symbol corresponding to the leaf has been encountered.
      - The weight of each internal node is the sum of the weights of its offspring.
    - ***Node number:***
      - The node number y_i is a unique number assigned to each internal and external node.
  - If we have an alphabet of size n, then the number of internal and external nodes is $2n - 1$ with node numbers: $y_1, \dots, y_{2n-1}$
  - If x_j is the weight of the node y_j, we have $x_1 \leq x_2 \leq \cdots \leq x_{2n-1}$
  - ***Sibling property:***
    - The nodes y_{2j-1} and y_{2j} are offspring of the same parent node, or siblings, for $1 \leq j < n$
    - the node number for the parent node is greater than y_{2j-1} and y_{2j}
    - Any tree that possesses this property is a Huffman tree.
  - ***Block***: The set of nodes with the same weight makes up a block.

Huffman Coding

- **<u>Adaptive Huffman Codes (3)</u>**



Huffman Coding

- **<u>Adaptive Huffman Codes</u>**
  - Neither transmitter nor receiver knows anything about the statistics of the source sequence at the start of transmission.
  - The tree at both the transmitter and the receiver consists of a single node that corresponds to all symbols not yet transmitted (NYT) and has a weight zero.
  - Before transmission, a fixed code for each symbol is agreed upon between transmitter and receiver.

Huffman Coding

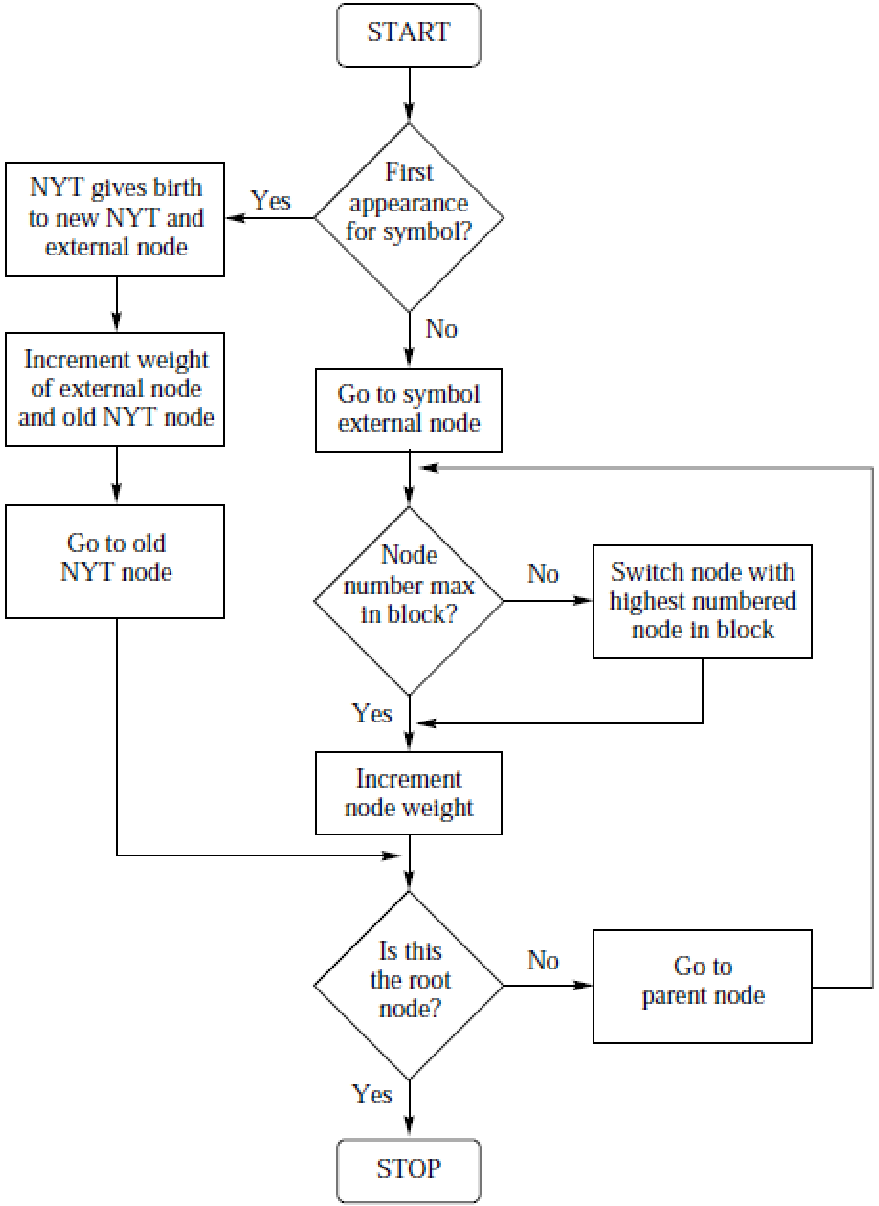
- **<u>Adaptive Huffman Codes</u>**

  <u>Initial Codewords</u>

  - For an alphabet $(a_1, a_2, \ldots, a_m)$ of size m,
    - Pick e and r such that:
      - $m = 2^e + r$
      - $0 \leq r < 2^e$
  - A letter a_k is encoded as:
    - the $(e+1)$-bit binary representation of $k-1$, if $1 \leq k \leq 2r$
    - else, the e-bit binary representation of $k-r-1$.
  - For example, for $m = 26$:
    - $e = 4, r = 10$
    - $a_1 = 00000$, $a_2 = 00001$, $a_{22} = 1011$

Huffman Coding

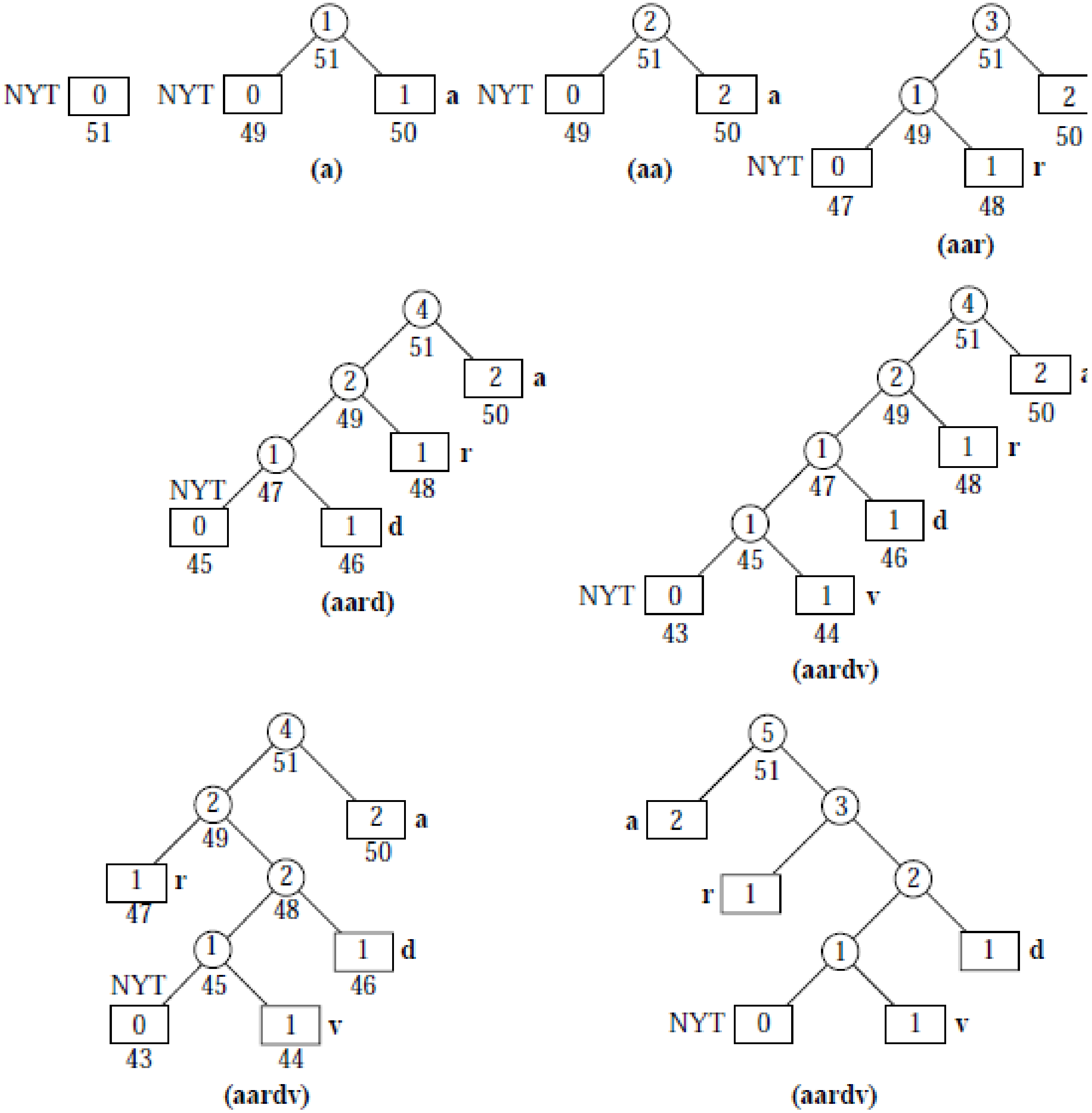
- **<u>Adaptive Huffman Codes</u>**
  - ***Update Procedure:***



Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

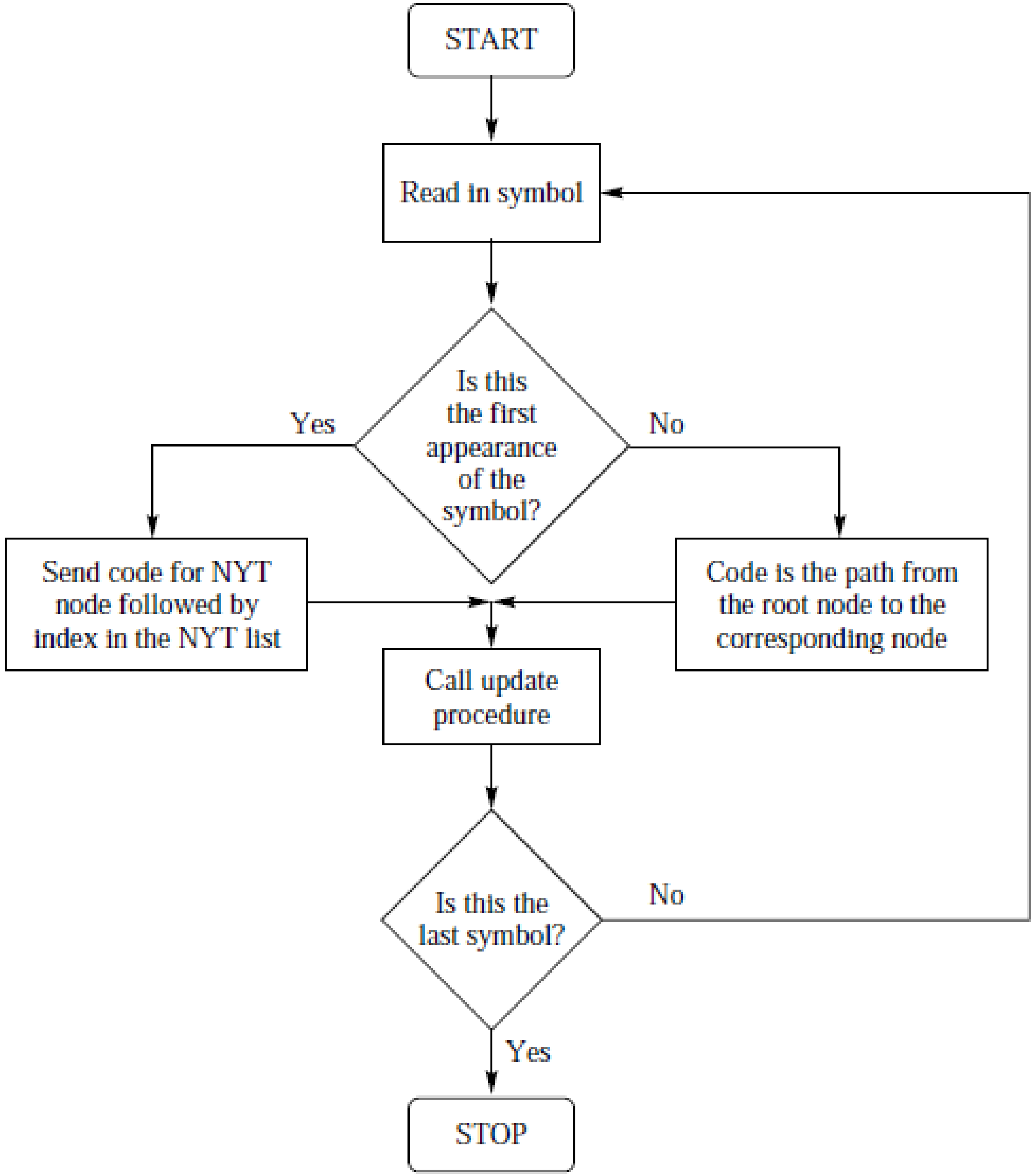
Huffman Coding

- **Adaptive Huffman Codes**
  - ***Update Procedure:***
  - ***Example:*** $aardv$



Huffman Coding

- **Adaptive Huffman Codes**
  - ***Encoding Procedure:***



Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

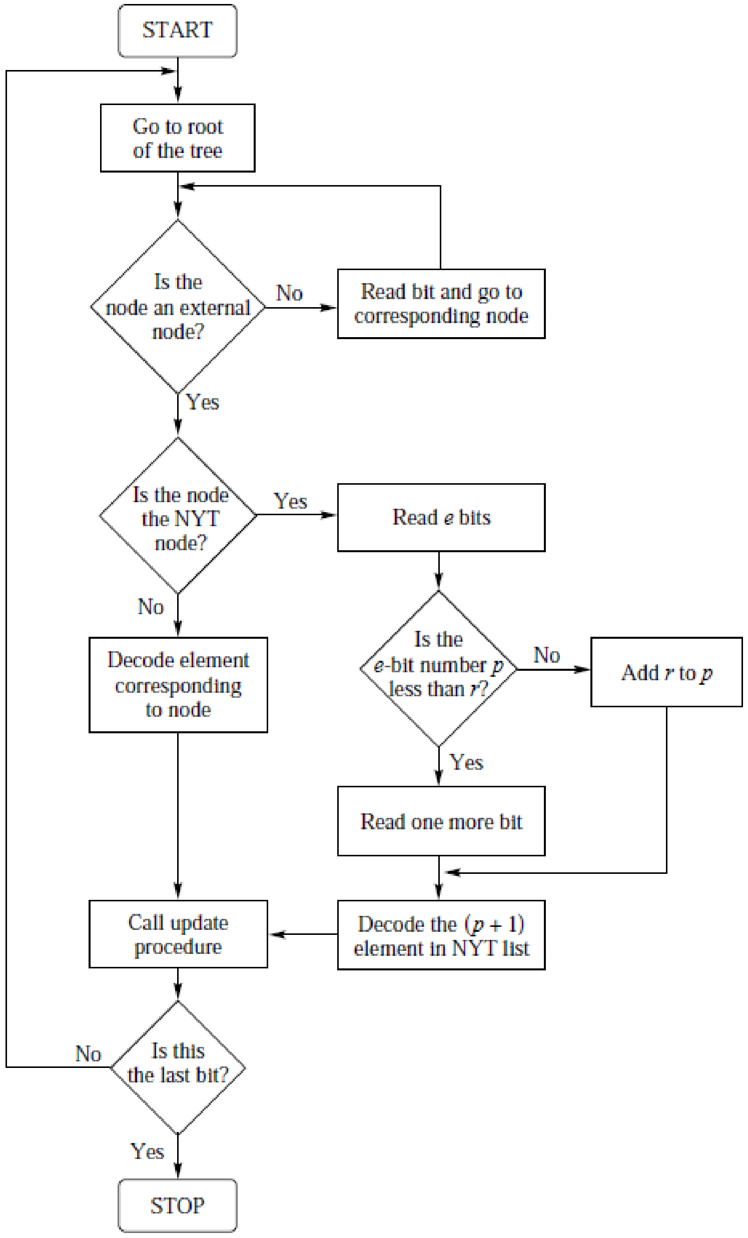
- **<u>Adaptive Huffman Codes</u>**
  - ***Encoding Procedure:***
  - ***Example:*** $\boldsymbol{aardva}$
    - $a = 00000$
    - $a = 1$
    - $r = 010001$
    - $d = 0000011$
    - $v = 0001011$
    - $a = 0$

    Code to transmit: 000001010001000001100010110

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **<u>Adaptive Huffman Codes</u>**
  - ***Decoding Procedure:***



Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

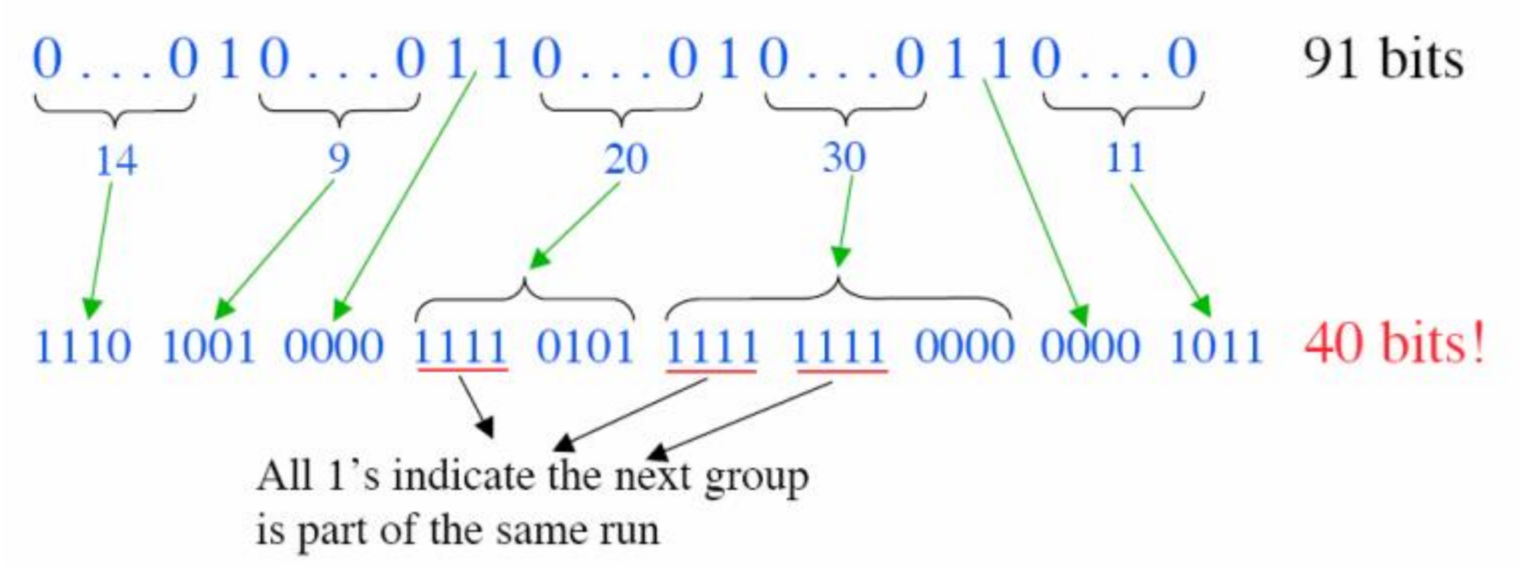
- <u>**Adaptive Huffman Codes**</u>
  - ***Decoding Procedure:***
  - ***Example:*** 00000101000100000110001011 0
    - $00000 = a$
    - $1 = a$
    - $010001 = r$
    - $0000011 = d$
    - $0001011 = v$
    - $0 = a$

Huffman Coding

- **Run-Length Coding (RLE)**
  - In many sources, it is possible to have consecutive identical symbols.
  - It is not efficient to encode each of these symbols separately.
  - Hence, the repeated value is generally coded once along with the number of times it appears.
  - For example, in a binary sequence, if 0’s are more probable, then we can code the run length of 0’s using k-bits and transmit the code.
  - In this case, we will not transmit the runs of 1’s.

Huffman Coding

- **Run-Length Coding (RLE)**
  - Suppose $k = 4$.



  - Is there a more efficient way of encoding the run lengths?
    - Why fixed length codes?

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **Unary Codes**
  - A simple code for integers which uses the following coding scheme:
    - Codeword for an integer n is n number of 1s followed by a 0.
    - E.g., codeword for 4 is 11110 and for 7 is 11111110.
  - Unary code is optimal when $A = \{1, 2, 3, \dots\}$ and $P[k] = \frac{1}{2^k}$

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **Golomb Codes**
  - A family of codes parameterized by an integer $m > 0$.
  - In the Golomb code with parameter m, we represent an integer $n > 0$ using two parameters q and r, where

$$q = \left\lfloor \frac{n}{m} \right\rfloor$$

  and

$$r = n - qm,$$

  - q is the quotient and r is the remainder when n is divided by m.
  - q can take on values $0,1,2,\ldots$ and is represented by the unary code of q.
  - r can take on values $0,1,2,\ldots,m-1$.
    - If m is a power of 2, we use the $\log_2 m$-bit binary representation of r.
    - If m is not a power of 2, we could still use $\lceil \log_2 m \rceil$ bits,
    - We can reduce the number of bits required if we use the $\lfloor \log_2 m \rfloor$-bit binary representation of r for the first $2^{\lceil \log_2 m \rceil} - m$ values and the $\lceil \log_2 m \rceil$-bit binary representation of $r + 2^{\lceil \log_2 m \rceil} - m$ for the rest of the values.

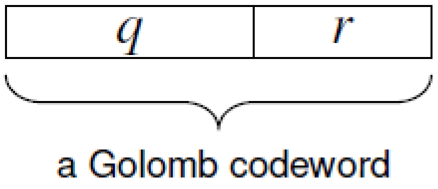
Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **<u>Golomb Codes</u>**
  - **<u>Example:</u>**
  - $m = 5, \lceil \log_2 5 \rceil = 3, \lfloor \log_2 5 \rfloor = 2, 2^{\lceil \log_2 5 \rceil} = 8$
  - So the first $8 - 5 = 3$ values of $r\ where\ (r = 0\ , 1\ , 2)$ will be represented by the 2 bit binary representation of r.
  - The next two values $(r = 3, 4)$ will be represented by the 3-bit representation of $r + 3$.
  - The quotient q is represented by the unary code for q.
  - So the codeword for 3 is 0110 and for 21 is 1111001.

Huffman Coding

- **Golomb Codes**
  - **Example:**



n	q	r	Codeword	n	q	r	Codeword
0	0	0	000	8	1	3	10110
1	0	1	001	9	1	4	10111
2	0	2	010	10	2	0	11000
3	0	3	0110	11	2	1	11001
4	0	4	0111	12	2	2	11010
5	1	0	1000	13	2	3	110110
6	1	1	1001	14	2	4	110111
7	1	2	1010	15	3	0	111000

Huffman Coding

- **<u>Golomb Codes</u>**

  <u>Example: Encoding Run Lengths using Golomb Codes</u>

  Message: 000000100000000010000000000010001001

  Golomb Code: 100110111

  17 bits → 9 bits

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **<u>Golomb Codes</u>**

<u>Optimality of Golomb Codes and Choosing "m":</u>

- Suppose that 0 has the probability p and 1 has probability 1-p
- Golomb code is optimal for the probability model:

$$P(n) = p^{n-1}q, \qquad q = 1 - p$$

when

$$m = \left\lceil -\frac{1}{\log_2 p} \right\rceil$$

For example, if p=127/128

$$m = \left\lceil -\frac{1}{\log_2(\frac{127}{128})} \right\rceil = 89$$

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **Golomb Codes**
  - Useful for binary compression when one symbol is much more likely than the other.

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **<u>Tunstall Codes</u>**
- All codewords are of equal length
- Each codeword represents a different number of letters
- The main advantages is that errors in codewords do not propagate
- Example: Alphabet $\mathcal{A} = \{A, B\}$

TABLE 3.18 A 2-bit Tunstall code.

Sequence	Codeword
AAA	00
AAB	01
AB	10
B	11

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **Tunstall Codes**

Example 1: Encode the sequence AAABAABAABAABAAA using Table 3.18.

- Coded String: 001101010100
- Entropy?
- Average Codeword Length using FLC?
- Average Codeword Length using Huffman?
- Average Codwowrd Length using Tunstall?

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **<u>Tunstall Codes</u>**

<u>Algorithm:</u>

- Tunstall Code: n-bit, Source: iid, Alphabet Size: N, Number of codewords: 2^n.
  - Start with the N letters of the source alphabet
  - Remove the entry in the codebook that has the highest probability
  - Add the N strings obtained by concatenating this letters with every letter in the alphabet (including itself).
- Repeat the above steps

Huffman Coding

- **Tunstall Codes**

Example 2:

- Tunstall Code: 3-bit, Source: iid, Alphabet: $\mathcal{A} = \{A, B, C\}$, $\mathrm{P}(A) = 0.6, P(B) = 0.3, P(C) = 0.1$.

TABLE 3.20 Source alphabet and associated probabilities.

Letter	Probability
A	0.60
B	0.30
C	0.10

TABLE 3.21 The codebook after one iteration.

Sequence	Probability
B	0.30
C	0.10
AA	0.36
AB	0.18
AC	0.06

TABLE 3.22 A 3-bit Tunstall code.

Sequence	Probability
B	000
C	001
AB	010
AC	011
AAA	100
AAB	101
AAC	110

Huffman Coding

- **Application of Huffman Coding**
  - **Lossless Image compression:**
  - Test Images

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **Application of Huffman Coding**
  - **Lossless Image compression:**

TABLE 3.23 Compression using Huffman codes on pixel values.

Image Name	Bits/Pixel	Total Size (bytes)	Compression Ratio
Sena	7.01	57,504	1.14
Sensin	7.49	61,430	1.07
Earth	4.94	40,534	1.62
Omaha	7.12	58,374	1.12

  - We get a reduction of about ½ to 1 bit/pixel.
  - If we are storing 1000s of images, 1 bit/pixel saves many megabytes
  - We can do even better by modelling the data first.

Huffman Coding

- **Application of Huffman Coding**
  - **Lossless Image compression:**
  - Using the model: $\hat{x}_n = x_{n-1}$

TABLE 3.24 Compression using Huffman codes on pixel difference values.

Image Name	Bits/Pixel	Total Size (bytes)	Compression Ratio
Sena	4.02	32,968	1.99
Sensin	4.70	38,541	1.70
Earth	4.13	33,880	1.93
Omaha	6.42	52,643	1.24

Huffman Coding

- **Application of Huffman Coding**
  - **Lossless Image compression:**
  - Using the model: $\hat{x}_n = x_{n-1}$
  - Using adaptive Huffman Coding

TABLE 3.25 Compression using adaptive Huffman codes on pixel difference values.

Image Name	Bits/Pixel	Total Size (bytes)	Compression Ratio
Sena	3.93	32,261	2.03
Sensin	4.63	37,896	1.73
Earth	4.82	39,504	1.66
Omaha	6.39	52,321	1.25

Huffman Coding

- **Application of Huffman Coding**
  - **Text compression:**

TABLE 3.26 Probabilities of occurrence of the letters in the English alphabet in the U.S. Constitution.

Letter	Probability	Letter	Probability
A	0.057305	N	0.056035
B	0.014876	O	0.058215
C	0.025775	P	0.021034
D	0.026811	Q	0.000973
E	0.112578	R	0.048819
F	0.022875	S	0.060289
G	0.009523	T	0.078085
H	0.042915	U	0.018474
I	0.053475	V	0.009882
J	0.002031	W	0.007576
K	0.001016	X	0.002264
L	0.031403	Y	0.011702
M	0.015892	Z	0.001502

TABLE 3.27 Probabilities of occurrence of the letters in the English alphabet in this chapter.

Letter	Probability	Letter	Probability
A	0.049855	N	0.048039
B	0.016100	O	0.050642
C	0.025835	P	0.015007
D	0.030232	Q	0.001509
E	0.097434	R	0.040492
F	0.019754	S	0.042657
G	0.012053	T	0.061142
H	0.035723	U	0.015794
I	0.048783	V	0.004988
J	0.000394	W	0.012207
K	0.002450	X	0.003413
L	0.025835	Y	0.008466
M	0.016494	Z	0.001050

- For Chapter 3, the file size dropped from about 70,000 bytes to about 43,000 bytes with Huffman coding.

Huffman Coding

- **Application of Huffman Coding**
  - **Audio compression:**

TABLE 3.28 Huffman coding of 16-bit CD-quality audio.

File Name	Original File Size (bytes)	Entropy (bits)	Estimated Compressed File Size (bytes)	Compression Ratio
Mozart	939,862	12.8	725,420	1.30
Cohn	402,442	13.8	349,300	1.15
Mir	884,020	13.7	759,540	1.16

TABLE 3.29 Huffman coding of differences of 16-bit CD-quality audio.

File Name	Original File Size (bytes)	Entropy of Differences (bits)	Estimated Compressed File Size (bytes)	Compression Ratio
Mozart	939,862	9.7	569,792	1.65
Cohn	402,442	10.4	261,590	1.54
Mir	884,020	10.9	602,240	1.47

References

[1] N. Faller. An Adaptive System for Data Compression. In *Record of the 7th Asilomar Conference on Circuits, Systems, and Computers*, pages 593–597. IEEE, 1973.

[2] R.G. Gallager. Variations on a theme by Huffman. *IEEE Transactions on Information Theory*, IT-24(6):668–674, November 1978.

[3] D.E. Knuth. Dynamic Huffman coding. *Journal of Algorithms*, 6:163–180, 1985.

[4] J.S. Vitter. Design and analysis of dynamic Huffman codes. *Journal of ACM*, 34(4):825–845, October 1987.