# Encoding Algorithm and Its Application in Image Compression

**Team Members:**
Shah Miten (180001049)
Sundesh Gupta (180001059)

# Citation:

**Author:** Erdal, Erdal & Erguzen, Atilla. (February 2019).

**Topic:** An Efficient Encoding Algorithm Using Local Path on Huffman Encoding Algorithm for Compression.

**Journal:** Applied Sciences. 9. 782. 10.3390/app9040782.

# Goals

1. Implement Huffman encoding and arithmetic coding algorithms.

2. Analyze Huffman encoding and arithmetic coding for image compression.

3. Implement and Analyze improved Huffman encoding for image compression.

4. Test the performance of the improved huffman encoding algorithm on different sets of pictures including grayscale and colored images and compare with existing algorithms.

5. Further optimize Huffman encoding for lesser time complexity and higher compression ratio, if possible.

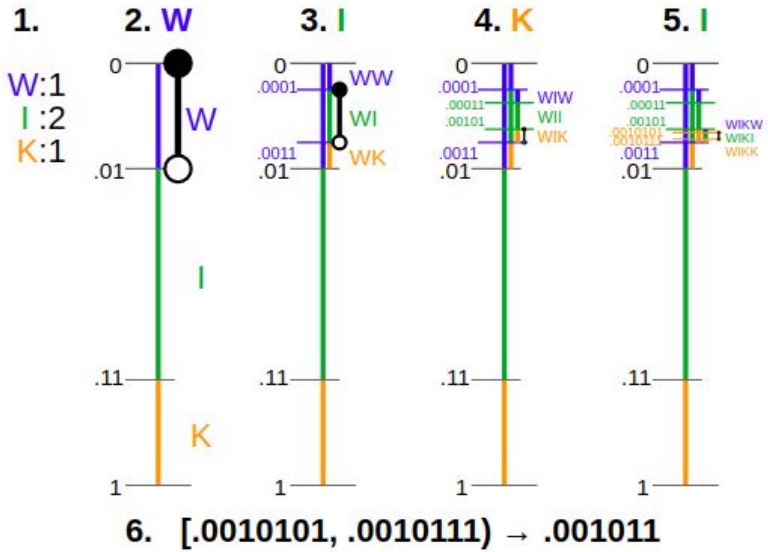
# Background

Huffman Coding: Lossless, prefix codes, but In cases where a small number of symbols are used and there are small differences in the probability/frequency of characters, the efficiency of Huffman encoding is reduced

Arithmetic Coding: Address deficiencies of Huffman coding but, difficult to implement, slower execution, no prefix codes generated

Improved Huffman Coding: All the benefits of Huffman, better compression even if probability/frequency of characters are comparable

# Arithmetic Coding




*Step 1. The letter frequencies are found.*

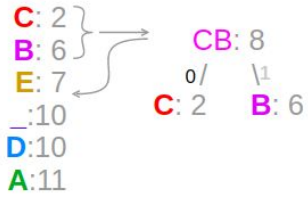
*Step 2. The interval [0, 1) is partitioned in the ratio of the frequencies.*

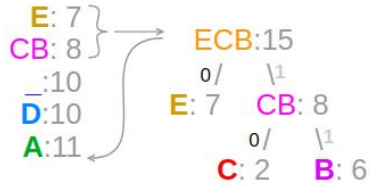
*Steps 3–5. The corresponding interval is iteratively partitioned for each letter in the message.*

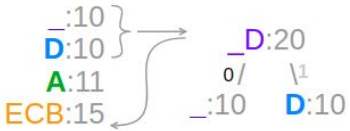
*Step 6. Any value in the final interval is chosen to represent the message.*

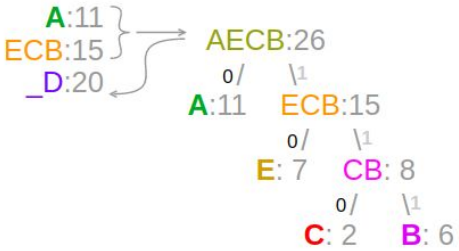
# Huffman Coding Algorithm

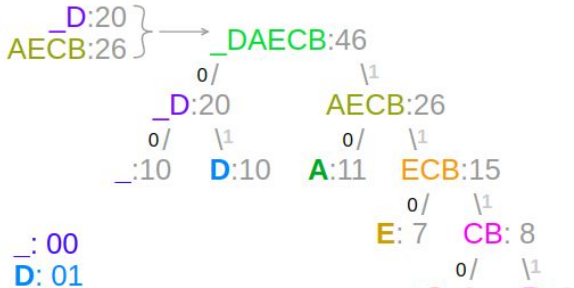
1. "A_DEAD_DAD_CEDED_A_BAD_BABE_A_BEADED_ABACA_BED"

2. 


3. 


4. 

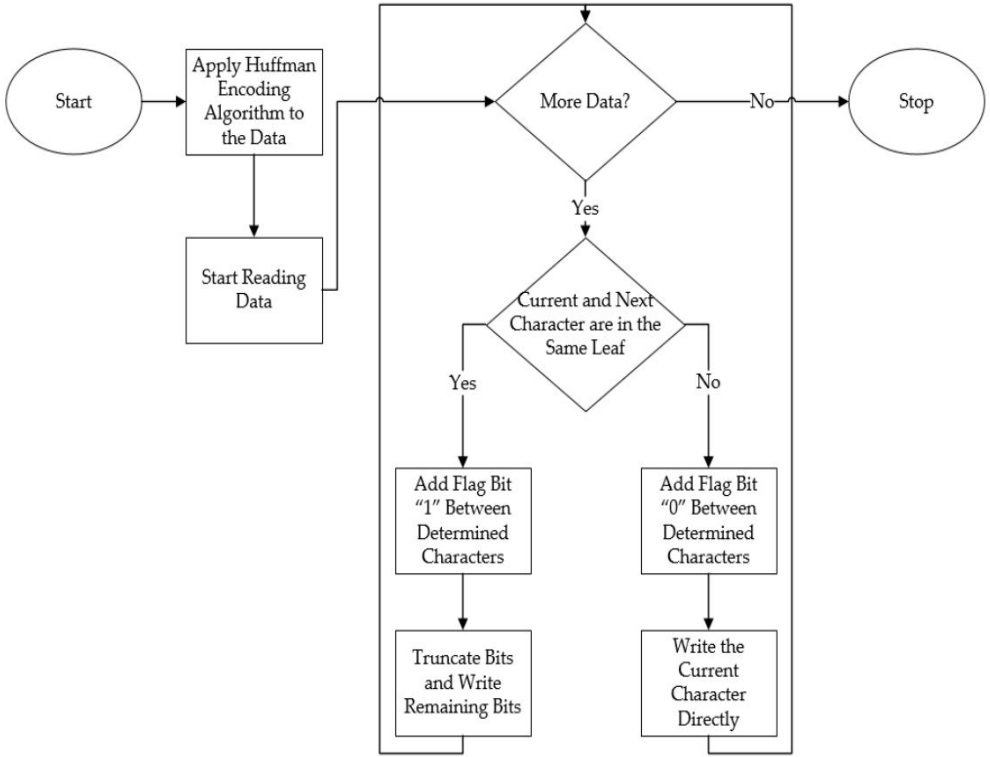

5. 


6. 

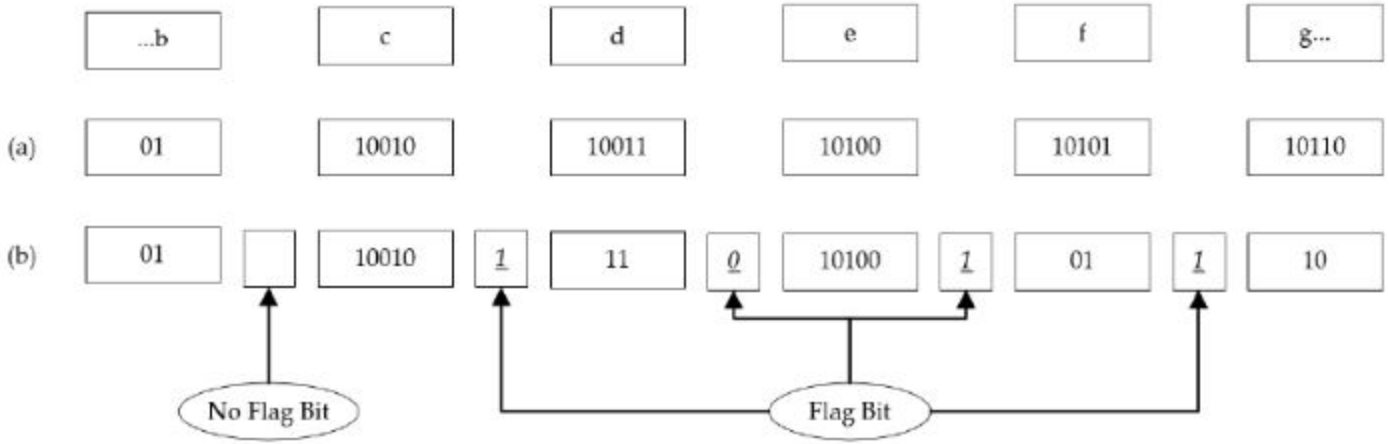

7. _: 00
D: 01
A: 10
E: 110
C: 1110
B: 1111

8. "100001110100100011001001110110011100100100011111001001111101111100010001111110100111001001010111110111010001111110 01"

# Improved Huffman Algorithm based on local Path

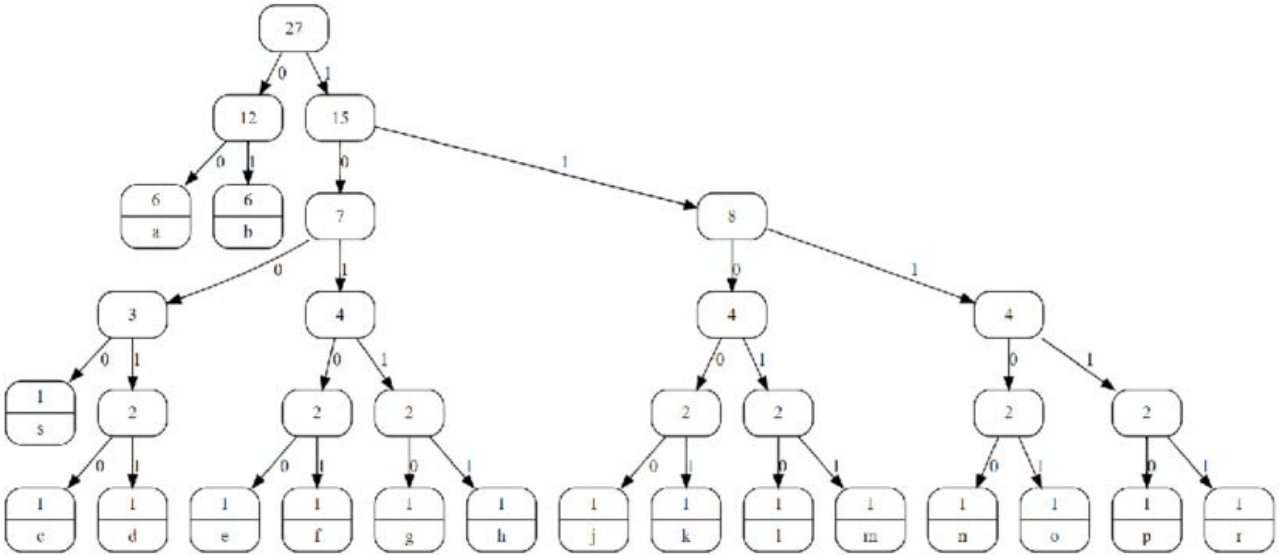


# Improved Huffman Algorithm based on local Path



# Improved Huffman Algorithm based on local Path

Message to encode: *aaaabbbbcdefghjklmnoprsaabb*




98 bits: 00000000010101011001010011101001010110110101111100011001110101101111100111011111011111100000000101
83 bits: 00000000010101011001011101010010111011101100010111011101110010111011101000000000101

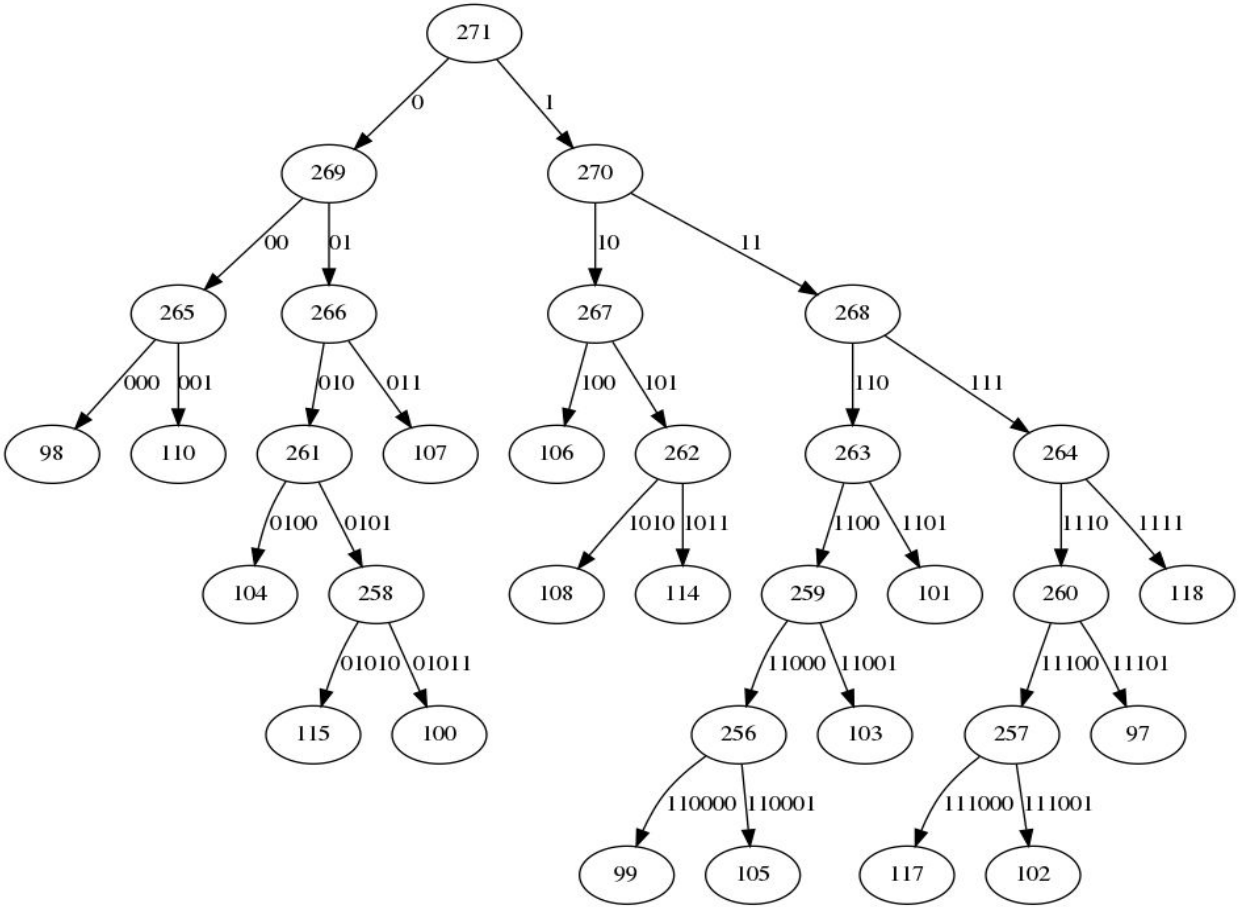
# Time and Space Complexity

| | | Time Complexity | Space Complexity |
|---|---|---|---|
| Huffman Coding | Encoding | O(P*lgP + n*lgP) ~ O(n*lgP) | O(n + P) ~ O(n) |
| | Decoding | O(n*lgP) | O(n) |
| Arithmetic Coding (In Progress) | Encoding | | |
| | Decoding | | |
| Improved Huffman Coding | Encoding | O(n*lgP) | O(n) |
| | Decoding | O(n*lgP) | O(n) |

Here, n = Number of pixels
P = Number of distinct values / characters.

# Implementation

Huffman Tree generated using our implementation of Huffman Coding Algorithm




Message to be encoded:

blvbjeanvnkjvnjlearknjnakrn
ekndcjbhbllivuekrjhggefskvh
fbvkjdbsrjgk

# Image Compression Results (as per paper)

| Image | Type | Size | Huffman Encoding Algorithm | | Arithmetic Coding Algorithm | | Proposed Algorithm | |
|---|---|---|---|---|---|---|---|---|
| | | | NoBPP | CP | NoBPP | CP | NoBPP | CP |
| Image 1 | Color | 1360 × 1024 | 7.10 | 11.19 | 7.08 | 11.50 | 7.03 | 12.08 |
| Image 2 | Color | 1360 × 1024 | 6.99 | 12.61 | 6.96 | 12.95 | 6.96 | 12.96 |
| Image 3 | Color | 1360 × 1024 | 6.80 | 14.98 | 6.78 | 15.25 | 6.76 | 15.48 |
| Image 4 | Color | 2160 × 1440 | 6.20 | 22.48 | 6.17 | 22.94 | 5.61 | 29.84 |
| Image 5 | Color | 2160 × 1440 | 6.08 | 23.95 | 6.06 | 24.27 | 5.38 | 32.81 |
| Image 6 | Color | 3584 × 2438 | 6.03 | 24.56 | 6.01 | 24.82 | 5.76 | 27.99 |
| Image 7 | Color | 3584 × 2438 | 5.76 | 28.03 | 5.73 | 28.42 | 5.49 | 31.41 |
| Image 8 | Color | 3584 × 2438 | 6.45 | 19.41 | 6.42 | 19.77 | 6.18 | 22.72 |

# Image Compression Results (ours)

| | | Huffman Coding | | | | Improved Huffman Coding | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Image Size** | **Type** | **CP** | **NoBPP** | **Comp. Time** | **Decomp. Time** | **CP.1** | **NoBPP.1** | **Comp. Time.1** | **Decomp. Time.1** |
| 112x131 | Color | 5.8206 | 7.5344 | 0.0046 | 0.0258 | 6.8083 | 7.4553 | 0.0085 | 0.0259 |
| 199x341 | Grey | 7.4638 | 7.4029 | 0.0104 | 0.0385 | 13.3136 | 6.9349 | 0.0155 | 0.0427 |
| 363x374 | Color | 7.4923 | 7.4006 | 0.0505 | 0.2085 | 12.0792 | 7.0337 | 0.0694 | 0.2153 |
| 403x456 | Grey | 6.392 | 7.4886 | 0.0247 | 0.111 | 6.1155 | 7.5108 | 0.043 | 0.1153 |
| 516x688 | Grey | 4.6612 | 7.6271 | 0.0538 | 0.2137 | 9.2226 | 7.2622 | 0.0769 | 0.2164 |
| 704x710 | Color | 4.0409 | 7.6767 | 0.1521 | 0.8812 | 4.6192 | 7.6305 | 0.2556 | 0.903 |
| 1001x1419 | Grey | 3.0224 | 7.7582 | 0.4044 | 0.8296 | 5.7363 | 7.5411 | 0.3213 | 0.8452 |

# TO DO:

- Implement Arithmetic Coding.
- Study more about different Encoding Algorithms, like LZM Algorithm.
- Try to optimize current Algorithms, and also increase compression ratio.
- Implement a Command Line Interface (CLI).