Programming 3 – NVC

Huffman Coding Algorithm Essay

Jonathan Garcia

December 3, 2023

Huffman Coding

Huffman Coding is an algorithm that is designed for lossless data compression. The word compression is described as “the process or result of becoming smaller or pressed together”. *(Vocabulary.com)* This algorithm is still used today and is commonly used to compress multimedia formats like JPEG, PNG image formats and MP3 audio. It is also used with the ZIP file format that allows users to compress data for faster and easier transfer. PKZIP is an archiving tool that uses the ZIP file format. Lossless data compression is significant because it allows for a faster and exceptional way to transfer data without compromising the quality of the data that is compressed. Compression also allows for less use of bandwidth over an internet connection and less space taken up on a storage device like a disc or drive.

The algorithm was developed in the 1952 by David A. Huffman for his final exam grade while he attended MIT. (*Huffman, D. A. (2023, December 3). David A. Huffman. Wikipedia)* “The assignment was to find an efficient way to represent the English alphabet using binary codes. While other methods of encoding existed, Huffman aimed to create a variable-length coding scheme that minimized the average length of the codes, taking into account the frequency of each character.” *(ChatGPT)*

Standard ASCII characters (American Standard Code for Information Interchange) uses 8 bits each. “This includes the English alphabet, upper- and lower-case letters, numbers 0 through 9 and punctuation symbols.” *(Peter Loshin, (2023, December 3). ASCII. Tech Target)* The Huffman coding algorithm compresses each character below 8 bits using the variable-length coding scheme that Huffman came up with. This scheme is the process of assigning shorter code schemes to the more frequent characters and longer code schemes to less frequent characters. This results in a lower bit count, high efficiency and lossless data compression.

A Binary tree can be used to determine the Huffman coding for a given value in the data. The process starts first by determining the frequency of each value. Each value is a leaf node. The leaf nodes make up a binary tree. The frequency of each character and the character is stored in each node. The algorithm then merges the pair of nodes that make up the lowest sum into a new node. It will continue on from the lowest to the highest pair of nodes until only 1 node exists. Finally, the encoding can begin and is obtained by traversing from the top node, down to each leaf. Going left down the tree will append a 0 and going right down the tree will append a 1. Traversing the binary tree allows for the prefix property, “which states that no bit-sequence encoding of a character is the prefix of the bit-sequence encoding of any other character.” *(Engineering, (2023, December 3) Purdue)* This avoids ambiguity when decoding the compressed data, allowing for high efficiency.

The Time complexity for Huffman coding is O(nlogn) where n is the number of unique characters. “Log-linear time complexity: The algorithm requires a number of operations that grows as the product of the input size and the logarithm of the input size.” *(Growth of functions and complexity)* This time complexity tends to have a low to mid-level time complexity which allows for a more efficient process with larger input sizes.

The Space complexity for Huffman coding is O(n), where "n" is the number of symbols in the input data. It tends to be very efficient and beneficial especially when characters appear in the data very frequently because they will be assigned a shorter code scheme. It would not be as effective if all characters only appeared once but would still be more efficient than uncompressed data.

Alongside multimedia, Huffman coding is also used to compress text files, transmit fax and usually found in the backend of other compression algorithms. Huffman coding can be found in many fields of study and work that deal with sharing files. These range from web and software development, photography, music production, digital art, records and archival work and many others.

Code Example (Kevin Roark, (2023 December 3) HuffmanDemo); In this code example I have modified it slightly to show the Huffman coding at work for the text, “Jack and Jill went up the hill.”

#include <iostream>

#include <queue>

#include <unordered\_map>

using namespace std;

// Leaf Nodes

struct Node {

char data; // Character data

int frequency; // Frequency of the character

Node\* left, \* right; // Left and right child pointers

};

struct Compare {

// Comparator for priority queue, used to build the Huffman tree

bool operator()(Node\* left, Node\* right) {

return (left->frequency > right->frequency);

}

};

unordered\_map<char, string> huffmanCodes; // Map to store the Huffman codes

// Function to encode the characters using the Huffman tree

void encode(Node\* root, string str);

// Function to decode a single character from the encoded string

void decode(Node\* root, int& index, string str);

// Function to build the Huffman tree based on the input text

Node\* buildHuffmanTree(string text);

// Function to encode the entire input text

string encodeText(string text);

// Function to decode the entire encoded text

void decodeText(Node\* root, string encoded);

int main()

{

string text = "Jack and Jill went up the hill."; // Text that will be compressed

Node\* root = buildHuffmanTree(text); // Build Huffman tree

string encodedText = encodeText(text); // Encode the text

cout << "Encoded text: " << encodedText << endl;

cout << endl;

return 0;

}

// Function to encode the characters using the Huffman tree

void encode(Node\* root, string str) {

if (root == nullptr) {

return;

}

if (!root->left && !root->right) {

huffmanCodes[root->data] = str; // Assign code for leaf node

}

encode(root->left, str + "0"); // Left child corresponds to '0'

encode(root->right, str + "1"); // Right child corresponds to '1'

}

// Function to build the Huffman tree based on the input text

Node\* buildHuffmanTree(string text) {

unordered\_map<char, int> freq;

for (char c : text) {

freq[c]++; // Calculate frequency of each character

}

priority\_queue<Node\*, vector<Node\*>, Compare> pq;

for (auto pair : freq) {

Node\* temp = new Node;

temp->data = pair.first;

temp->frequency = pair.second;

temp->left = nullptr;

temp->right = nullptr;

pq.push(temp); // Push nodes into the priority queue

}

while (pq.size() > 1) {

Node\* left = pq.top(); pq.pop();

Node\* right = pq.top(); pq.pop();

Node\* temp = new Node;

temp->data = '$'; // Internal node

temp->frequency = left->frequency + right->frequency;

temp->left = left;

temp->right = right;

pq.push(temp); // Push combined node back into the queue

}

Node\* root = pq.top();

encode(root, ""); // Generate Huffman codes

return root; // Return the root of the Huffman tree

}

// Function to encode the entire input text

string encodeText(string text) {

string res = "";

for (char c : text) {

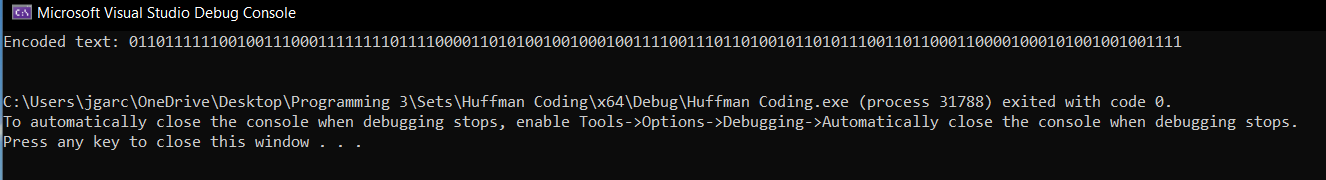
res += huffmanCodes[c]; // Concatenate Huffman codes

}

return res;

}

Huffman coding result:



Huffman coding is still a highly efficient way to compress many data types including multimedia and text files. The use of value-based coding, priority queue and binary tree all lend to its efficiency in lossless compression. Despite new algorithms, it’s still used and preferred in certain cases even 70 years after it was created. Compression techniques are becoming more important than ever with the growing amount of digital data. With this in mind, it seems like Huffman coding will continue to find a place in many fields and grow in importance as the future is more and more digitized.

Sources

1. [David A. Huffman - Wikipedia](https://en.wikipedia.org/wiki/David_A._Huffman)
2. [Compression - Definition, Meaning & Synonyms | Vocabulary.com](https://www.vocabulary.com/dictionary/compression)
3. Author: Huffman, David A. Title: "A Method for the Construction of Minimum-Redundancy Codes" Journal: Proceedings of the I.R.E. (Institute of Radio Engineers) Year: 1952
4. Author: Loshin, Peter: “ASCII (American Standard Code for Information Interchange)” [What is ASCII (American Standard Code for Information Interchange)? (techtarget.com)](https://www.techtarget.com/whatis/definition/ASCII-American-Standard-Code-for-Information-Interchange#:~:text=ASCII%20encoding%20is%20based%20on,9%20and%20basic%20punctuation%20symbols.)
5. *(Huffman Coding: “Engineering (Purdue)“* [Huffman Coding (purdue.edu)](https://engineering.purdue.edu/ece264/17au/hw/HW13?alt=huffman)
6. NVC Computer Science Department “Growth of functions and complexity” Year 2023 [Growth of functions and complexity - Programming 3 - Data Structures - NVC Computer Science Department (atlassian.net)](https://dr-roark.atlassian.net/wiki/spaces/P3DS/pages/1377599606/Growth+of+functions+and+complexity)
7. Author: Roark, Kevin. Title: “HuffmanDemo” (NVC Computer Science Department) Year 2023 [Huffman Compression - Programming 3 - Data Structures - NVC Computer Science Department (atlassian.net)](https://dr-roark.atlassian.net/wiki/spaces/P3DS/pages/1376158520/Huffman+Compression)