

Programming Assignment Report

U10P32002, School of Computer Science and Technology, NPU, Spring 2019

Written by Dikshya Kafle

2018380039

Programming Assignment 8 Due date: 3rd June 2019

Huffman Encoder & Decoder

□ Vital information:

U10P32002, School of Computer Science and Technology, NPU, Spring 2019

Written by Dikshya Kafle

2018380039

Programming Assignment 8 Due date: 3rd June 2019

□ Problem Statement:

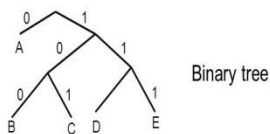
The purpose of this assessment is to design a Huffman encoding and Huffman decoding system, which is capable of encoding and decoding the messages that are to be transmitted. Therefore, in the process of building a Huffman tree it is necessary for a programmer to implement a little weight to the left and large weight to the right. Thus, the left child tree is to be encoded as 0 whereas the right child is to be encoded as 1. In doing so the program should follow the encoding prefix nature.

□ Structure Chart:

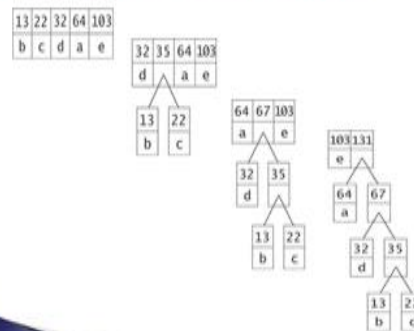
Huffman Code: Decoding

A - 0
B - 100
C - 101
D - 110
E - 111

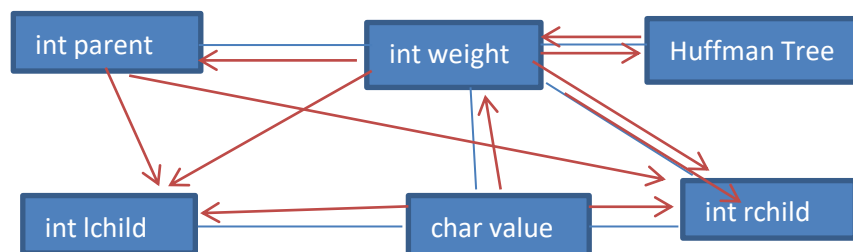
Bitstream: 10001000101010110111 (22 bits)
Codes: 100 0 100 0 101 0 101 0 110 111
Message: B A B A C A C A D E



Huffman encoding example



v



Programming Assignment Report

U10P32002, School of Computer Science and Technology, NPU, Spring 2019

Written by Dikshya Kafle

2018380039

Programming Assignment 8 Due date: 3rd June 2019

□ Implementation

Huffman coding assigns **codes** to characters such that the length of the **code** depends on the relative frequency or weight of the corresponding character. Thus, it is a variable-length **code**.

A Huffman coding library and command line interface to the library. The encoder is a 2 pass encoder. The first pass generates a Huffman tree and the second pass encodes the data. The decoder is one pass and uses a Huffman code table at the beginning of the compressed file to decode the data.

Huffman.cpp file includes different library data's (functions declaration) as it includes files like iostream, cstdio, cstring, and algorithm. In addition on that, #define directive are being used; this allows the definition of macros within the source code. These macro definitions allow constant values to be declared for use throughout the code.

Two different types of data types i.e. int and char are being used in the program. In the c programming int represent integers while char represent characters.

Name and types of parameters: HNodeType HuffNode[], n- int, str[]- char, HNodeType hufTree[]

Huffman Encoding:

There are two major parts of encoding:

1. Building a Huffman Tree from input characters.
2. Traverse the Huffman Tree and assign code to each character

Input: N unique characters along with their frequencies of occurrence. Enter a positive integer n representing the size of the character set ($n \leq 100$) as well as n characters and n weights (weight is a positive integer. It has a larger value, then it has the greater probability of occurrence); the user need to enter messages whose length is smaller or equal to 100.

Firstly, create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap.) Secondly, extract two nodes with the minimum frequency from min heap. Thirdly, create a new internal node with a frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and other extracted node as its right child. Add this node to min heap. Repeat second step and third step until the heap contains only one node. The remaining node is the root node and the tree is completed.

Output: A Huffman tree.

Programming Assignment Report

U10P32002, School of Computer Science and Technology, NPU, Spring 2019

Written by Dikshya Kafle

2018380039

Programming Assignment 8 Due date: 3rd June 2019

For Huffman Decoding:

Input: To decode the encoded data the Huffman tree is required. Therefore the iteration is done through the binary encoded data. In order to find character corresponding to current bits, the following simple steps are being used.

1. Initiate from the root and do following until a leaf is found.
2. If current bit is 0, it is moved to left node of the tree.
3. If the bit is 1 it is moved to the right node of the tree.
4. If during traversal a leaf node is encountered, the character of that particular leaf node is printed and then again iteration of the encoded data starting from step 1 is continued.

prefix code is a type of **code** that let programmer to decode the encoded text without special marker. for example, the map {a=0, b=10, c=11} is a **prefix code** as no marker is needed for decryption of any string.

□ Test Description and Results:

The following is the result of the assessing infection program. After understanding the human code documents a lot of research was done and finally the Huffman Encode and Decode were written. After accomplishing writing the correct code the code was being runned in Dev C++.

Afterwards the input is made by entering the following:

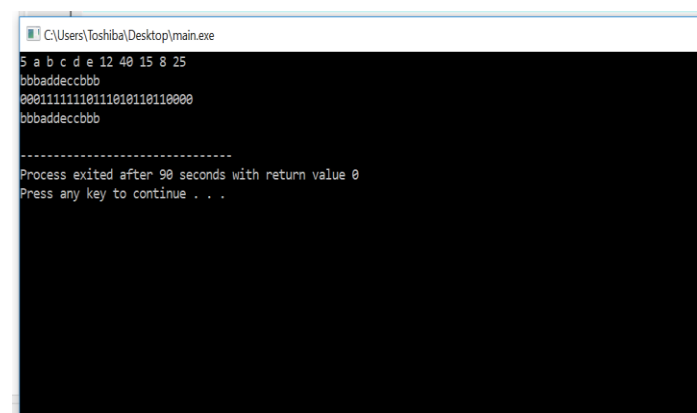
5 a b c d e 12 40 15 8 25

bbbaddeccbbb

And the displayed output is: 00011111110111010110110000

bbbaddeccbbb

The following is the result of the program after running the code:



```
C:\Users\Toshiba\Desktop\main.exe
5 a b c d e 12 40 15 8 25
bbbaddeccbbb
00011111110111010110110000
bbbaddeccbbb

-----
Process exited after 90 seconds with return value 0
Press any key to continue . . .
```

Programming Assignment Report

U10P32002, School of Computer Science and Technology, NPU, Spring 2019

Written by Dikshya Kafle

2018380039

Programming Assignment 8 Due date: 3rd June 2019

□ Epilogue:

The time complexity of the Huffman algorithm is $O(n \log n)$. Using a heap to store the **weight** of each tree, each iteration requires $O(\log n)$ time to determine the cheapest **weight** and insert the new **weight**. There are $O(n)$ iterations, one for each item.

While doing the program I had many bugs which had influence on program's working and many troubles with compilation programs, because of syntax. It was very difficult for me to understand what errors I made. Some minor errors like after the completion of the for loop I forgot to close it with the braces. Some conditions did not really meet requirements and I had to do various researches in the internet and corrected the codes which lead to eliminate bugs. The following picture is the corrected section of the code:

```
for(a=0;a<n;a++)
{
    cd.start=n-1;

    int cur=a;
    int p=HuffNode[cur].parent;

    while(p!=-1)
    {
```

After correcting all the minor and major bugs in the codes I finally manage to get the codes right and finally, got the final results. This program is too time consuming and I found that this topic is a very tough one yet very interesting one. It was kind of confusing to understand and explain well. However, I managed to do deal all these bugs with the help of a lot of You Tube videos, blogs and websites. This program helped me understand how the Huffman coding works and how the Huffman decoding is being done. C language is being used in doing this project. Although the coding was tuff and hard to understand the general concept of how Huffman tree works is very interesting.

□ **Attachments:** The file Huffman.cpp is being attached in the email. And this is the report file.

□ **Acknowledgement:** In the processes of solving this assignment I looked for a lot of information in the internet. I watched several videos related to Huffman tree, encoding Huffman tree and decoding Huffman tree and took few notes in the process of understanding. I would like to thank You Tube and a lot of coding related websites. These resources helped me a lot to understand what I have done in this report. Throughout the coding and researching process I came to know that Huffman coding is a method of **data compression** that is independent of the data type, that is, the data could represent an image, audio or spreadsheet. This compression scheme is used in JPEG and MPEG-2. Huffman coding works by looking at the data stream that makes up the file to be compressed.