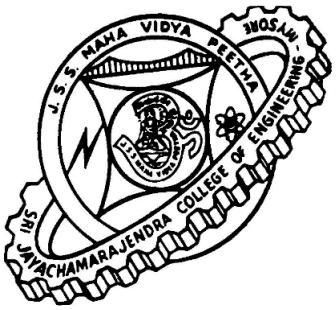
VISVESWARAYA TECHNOLOGICAL UNIVERSITY

BELGAUM

SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
MYSORE-570006

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Project on

MINIMUM VARIANCE HUFFMAN CODING

Under the Guidance of
Smt.Trishiladevi C Nagavi
Senior Scale Lecturer
Dept of CS&E,SJCE Mysore.

Team:

NAME	ROLL NO	USN
SHARAD D	03	4JC06CS089
VIKRAM TV	61	4JC07CS120
PRAVEEN KUMAR	39	4JC07CS073
RAVISHANKAR MU	47	4JC07CS090

Contents

1 Introduction

The Huffman Coding technique was developed by David Huffman as a part of class assignment. The codes generated using this technique are called Huffman Codes. These codes are prefix codes and are optimum for a given model.

The Huffman procedure is based on two observations regarding optimum prefix codes.

1. In an optimum code,symbols that occur more frequently(have a higher probability of occurence)will have shorter codewords than symbols that occur less frequently.

2. In an optimum code,the two symbols that occur least frequently will have the same length.

2 Requirements

2.1 Input Requirements

The input to the program should be given through a file. The input is a large text files which contains many repeated and non repeated sequence of characters or sentences.

2.2 Output Requirements

The output of the minimum variance huffman coding is a compressed text file which requires less number of bits to represent them.

2.3 Functional Requirements

The program should accept a large text file consisting of many repeated and non repeated sequence of characters or sentences. It should be effectively able to compress the text file.

2.4 Nonfunctional Requirements

1. Easy to use User interface.

2. Efficiency in analyzing.

3. Versatility in handling all types of text files.

4. Output suited to be provided for other tool as input.

2.5 Hardware Requirements

Minimum Requirements:

1. 1.1 GHz processor
2. 256 MB RAM or Higher
3. 4GB hard disk

2.6 Software Requirements

1. Linux OS
2. Text Editor
3. GNU C compiler (gcc)
4. make
5. dotty

3 Design and Implementation

Minimum Variance Huffman Code Design
Procedure:

1. Sort the probability of all source symbols in a descending order.
2. Merge the last two into a new symbol, add their probabilities.
3. Put the combined symbol as high as possible in the sorted list
4. Repeat Step 1, 2,3 until only one symbol (the root) is left.
5. **Code assignment:** Traverse the tree from the root to each leaf node, assign 0 to the left branch and 1 to the right branch.
6. Compute average codeword length

3.1 Compression

The frequency of each character is counted. Using the frequency count, a Minimum Variance Huffmann tree is constructed. A stack is used that contains the frequency of all ascii characters. An insertion sort is performed on the stack to arrange all the frequencies in non-decreasing order. The least two frequency count is got from the stack and a binary tree is constructed. The stack contents are decreased. The stack is then inserted with the new value of the combined binary tree and reordered. The construction of the least two frequencies' binary tree is continued until we are left with one node. The binary tree that gets constructed is attached to its parent.

A dictionary containing the codewords for each character is generated by traversing the minimum variance huffmann tree. On traversing to the left child, we assign a bit value '0' and on traversing to the right we assign a bit value 1. The dictionary contains both the character and its appropriate codeword. Then the dictionary is printed onto a file.

Compression is now performed by referring to the dictionary. Each character in the input file is read and assigned with a corresponding codeword. The conversion from bytes to bits is necessary as storing in bytes would expand the input file. Hence all byte values in compressed file are converted to bit values and stored. In case of last bit values, zeroes are padded. Finally, the compression ratio is calculated. Both the compressed file and the dictionary is sent.

3.2 Decompression

In order to decompress a file, the dictionary contents are read and written into a dictionary structure that contains both the character and the corresponding codeword. The compressed file is read bit by bit and matched with the codewords. On a successful match, we decode the bits into the corresponding character and write to the target file.

4 Screen Shots

These are the screen shots:

.

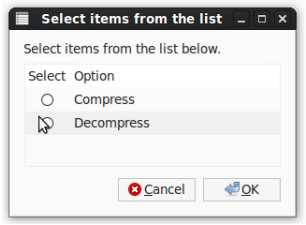


Figure 1: Select Compression or Decompression

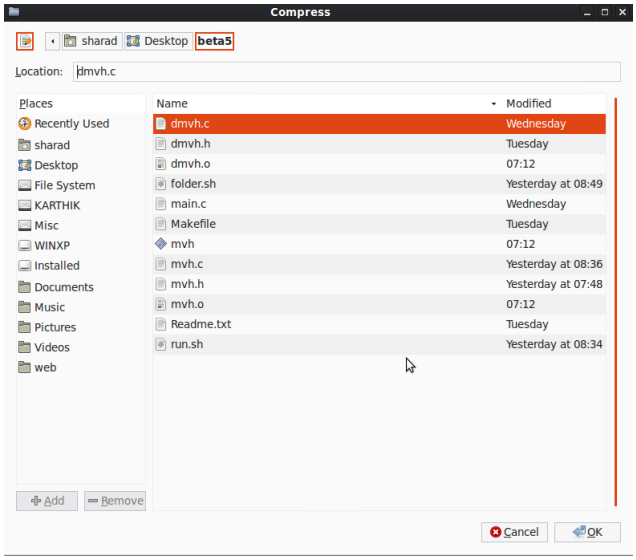


Figure 2: Select Compression Source

.

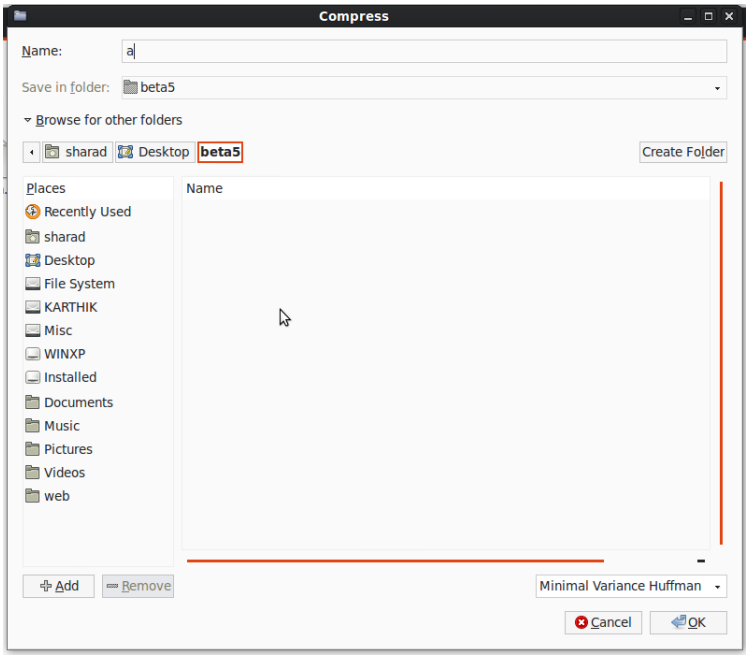


Figure 3: Select Compression Destination

.

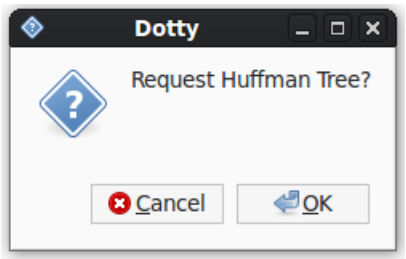


Figure 4: Request for Huffmann tree

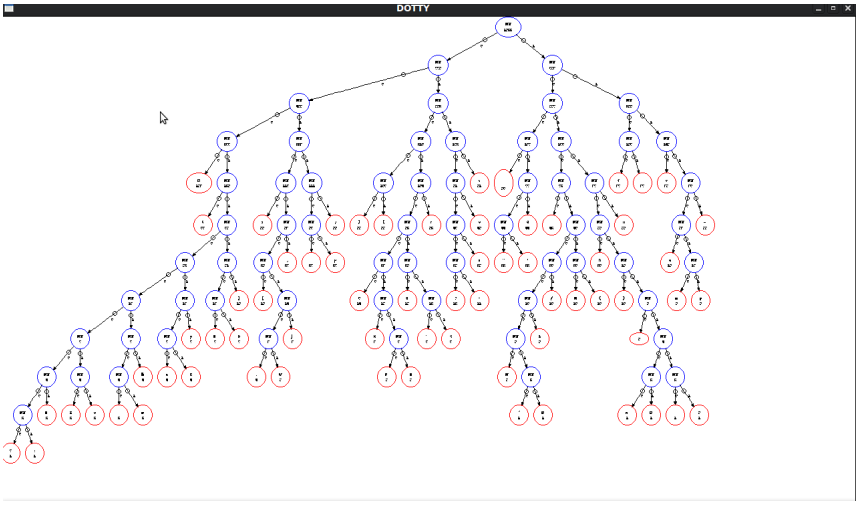


Figure 5: Dotty Display

Figure 6: Show Compression Ratio

.

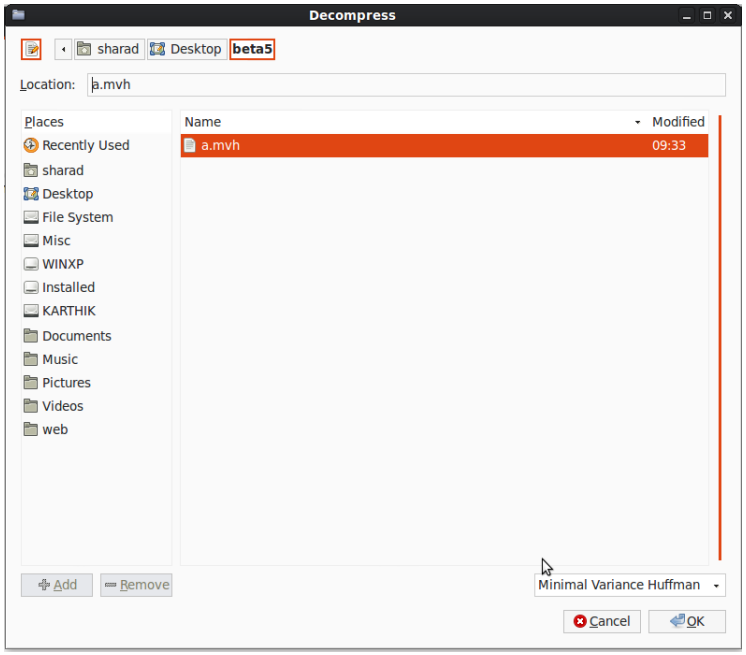


Figure 7: Select Decompression Source

.

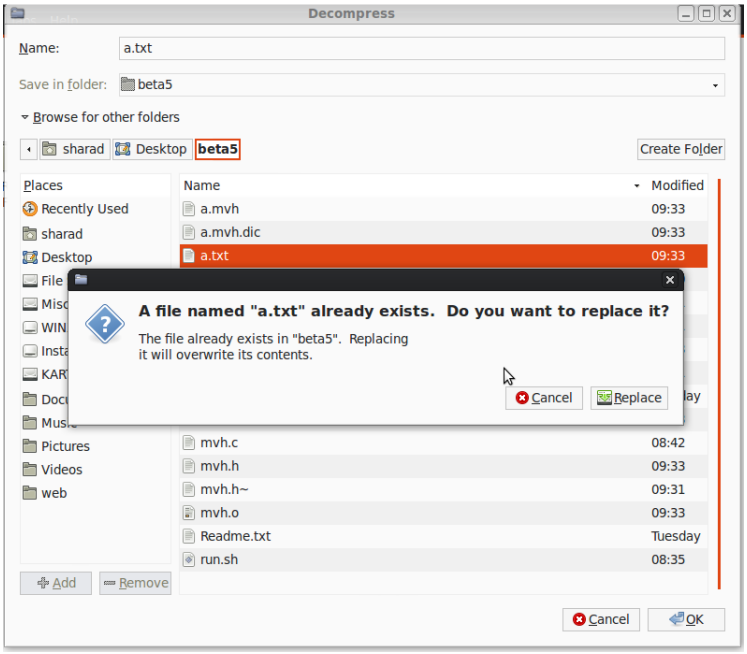


Figure 8: Select Decompression Destination

```
sharad@sharad-laptop:~/Desktop/beta5$ diff a.txt mvh.h
sharad@sharad-laptop:~/Desktop/beta5$
```

Figure 9: Difference between Original and Decompressed Files

5 System Testing

File	Original Size	Compressed Size	Ratio
Mvh.h	4783	2182	33 %
Fulla.txt	271	34	87 %
Folder	876	570	34 %
Image19.jpg	40700	40648	0%
cs.pdf	16265	15973	1%
Barney.bmp	1000054	834136	16 %

6 Application

Arithmetic Coding can be viewed as a generalization of Huffman coding; indeed, in practice arithmetic coding is often preceded by Huffman coding, as

it is easier to find an arithmetic code for a binary input than for a nonbinary input. Also, although arithmetic coding offers better compression performance than Huffman coding, Huffman coding is still in wide use because of its simplicity, high speed and lack of encumbrance by patents.

Huffman coding today is often used as a "back-end" to some other compression method.DEFLATE(PKZIP's algorithm) and multimedia codecs such as JPEG and MP3 have a front-end model and quantisation followed by Huffman coding.

7 Future Extensions

Whenever we transfer a text file from the computer to the pendrive,the file should be compressed,and if we click on the compressed file which is present in the pendrive,it should be decompressed to show the original file.

8 Conclusion

Minimum variance Huffman Coding is an efficient algorithm for compressing the text files.If the text file contains many repeated sequence of characters or words then a greater compresion is achieved. The compression that was written for a general text file can be modified for a binary file too, so that compression can be done for image files, pdfs, although the compression was not that good enough compared to a text file. Finally minimal variance huffmann tree provides a good compression ratio.

9 References

- Khalid Sayood: Introduction to Data Compression,3rd Edition,Elsevier,2006
- Wikipaedia : http://en.wikipedia.org/wiki/Huffman_coding