A Mini Project Report

on

Personal Voice Assistant

In Subject: **Design Thinking Process-I**

by

STUDENT NAME	ROLL NO	PRN NO
GANESH BHAVAR	371018	22010331
HARSHIT KUMAR SINGH	371021	22011072
ISHWAR ZATKE	371022	22010600
VIDHYAPATI SHELKE	371069	22120037

Department of Artificial Intelligence and Data Science

VIIT

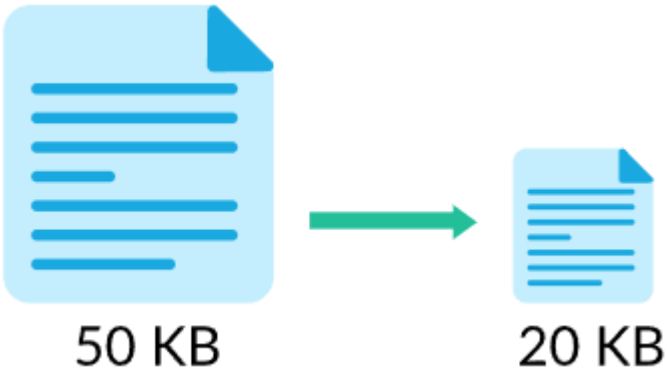
2021-2022

Contents

Abstract:

Nowadays, Modern computers can store increasingly large numbers of files, but file size still matters. The smaller our files are, the more files we can store.

We use compression algorithms to reduce the amount of space needed to represent a file.



File compression is a method of data compression that reduces the logical size of files to save disk space for easier and faster transmission over the network or Internet. It can create one or more versions of files with the same data that are much smaller than the original file.

By compressing a file, data takes up less space, and files can be sent and received a lot more quickly.

CHAPTER 1 – INTRODUCTION

Our project aims to make a program which can compress a file for us using Huffman Coding.

Huffman coding is a lossless way to compress and encode text based on the frequency of the characters in the text. Huffman code is a special type of optimal prefix code that is often used for lossless data compression.

The idea is to assign variable-length codes to input characters and the most frequent character gets the smallest code and the least frequent character gets the largest code.

1.1 Project Aim and Objective

1.1.1. Objective

Our project aims to make a program which can compress a file for us. Thus resulting in reduced file size which can be easily sent or transferred faster than the original text file.

1.2 Requirements

We would be using Python for coding of the Huffman Coding Method and different python modules will be used to perform operations related to file compression.

CHAPTER 2 – METHODOLOGY

In this chapter, we will discuss and analyze about the file compression process using Huffman Coding method.

2.1. Huffman Coding

- Lossless compression algorithms reduce the size of files without losing any information in the file, which means that we can reconstruct the original data from the compressed file.
- Huffman coding is a lossless way to compress and encode text based on the frequency of the characters in the text. The most frequent character gets the smallest code and the least frequent character gets the largest code.
- The variable-length codes assigned to input characters are Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character.
- Compression technique works by creating a binary tree of nodes. There are mainly two major parts in Huffman Coding
    (a) Build a Huffman Tree from input characters.
    (b) Traverse the Huffman Tree and assign codes to characters.

2.2. LIBRARIES USED

- **OS –** This module provides the facility to establish the interaction between the user and the operating system.
- **HeapQ –** This module provides an implementation of the heap queue algorithm, also known as the priority queue algorithm.
- **Tkinter –** Tkinter is a standard Python interface to the Tk GUI toolkit shipped with Python.

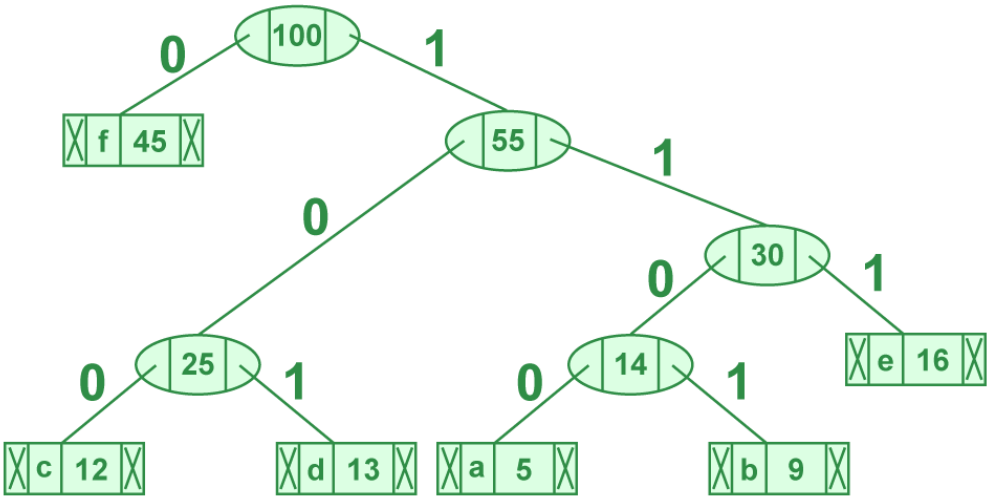
2.3. ALGORITHM

1. Build a min heap that contains N nodes where each node represents the root of a tree with a single node. All nodes have a weight equal to the weight of the character in the node.

2. Characters that occur most frequently have the highest weights. Characters that occur least frequently have the smallest weights. Repeat this step until there is only one tree.

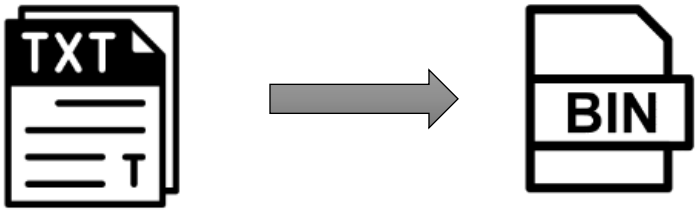
3. Choose two trees with the smallest weights, and call these trees T1 and T2. Create a new tree whose root has a weight equal to the sum of the weights T1 + T2, whose left subtree is T1, and whose right subtree is T2.

4. A single tree left after the previous step is an optimal encoding tree.



2.4. FILE COMPRESSION

1. Build frequency dictionary
2. Build priority queue (used MinHeap)
3. Build Huffman Tree by selecting 2 min nodes and merging them
4. Assign codes to characters (by traversing the tree from root)
5. Encode the input text (by replacing a character with its code)
6. If the overall length of the final encoded bit streams is not multiple of 8, add some padding to the text
7. Store this padding information (in 8 bits) at the start of the overall encoded bit stream
8. Write the result to an output binary file

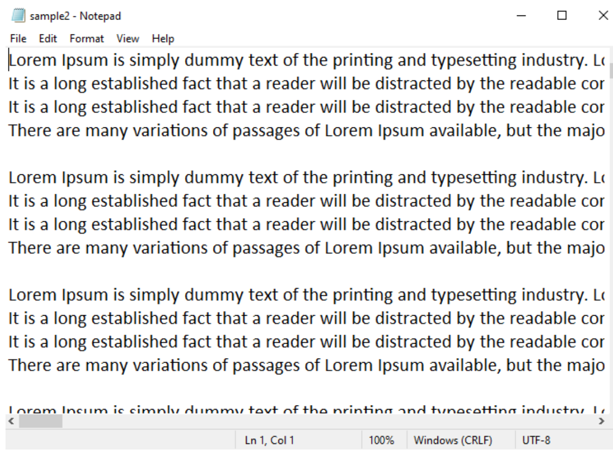


2.5. FILE DECOMPRESSION

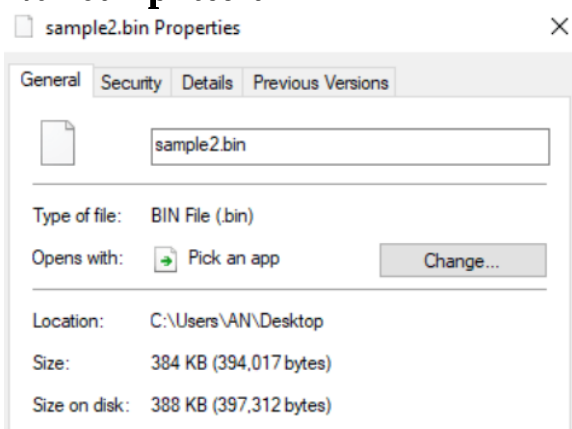
1. Read binary file
2. Read padding information. Remove the padded bits
3. Decode the bits - read the bits and replace the valid Huffman Code bits with the character values (Need to store the Huffman codes mapping while compression)
4. Save the decoded data into the output file (Gets original data back)

2.6. RESULTS

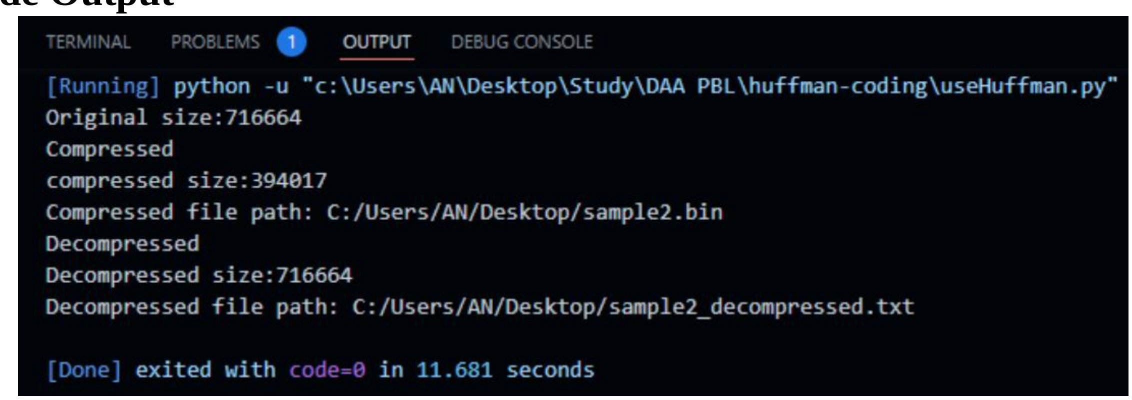
.txt file before compression



.bin file generated after compression



Code Output



Generated files

CHAPTER 3 – CONCLUSION

1. With the help of file compressor the user's input file is compressed drastically and can be used or transferred at a much faster speed.
2. This text file input of user gets successfully compressed and two files are generated with program. First is .bin compressed file and .txt decompressed file.
3. Further, Huffman Coding can be applied used for transmitting fax and text or they can be used for conventional compression formats like PKZIP, GZIP, etc.
4. Analysis of Program –

   The performance of the Huffman Code depends upon 2 factors

   - Average Length: It is defined as the average number of bits required to represent a character in the string.
   - Efficiency: Efficiency is a measure of the number of bits wasted.
5. **Time complexity: O(N log N)** where n is the number of unique characters.

References:

In given coursework after various research and practice we were able to know how File compression works using Huffman coding. Lots of websites, journals and books were consulted for gaining information on various important topics. Some of the references are mentioned below.

- https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/
- http://en.wikipedia.org/wiki/Huffman_coding
- https://www.section.io/engineering-education/huffman-coding-python/