IMPLEMENTING THE HUFFMAN CODING ALGORITHM

CO 201 : PROJECT PRESENTATION

SIDDHANT KHANNA - 2K20/CO/441
VARIN THAKUR - 2K20/CO/475

Table of contents

01

Introduction

What is Data Compression?

02

Basic Idea & Approach

The idea behind Huffman Coding Algorithm

03

Algorithm

A rough sketch

04

Our Implementation

This is where the magic happens!

05

Time Complexity

The one thing we all try to reduce.

06

Pros, Cons & Applications

Is it useful after all? Where is it used in Real Life Scenarios?

01

Introduction

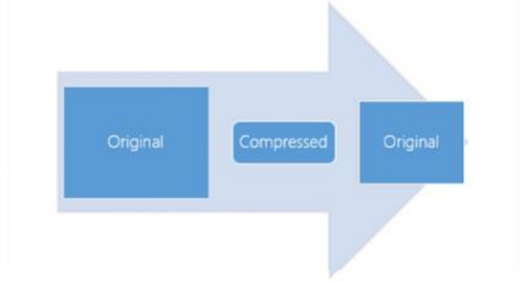
This one is sure not monotonous!

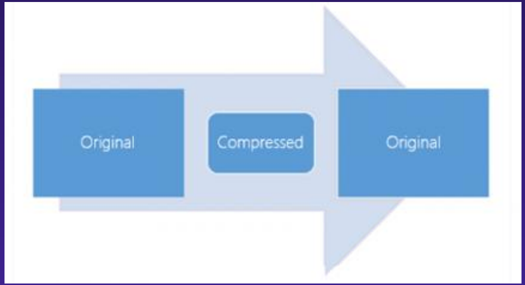
Data Compression

Reconstructing, Encoding or Modifying Data, in order to reduce its size.

Involves re-encoding information using fewer bits than original representation.

Types of Data Compression





Lossy Compression

Involves some loss of Data.

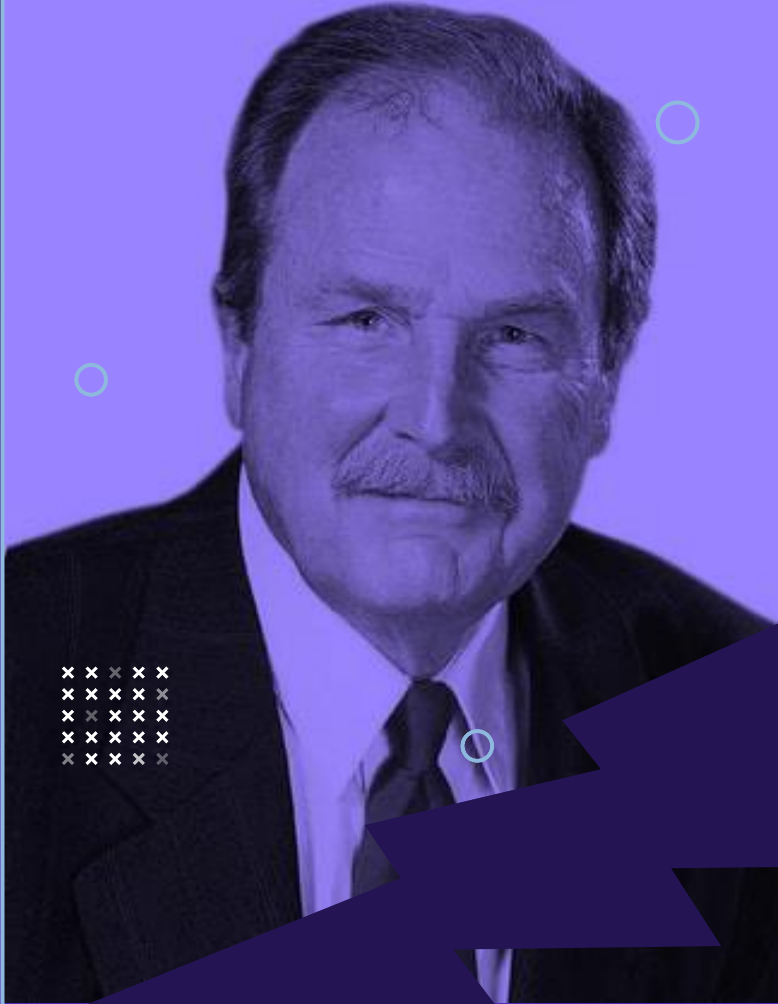
Lossless Compression

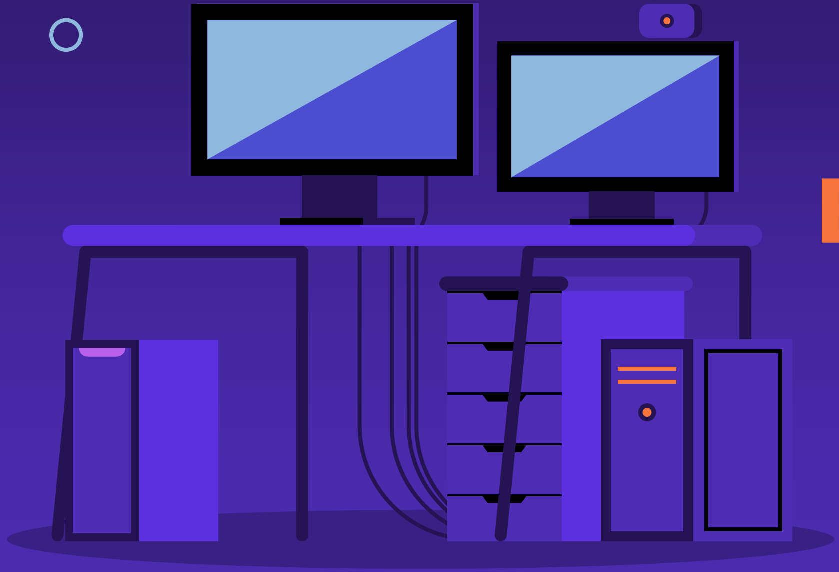
Involves no loss of Data.

Huffman Coding

Developed by David A. Huffman.

- A lossless Data Compression Algorithm.
- Assigns variable length bitstrings to characters.
- Frequent characters are assigned smaller codes for more efficiency.





02

Basic Idea & Approach

Knowing the basic technique

Presently

Characters are being represented by 8 bits.

Every string would be of 8 * length(string) bits in space.

Do we really need all 8 bits for a string with a specific number of Characters? – NO.

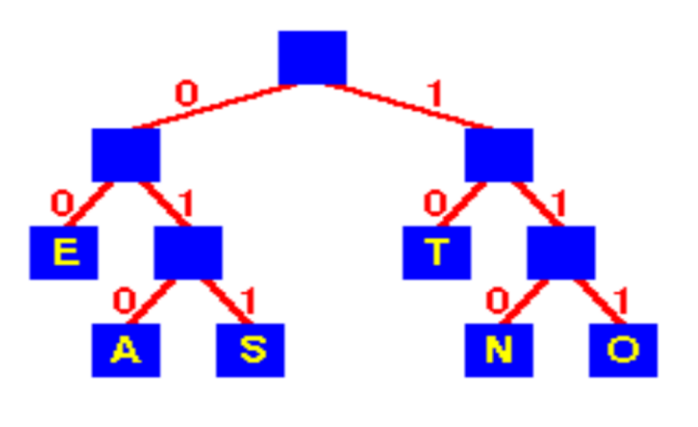
An idea

Shorter length codes for characters in our string!

Codes should be prefix free.

New size would be surely less.

Approach – A Huffman Tree



A Binary Tree

Complete Set of Codes is represented.

Leaf Nodes

While travelling from the root to the leaf,
⇒ Travel left , assign 0.
⇒ Travel right, assign 1.

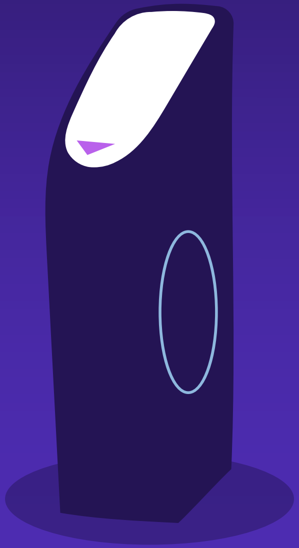
Efficient

Codes with shorter length are given to those with max frequency.

03

Algorithm

Expressing the approach in logical steps!



THE STRATEGY

Step 1

Make characters with their frequencies as leaf nodes.

Step 2

Extract 2 nodes with min. frequencies. Make a node with the sum of the frequencies as its parent.

Step 3

Repeat until a single node is left, assign it as the root of the Huffman Tree.

Step 4

Travel the tree and assign codes. While travelling to the left, assign 0 and while travelling to the right assign 1.

Step 5

Using these codes, encode and decode.

04

Our Implementation

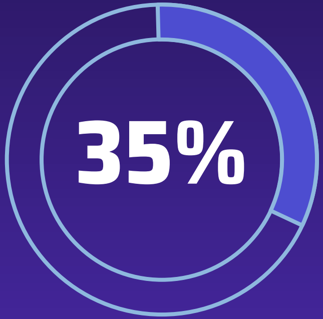
Idea brought into action!

05

TIME COMPLEXITY

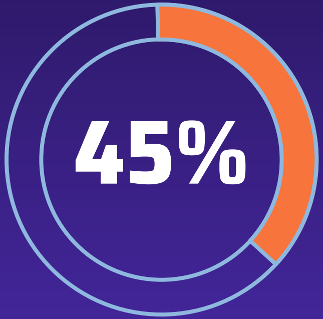
Figuring out the time taken for the code to execute!

The Time Taken = $T_1 + T_2 + T_3 + T_4$



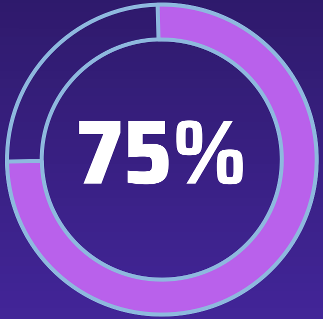
Inserting in Min Heap

$T_1 = n\log(n)$



Deleting from Min Heap

$T_2 = n\log(n)$



Assigning Codes

$T_3 = (2n - 1)$

Encode/Decode

$T_4 = n*\text{length(string)}$

06

Pros, Cons & Applications

Exploring the merits, flaws and practical uses

Two sides of a coin

Pros

- Since variables code lengths are assigned, this saves a lot of space.
- Binary codes generated are prefix-free, hence ambiguity is avoided.

Cons

- Extra space is used.
- A slower process as it happens in phases.
- Variable length codes make it difficult to check if the file is corrupt.

Real Life Applications

WinZip and WinRAR

Used in Compression Formats like GZIP and PKZIP.

Multimedia Codecs

Used in codecs in JPEG and MP3.

THANK YOU!