**GOVERNMENT COLLEGE UNIVERSITY LAHORE**

**Project Title :**

File (Text or Programs) Compression

**Type :**

Report of Proposed Project

**Section :**

C | 2020 - 2024

**Group No :**

53

**Submitted To :**

Professor Saeed-Ur-Rahman Turk

**Submitted By :**

Rizwan Ashraf (0084-BSCS-20)

Muhammad Rameez (0090-BSCS-20)

**DESIGN ANALYSIS AND ALGORITHM**

Table of Contents

[**Problem :** 3](#_Toc106835849)

[**Introduction :** 3](#_Toc106835850)

[**Lossless Compression Technique:** 3](#_Toc106835851)

[**Lossy Compression Technique :** 3](#_Toc106835852)

[**Encoding Digram :** 4](#_Toc106835853)

[**Decoding Digram:** 5](#_Toc106835854)

[**Algorithm of Encoding :** 6](#_Toc106835855)

[**Algorithm of Decoding :** 6](#_Toc106835856)

[**Conclusion :** 7](#_Toc106835857)

[**Refrence :** 7](#_Toc106835858)

# **Problem :**

As we know that the compression of files, texts, images videos is one of the important and useful research field. Various algorithms, programs and documentation have been written on their development. As we have large amount of data and as well as we have some applications that accept less size of data, therefore compression on these moments are very useful. Also it will take less space and program runs more quickly.

We can compress our data and can easily save and share it. There are different techniques are there on the internet and books here we are going to use Burrows-wheeler transform and we are also using Huffman coding which based on lossless text compression algorithm.

In our project we are going to use an algorithm with two keys that is used to reduce the frequently repeated characters after using BWT. Than we find patterns of certain length from the reduced text and apply Huffman coding. We will talk about the previous project’s algorithms that are used in compression

# **Introduction :**

Managing the increasing amount of data produced by modern daily life activities is not a simple task for symmetric communication. There were present a large amount of data which we need in our daily life and size of data could be zettabytes, Exabyte, terabytes, so these types of large data may result in use of large amount of storage size and as we discussed that some applications didn’t support sharing of large amount of data.

Here we need the compression techniques to reduce the size of data.

Compression reduce the data form so that it can use small storage. There are two types of data compression discussed below.

# **Lossless Compression Technique:**

In lossless compression technique original size of the data remain same. In this technique it allows the original data to be perfectly reconstructed from the compressed data.

The popular algorithms for this techniques are RLW, LZW, and Huffman coding etc. It is mostly used to compress the text or programs.

It maintains the quality of the data. Conversion is possible without loss of data quality.

It reduce the size of data but in comparison with lossy method it compression size is less. It is helpful when we didn’t want to reduce the quality of the data.

# **Lossy Compression Technique :**

In lossy Compression Technique, it removes some data from the original data in order to reduce the size of the data. It compress the data which is in the form of videos and audios. So here we can see that it reduce the number of colors in an image or can reduce the number of samples in a sound.

The popular algorithms for this compression are Transform coding, DCT, RSSMS etc. It is used to convert images, audios, and videos.

It makes some difference in the file which is compressed. Such as it reduce its quality. Its compression size is lesser than a file which is compressed by the lossless method.

**Background of File Compression Techniques:**

Huffman coding is one of the famous and brilliant data compression technique. Huffman coding is a tree, built bottom up. In the decomposition process returns the compressed file to the beginning of the text. The “decompression” results depends upon the nature of the compression used. For our project and in our study for file compression we see the Arithmetic compression technique that is a lossless technique. There is also a technique that is used in the compression which is Run Length Coding (lossless technique). In it a series of the data with the same value in sequence will be saved into a data. It is used where there a lot of data with the same values and have icons files, line drawings and animations and it is not suitable for the normal data.

LWZ i.e. Lempel Ziv Wich algorithm is a lossless technique. In it compression result in term of dictionary. Such as it find it find the new dictionary from the new character.

Huffman Coding which is our **main concern** in the project with BWT technique. In the method symbols or characters are encode with the help of the binary tree by combining the two smallest frequencies to form a code tree. It base on the number of the character frequencies that often appear. Larger the Huffman code frequency smaller or less number of bits appeared and conversely.

For example a file contain a row of characters “AAABCD” the number of bits are 6\*8 is 48 bits or 5 byte. And if each character is given a code e.g. A=0, B=10, C=111, D=110 then we only need a file with the size 10 bits. A code cannot form from the other codes.

**Proposed Method:**

We are going to use the Huffman coding with the borrow wheel transform algorithm. Many algorithms focus on the encoding or decoding speed of the algorithms during compression and some others concentrate on average algorithms.

In our work we split the large file into a set of smaller files, where each file will contain the exactly same number of characters than we apply borrow wheel transform to the smaller files individually to speed up the transformation. We do not use static dictionary because its time consuming and very complicated. We change a bit, use of run length coding after the borrow wheel transformation. Because run length coding takes symbol and its count and work well only when characters are repeated more in a text. When a character is alone in a text which normally happens, value of character and its count are stored after encoding which increase the storage size. Our next change from others programs is that we replace the characters repeated more than three times by the key, the character and its count. And the position of the character’s reduced text is identified by the key. Than Huffman coding provides more compression if a text have higher frequency of characters.

We perform encoding firstly than we about the decoding. This encoding will be helpful for us and in next phase of our project we will see our proposal algorithm but before this we will see the decoding procedure that how we use above techniques to decode the encoded text. Than we see both algorithms.

We have proposed a completely lossless text compression using Huffman coding, two keys, and Borrows wheeler transform for the compression of the text.

It ultimately increase the compression technique when we split the original text into the smaller set of texts. And we do not count all the characters that is done in run length coding, we only count the characters that are repeated more than two times consecutively and then replace value of the letter count by a lower case letters.

# **Encoding Digram :**

Original Text File

Text1

Text2

Text n

Text n-1

Transformed Text

Reduce the length by two keys

Encoded Bit Stream

Find the Specific Pattern

**Fig: (a) General Encoding diagram for our Proposal**

BWT

Split the text file

Apply Huffman Coding

**Explanation :**

The above fig (a) shows the procedure of our proposal that what will happen in the encoding procedure of algorithm. It shows that firstly our original file splits in the smaller set of text files than these text files go through the procedure of the borrow wheel transformation and transformed the text than we use run length coding but differently i.e. by two keys like when we got character more than tree repeat than we set it in a key have the character and a count. This will reduce time and complexity from other methods. Than we use Huffman encoding coding for further encoding. In this way our encoding is done for a file. Which is different from just using Huffman or LWZ and others. This encoding will be helpful for us and in next phase of our project.

# **Decoding Digram:**

Encoded Bit Stream

Apply Huffman decoding

Perform Enlargement of text based upon two keys

Text1

Text n

Text n-1

Text2

Split the text

Reconstructed Text

BWT use

**Fig: (b) General Diagram for the decoded technique of our proposal**

# **Algorithm of Encoding :**

**1:** Read the text file as an input and calculate the length of the file (N).

**2:** Split the text into a set of small files of the same size (SS).

**3:** Apply Borrows Wheel Transform on each text file separately and save the corresponding transform key.SetI =1;

**4:** **while** i <= N do

**5: if** the number of same consecutive numbers is greater than 3 then

**6:** store tuple (key1, the character and their count) to ReducedText where count is converted into a character using the formula counts+96;

**7:** I=I + count;

**8:** **else if** the number of the same consecutive characters is exactly three **then**

**9:** store key2 and character to ReducedText;

**10:** I=I+1;

**11:** end// else if

**12:** else

**13:** save the character to the array ReducedText;

**14:** I=I+1;

**15:** end// else

**16:** end// while

**17:** Find the specific patterns from the reduced text that have higher frequencies;

**18:** Apply Huffman coding technique to get an encoded bit stream

# **Algorithm of Decoding :**

**1:** Apply the Huffman decoding technique on the receive encoded bit-stream and store to PreReconstructedText1;

**2:** Set I=1 and calculate the length of the PreReconstructedText1 (Len);

**3: while** I<=Len do

**4:**  **if** PreReconstructedText1 [I] =Key1 then

**5:**  Add (int (PreReconstructedText1 [I+2])-96) number of PreReconstructedText1 [I+1] character to PreReconstructedText2;

**6:** I =I+3;

**7: else if** PreReconstructedText1 [I] ==key2 then

**8:** Add three PreReconstructedText1 [I+1] characters to PreReconstructedText2;

**9:** I =I+2;

**10:** end;

**11: else**

**12:** Add PreReconstructedText1 [I] character to PreReconstructedText2;

**13:** I =I+1;

**14:** end// if

**15:** end // while

**16:**  Split PreReconstructedText2 into a set of small files, where each file contains SS number of character and apply the inverse Burrows-Wheeler transform to each file with its corresponding Transform get back and reconstructed the Text;

# **Conclusion :**

When we have less storage and have a highly narrow band communication channel. We have proposed a completely lossless text compression using Huffman coding, two keys, and Borrows wheeler transform for the compression of the text.

It ultimately increase the compression technique when we split the original text into the smaller set of texts. And we do not count all the characters that is done in run length coding, we only count the characters that are repeated more than two times consecutively and then replace value of the letter count by a lower case letters.

Then we look for the patterns of certain length that have high frequencies so that we can get better result after applying the Huffman coding.

# **Refrence :**

* Semantic Scholar
* Managing free context center
* Internet
* GitHub
* Research Gate