**1.Implementation**:

COSC 5330-01  
Multimedia Processing  
  
INSTRUCTOR:  
Prof. Jiangjiang (Jane) Liu  
  
  
Implement encoder and decoder for

LZW compression algorithms  
  
  
Name: ANIL K PUPPALA  
LUID: L20375862  
Date: 07/10/2015

The project is prepared to encode and decode files of various formats like text file, image file, audio file, dll file and also executable file. Using the LZW implementation I was able to encode all these files with a remarkable compression ratio. I also used WINZIP for the file compression and noted compression ratio to compare with my LZW implementation. The compressed files were created with LZW file format. I have also implemented the decompression technique to convert the compressed files to the original format using the Java source code.

* 1. **Encoding**

The input files for encoding are of formats like text, image, audio, dll and exe.

**Result**

Files with LZW format were created with a noticeable compression ratio after compression.

* 1. **Decoding**

The compressed files created after the encoding are taken as input parameter.

**Result**

The files are decompressed converted to the original format.

* 1. **WINZIP Encoding**

Now the files are compressed using the WINZIP encoder and the compression ratios are noted.

**2. Comparison**

When compared with the WINZIP compression ratio, my LZW implementation as a better compression ratio.

**3. Methodology**

LZW encoding algorithm is used to compress the files and LZW decoding algorithms to decompress the compressed file.

**3.1 LZW Encoding Algorithm**

* If the message to be encoded consists of only one character, LZW outputs the code for this character; otherwise it inserts two- or multi-character, overlapping (The last character of a pattern is the first character of the next pattern) distinct patterns of the message to be encoded in a Dictionary.
* The patterns are of the form: C0C1 . . . Cn-1Cn. The prefix of a pattern consists of all the pattern characters except the last: C0C1 . . . Cn-1
* Pseudo-Code

Initialize Dictionary with 256 single character strings and their corresponding ASCII codes;

***Prefix****← first input character;*

***CodeWord****← 256;*

*while(not end of character stream)*

*{*

***Char****← next input character;*

*if(****Prefix + Char*** *exists in the Dictionary)*

*Prefix ←****Prefix + Char****;*

*Else*

*{*

***Output:*** *the code for* ***Prefix****;*

*insertInDictionary( (CodeWord ,* ***Prefix + Char****)*

*CodeWord++;*

***Prefix****←****Char****;*

*}*

*}*

* 1. **LZW Decoding Algorithm**
* The LZW decompressor creates the same string table during decompression.
* Initialize Dictionary with 256 ASCII codes and corresponding single character **strings** as their translations;
* Pseudo Code

*PreviousCodeWord← first input code;*

***Output:****string(PreviousCodeWord) ;*

*Char ←character(first input code);*

***CodeWord****← 256;*

*while(not end of code stream){*

*CurrentCodeWord← next input code ;*

*if(****CurrentCodeWord*** *exists in the Dictionary)*

*String ←string(CurrentCodeWord) ;*

*else*

*String ←string(PreviousCodeWord) + Char ;*

***Output:*** *String;*

*Char ← first character of String ;*

*insertInDictionary( (****CodeWord*** *, string(****PreviousCodeWord****)* ***+ Char****) );*

*PreviousCodeWord←CurrentCodeWord ;*

*CodeWord++ ;*

**4. Testing Results:**

I was able to compress and decompress 5 different types of files. For example audio files, EXE files, image files, data files-with .doc and dll files. I used JAVA to develop my application which involved concepts like HASH TABLES, SWINGS in JAVA to build both compression and decompression techniques.

The files used in my project are:

Word File----------Content.doc

Image File--------- WP\_20141106\_003.jpg

Audio File--------- Neverbackdown.mp3

DLL File-----------example.dll

EXE File ----------Sample.exe

These files were compressed to reduce the size and new files with .lzw extension were created.

**Compression Analysis:**

In Following Table, we have calculate compression ratio using following formula:

LZW Compression Ratio = Original Size/LZW Size

WINZIP Compression Ratio = Original Size/WINZIP Size

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **File Name** | **Original Size** | **LZW-SIZE** | **LZW Compression Ratio** | **WINZIP SIZE** | **WINZIP Compression**  **Ratio** |
| NeverBackDown.mp3 | 3124KB | 3192KB | 0.97 | 3091KB | 1.01 |
| Content.doc | 1080KB | 1043KB | 1.04 | 381KB | 2.75 |
| example.exe | 11328KB | 9361KB | 1.21 | 2186KB | 5.11 |
| Sample.dll | 3170KB | 3138KB | 1.01 | 1205KB | 2.63 |
| WP\_20141106\_003.jpg | 1293KB | 1637KB | 0.78 | 1097KB | 1.03 |

The above table shows the compression ratios of all the files compressed using LZW and WINZIP.

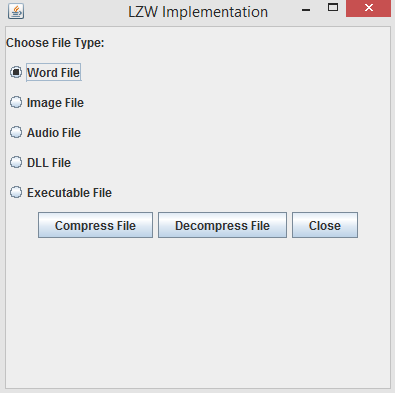
**Compression Ratio**

**5. User Manual:**

**User Manual:**

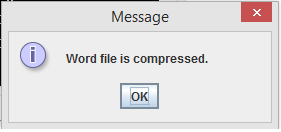
The User can select a radio button for the file compression based on the format of the file which he wants to compress.

Below is the screen shot for the types of formats available in my application:



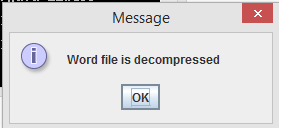
1. After Compress Files:

After selecting the file from the location, user can click on compress button to compress files. After compressing file, Following Message box is displayed:



1. To Decompress Files:

To decompress file, user have to select a radio button and click on decompress button. If File has already encoded then decompression will be done successfully and message box will be displayed as under:



If File is not encoded earlier then no file is found for decompression. So following message will be displayed as under:

