CS 337: Algorithms: Design & Practice

Lab#11 Data Compression: LZW Algorithm

In this lab we will implement the LZW algorithm for lossless data compression of text files (Cormen, Chapter 9, MacCormick, Chapter 7). Implementing LZW requires two parts: a procedure to compress the given text, and procedure to decompress the compressed text. In this lab, we will implement the compression algorithm. Code for decompression is provided. Here is the algorithm:

```
Procedure LZW-compress(text)

Inputs: text: A sequence of characters in the ASCII set.
Results: A sequence of indices.

Let: dictionary = hashtable with strings as keys and
integers as values, initially empty
Let: Output = an empty list
For j in 0..255:
   add to DICT key=string_value(j), value=j
       // see page 3 for string value of an integer

Let: dp = 256
Set: s to the first character of text
while text is not exhausted, do the following:
   Let: c = the next character from text
   Let: sc = s+c //ie append c to the string s
   If sc is in dictionary
      set s to sc
   else
      Append to Output the value of s in dictionary
      Add to dictionary key=sc value=dp
      Increment dp
      Set s = c

Append to Output the value of s in dictionary
return Output
```

For example, given the text

`It was the best of times, it was the worst of times.`

The procedure above will return the following list

[73 116 32 119 97 115 32 116 104 101 32 98 101 115 257 111 102 262 105 109 268 44 32 105 257 259 261 263 265 119 111 114 269 32 271 273 275 115 46]

As you can see, the output is a number of indices into the dictionary. For example, the first index, 73, corresponds to the entry "I" (in fact, it is the ASCII code for "I"). Indices from

256 onwards are entries created by the compression procedure and represent composite strings from the text.

Task#1: Implement the LZW-compress procedure in either Java or Python, based exactly (and only) on the procedure on page 1.

Test your compression procedure on the example text above. Your program for Task 1 should print out exactly the numbers shown above in exactly the order above.

Another sample — this sample should have NO carriage returns:

```
That struts and frets his hour upon the stage, and then is
heard no more. It is a tale told by an idiot, full of sound
and fury, signifying nothing.
```

[84 104 97 116 32 115 116 114 117 116 115 32 97 110 100 32 102 114 101 265 32 104 105 266 104 111 117 114 32 117 112 111 110 32 116 104 101 260 116 97 103 101 44 267 269 289 291 288 278 276 101 97 114 270 110 111 32 109 111 273 46 32 73 259 304 97 289 97 108 292 116 111 108 270 98 121 299 32 105 100 105 111 116 298 102 117 108 108 32 111 102 260 281 300 268 270 340 114 121 298 115 105 103 110 105 102 121 105 110 103 32 310 290 363 103 46]

When done compressing, your program should also print out the length of the text (in bytes) as well as the number of bytes needed to store the compressed text (assume each integer takes four bytes) for comparison purposes. Also show the number of entries in the dictionary. Put this info into the table on page 4.

Task#2: Modify your program to write a compressed binary file when reading from an uncompressed file. (See hints below for writing binary files and reading non-binary files.) Use your program to compress the samples above (put them in files) and the full text of Moby Dick. It is available at:

Moby Dick file: /home/gtowell/Public/337/`Lab11/moby10b.txt`

Once Moby Dick and the samples are compressed into files, decompress those files using either (or both!) of the programs below. NO CODE WRITING. Just use these programs. The output should be identical to the original text. If it is not identical, the problem is with your compression program. (You can use — and should — the UNIX 'diff' command to check that the files are identical.) Differing by an extra return after the last line is OK. Indeed an extra return is expected if you use the Java reader given on page 3.

Python: /home/gtowell/Public/CS337/`Lab11/LZWde.py`
Java: /home/gtowell/Public/CS337/`Lab11/LZWde.java`

Helpful Hints:

Note that there are many other ways to do the tasks below

To read the contents of a text file:

Java:

```
     public String readFile(String filename) {
        StringBuffer sb = new StringBuffer();
        try (BufferedReader br = new BufferedReader(new
FileReader(filename))) {
            String line = "";
            while ((line = br.readLine()) != null) {
                sb.append(line+"\n");
            }
        } catch (Exception ee) {
            System.err.println("Problem reading file" + ee);
        }
        System.out.println("Bytes read " + sb.length());
        return sb.toString();
    }
```

Python:

```
with open(fileName, “rb”) as instream:
     Contents = inS.read().decode(“utf-8”)
```

To convert an integer value into ASCII code string (e.g. to convert code 65 (“A”))

Python:
```
aa = chr(65)
print(aa)
```

Java:
```
String aa = "" + ((char) 65);
System.out.println(aa);
```

To write a binary file of Integers:

Java:

```
import java.io.DataOutputStream;
import java.io.FileOutputStream;

public class BW {
    public static void main(String[] args) {
        try (DataOutputStream dos = new DataOutputStream(new
FileOutputStream("BW.bin"))) {
            dos.writeInt(5);
            dos.writeInt(12345);
        } catch (Exception ee) {
            System.err.println(ee);
        }
    }
}
```

Python:

```
with open('BWp.bin', 'wb') as file:
    file.write((5).to_bytes(4, byteorder='big', signed=False))
    file.write((12345).to_bytes(4, byteorder='big', signed=False))
```

As you work through the lab, record your results in the table below:

	Size of Text (in bytes)	**Output of Compress (in bytes)**	**#Entries in Dictionary (compress)**	**Output of Decompress (in bytes)**
Sample Text (Page 1)				
Macbeth (Page 2)				
Moby Dick (Input from file)				

What to hand in

A **report** describing:

- your implementation of the compression algorithm (include, at least portions of, your actual code of the compression procedure in the narrative).
- the decompression algorithm.   You do not have to write this code, as I provide it in both Python and Java. You must describe how it works.
- Using the results from the table above, discuss the compression algorithm itself. Be sure to include the table in your report. In this discussion, you might consider the following questions:
    - How could you improve compression with only minor changes to the algorithm? (Your change may require changes in the decompression algorithm.)
    - What would happen if a character not in ASCII 0-255 was in the text?   Why?
    - Why is the “compressed” of the two sample texts actually larger then the source?
        - What sort of changes to the algorithm would prevent this issue?
- Include your compression code in an appendix.

Approximate Rubric

Description	**Point Value**
Introduction / conclusions	10
Compression code descriotion	20
Decompression code description	20
Table	10
Discussion of compression	30
Appendix with compression code	10

Secret identity
2 movie superheroes