# CMPT 365 Multimedia Systems

## Programming Assignment 2

## Question 1

**In this section, we are required to read a sequence that consists of letters A, B, and C only and assume that the initial dictionary is**

**0 A**

**1 B**

**2 C**

**3 AB**

**4 BA.**

**The mission was divided into three parts:**

**Part 1. Print out the entropy of the input sequence**

The entropy of a sequence represents the degree of confusion of the sequence.In this section， I implement Equation 1 via a hash table that calculates the entropy value of the input sequence and prints it to the console, as shown in Figure 1. H denotes entropy and P denotes the probability of pairing elements in a sequence.


Figure 1. result

$$H\ (x)\ = -\ ((P0 \times \log2(P0))(P0 \times \log2(P0)) + \ldots + (Pm \times \log2(Pm)))$$

Equation 1. sequence entropy

**Part 2. Print out the output sequence for the input sequence using the LZW coding**

**algorithm.**

The basic principle of LZW is to extract different characters from the original text file data, create a compilation table based on these characters, and then replace the corresponding characters in the original data with the index of the characters in the compilation table, reducing the size of the original data.

From a technical point of view, the difficulty of this task lies in how to achieve compression encoding through code. To solve this problem, I wrote code with Java, which I was better at.Firstly, I set my initial characters into the dictionary, then set two variables P and C.P represents an existing string that has not yet been encoded, and C represents a newly read character. In the initial state, only all default entries are in the dictionary, and both P and C are empty. After receiving the new character, P and C are stitched together into the new character P+C, and then I look up the dictionary for the presence of the character P+C. If P+C is in the dictionary, then P = P+C, otherwise the tick of P is output and a tag mapping is created for P+C in the dictionary, updating P=C.Repeat until you have read all the characters in the original string. Finally, when the loop is over,I create a tag mapping for P in the dictionary.The effect is shown in Figure 2.

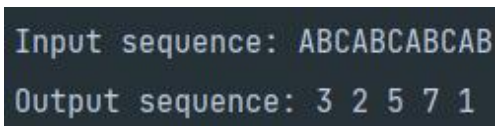


Figure 2.the output after compression encoding

**Part 3.Print out the dictionary in the end of the LZW coding.**

In this section, I will output the LZW processed dictionary through the cache stream that java comes with, which saves a lot of time.

The effect is shown in Figure 3.

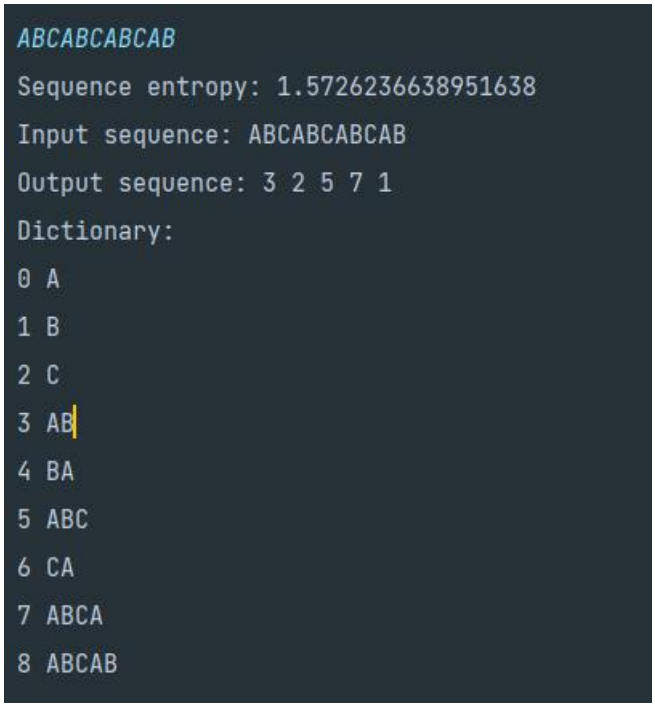


Figure 3.dictionary after LZW processing