Lempel-Ziv-Welch (LZW) Compression Algorithm

- Introduction to the LZW Algorithm
- LZW Encoding Algorithm
- LZW Decoding Algorithm
- LZW Limitations

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*, distinct patterns of the message to be encoded in a Dictionary.

*The last character of a pattern is the first character of the next pattern.

- The patterns are of the form: $C_0C_1 \ldots C_{n-1}C_n$. The prefix of a pattern consists of all the pattern characters except the last: $C_0C_1 \ldots C_{n-1}$

LZW output if the message consists of more than one character:

- If the pattern is not the last one; output: The code for its prefix.
- If the pattern is the last one:
  - if the last pattern exists in the Dictionary; output: The code for the pattern.
  - If the last pattern does not exist in the Dictionary; output: **code(lastPrefix)** then output: **code(lastCharacter)**

Note: LZW outputs codewords that are 12-bits each. Since there are 2^{12} = 4096 codeword possibilities, the minimum size of the Dictionary is 4096; however since the Dictionary is usually implemented as a hash table its size is larger than 4096.

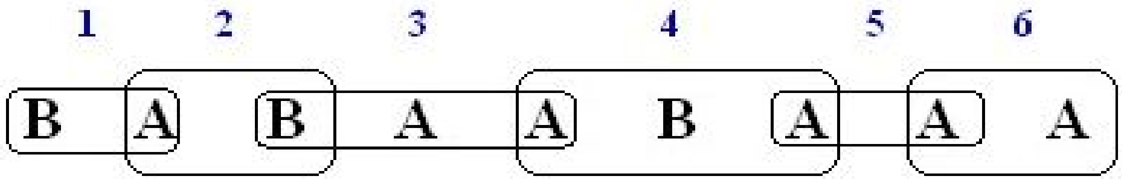
LZW Encoding Algorithm (cont'd)

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;
         }
}

Output: the code for Prefix;
```

Example 1: Compression using LZW

Encode the string **BABAABAAA** by the **LZW** encoding algorithm.



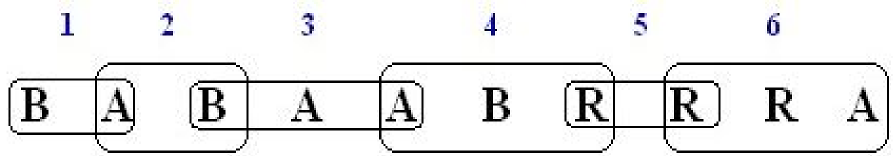
1. BA is not in the Dictionary; insert **BA**, output the code for its prefix: **code(B)**
2. AB is not in the Dictionary; insert **AB**, output the code for its prefix: **code(A)**
3. BA is in the Dictionary.
BAA is not in Dictionary; insert **BAA**, output the code for its prefix: **code(BA)**
4. AB is in the Dictionary.
ABA is not in the Dictionary; insert **ABA**, output the code for its prefix: **code(AB)**
5. AA is not in the Dictionary; insert **AA**, output the code for its prefix: **code(A)**
6. AA is in the Dictionary and it is the last pattern; output its code: **code(AA)**

output	Dictionary	
	Code Word	String
66	256	BA
65	257	AB
256	258	BAA
257	259	ABA
65	260	AA
260		

The compressed message is: **<66><65><256><257><65><260>**

Example 2: Compression using LZW

Encode the string **BABAABRRRA** by the LZW encoding algorithm.



1. **BA** is not in the Dictionary; insert **BA**, output the code for its prefix: **code(B)**
2. **AB** is not in the Dictionary; insert **AB**, output the code for its prefix: **code(A)**
3. **BA** is in the Dictionary.
   BAA is not in Dictionary; insert **BAA**, output the code for its prefix: **code(BA)**
4. **AB** is in the Dictionary.
   ABR is not in the Dictionary; insert **ABR**, output the code for its prefix: **code(AB)**
5. **RR** is not in the Dictionary; insert **RR**, output the code for its prefix: **code(R)**
6. RR is in the Dictionary.
   RRA is not in the Dictionary and it is the last pattern; insert **RRA**, output code for its prefix: **code(RR),** then output code for last character: **code(A)**

output	Dictionary	
	Code Word	String
66	256	BA
65	257	AB
256	258	BAA
257	259	ABR
82	260	RR
260	261	RRA
65		

The compressed message is: **<66><65><256><257><82><260> <65>**

LZW: Number of bits transmitted

Example: Uncompressed String: **aaabbbbbbaabaaba**

Number of bits = Total number of characters * 8

= 16 * 8

= 128 bits

Compressed string (codewords): **<97><256><98><258><259><257><261>**

Number of bits = Total Number of codewords * 12

= 7 * 12

= 84 bits

Note: Each codeword is 12 bits because the minimum Dictionary size is taken as 4096, and

$2^{12} = 4096$

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;
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++ ;
}
```

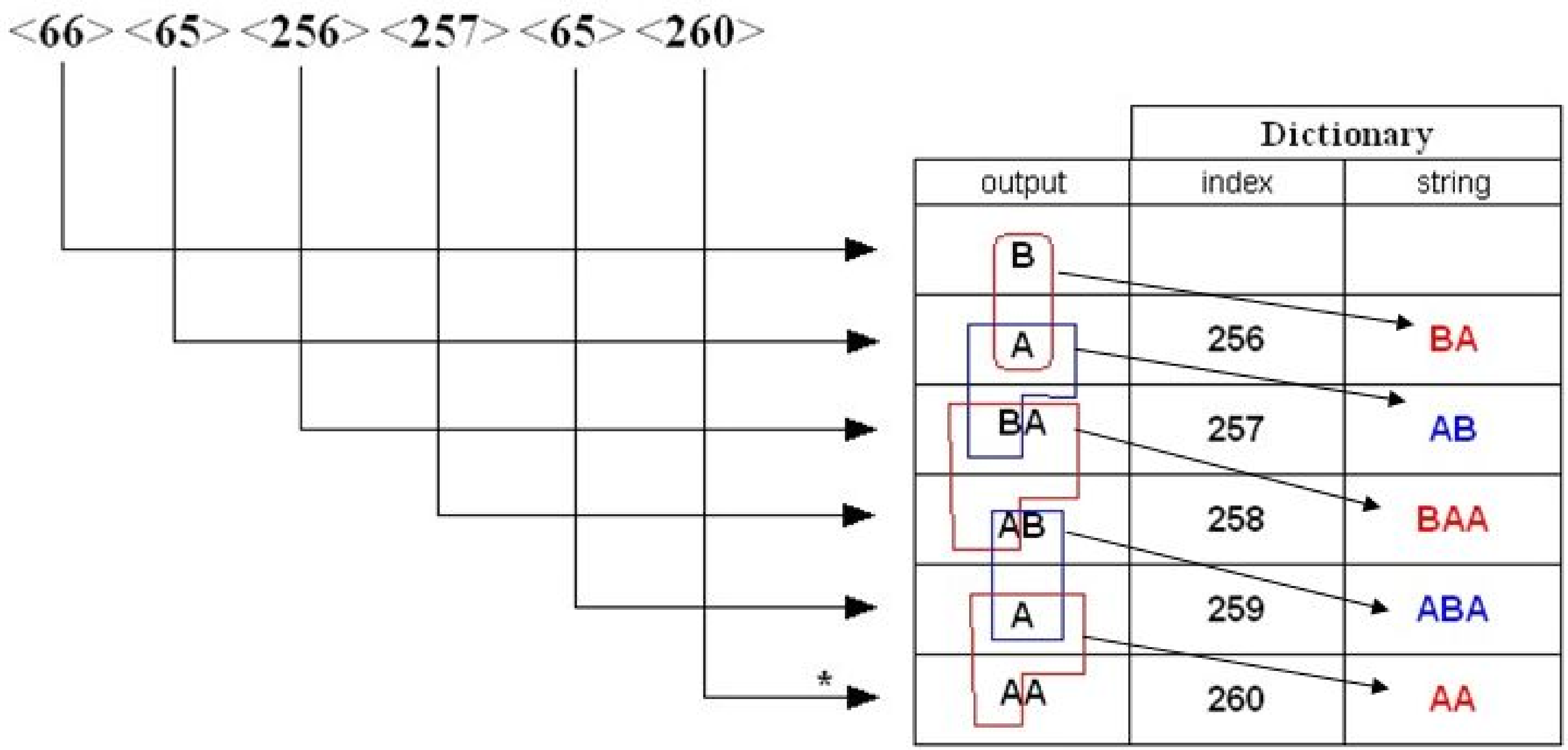
LZW Decoding Algorithm (cont'd)

Summary of **LZW** decoding algorithm:

```
output: string(first CodeWord);

while(there are more CodeWords){
    if(CurrentCodeWord is in the Dictionary)
      output: string(CurrentCodeWord);
   else
      output: PreviousOutput + PreviousOutput first character;



   insert in the Dictionary: PreviousOutput + CurrentOutput first character;
}
```

Example 1: LZW Decompression

Use LZW to decompress the output sequence <66> <65> <256> <257> <65> <260>



1. **66** is in Dictionary; output **string(66)** i.e. **B**
2. **65** is in Dictionary; output **string(65)** i.e. **A**, insert BA
3. **256** is in Dictionary; output **string(256)** i.e. **BA**, insert **AB**
4. **257** is in Dictionary; output **string(257)** i.e. **AB**, insert **BAA**
5. **65** is in Dictionary; output **string(65)** i.e. **A**, insert **ABA**
6. **260** is **not** in Dictionary; output

   previous output + previous output first character: AA, insert AA

Example 2: LZW Decompression

Decode the sequence **<67> <70> <256> <258> <259> <257>** by LZW decode algorithm.

<67> <70> <256> <258> <259> <257>

	Dictionary	
output	index	string
C		
F	256	CF
CF	257	FC
* CFC	258	CFC
* CFCC	259	CFCC
FC	260	CFCCF

1. **67** is in Dictionary; output **string(67)** i.e. **C**
2. **70** is in Dictionary; output **string(70)** i.e. **F**, insert **CF**
3. **256** is in Dictionary; output **string(256)** i.e. **CF**, insert **FC**
4. **258** is **not** in Dictionary; output **previous output + C** i.e. **CFC**, insert **CFC**
5. **259** is **not** in Dictionary; output **previous output + C** i.e. **CFCC**, insert **CFCC**
6. **257** is in Dictionary; output **string(257)** i.e. **FC**, insert **CFCCF**

LZW: Limitations

- What happens when the dictionary gets too large?
- One approach is to clear entries 256-4095 and start building the dictionary again.
- The same approach must also be used by the decoder.

Exercises

- Use LZW to trace encoding the string ABRACADABRA.
- Write a Java program that encodes a given string using LZW.
- Write a Java program that decodes a given set of encoded codewords using LZW.