**Complexity of LZW Algorithm**

**Time Complexity:**

We will be calculating the time complexity of LZW Compression Algorithm by calculating the time taken by each statement of the pseudo code.

In the following calculation n represents the number of characters read from the file which is to be compressed.

***Statement Cost Time*s**

STRING = get input symbol 1

WHILE there are still input symbols DO n

SYMBOL = get input symbol n-1

IF STRING + SYMBOL is in the STRINGTABLE THEN n-1

STRING = STRING + SYMBOL

ELSE

Output the code for STRING

Add STRING + SYMBOL to STRINGTABLE

STRING = SYMBOL

END

END

Output the code for STRING 1

The running time of the algorithm is the sum of running times for each statement executed; a statement that takes steps to execute and executes n times will

contribute n to the total running time.

For Best Case value of x will be small and value of y will be large Means the statement in if condition will be repeated more times than the statement in the else condition. Opposite to this for Worst Case x will be large and y will be small. Since here we are discussing the worst case so we assume that y=1 and x is maximum so that the statement in if condition is repeated very less or it runs 0 times means every time loop is repeated it goes in else statement.

To compute T (n), the running time of LZW on an input of n characters, we sum the products of the cost and times columns, obtaining

T(n) = + n + (n-1) +(n-1) + (0) +(n-1) +(n-1) +(n-1) +

= + n + n - +n - +n - +n - +n - +

= (+ + + + + ) n + (- - - - - + )

Here we let + + + + + = a and - - - - - + = b

T(n) = an + b

We expressed the worst-case running time as an + b for some constants a, and b that depend on the statement costs . We thus ignored not only the actual statement costs, but also the abstract costs .

We shall now make one more simplifying abstraction: it is the **rate of growth**, or **order of growth**, of the running time that really interests us. We therefore consider only the leading term of a formula which is ***an***. We also ignore the leading term’s constant coefficient, since constant factors are less significant than the rate of growth in determining computational efficiency for large inputs. We are left with the factor of n from the leading term. We write that LZW has a worst-case running time of O (n) (pronounced as Big O of n).

T(n) = O (n).

Hence our algorithm takes linear time.

**Space Complexity:**

We know that Space Complexity Depends on the Memory taken by the Algorithm to complete its execution. In Addition to memory used by code ( Variables initialized etc) this algorithm also loads the file to memory/buffer which is to be compressed. Thus the Space Complexity of LZW heavily depends on size of file because the file is first loaded in memory and then it is encoded.