**Burrows Wheeler**

**Problem Statement**

Burrows Wheeler is a transformation technique that rearranges a character string into runs of similar characters. This is useful for compression, because it is easy to compress a string that has runs of repeated characters by making use of techniques like move-to-front transform, run-length encoding etc. In this project we use a move-to-front transformation technique. Burrows Wheeler transformation is reversible and does not require to store any additional information except the position of the first original character.

The program expects us to

1, Perform Move - To - Front encoding and decoding

2. Find all the Circular Suffixes of the given String

3. Burrows Wheeler Transformation

**Related Concepts**

In order to sort less than nlogn time complexity we have sorting techniques like LSD, MSD, 3-way radix sort etc. The advantage with 3 - way radix sorting technique is that it does not compare the entire string even though both suffixes are same.

**Test Cases**

1. Two Test cases Failed and scored 95/100.
2. The two test cases are related to memory.

**API**

**MoveToFront :**

public class MoveToFront {

**Time Complexity :** length of input \* Radix (R)

**Space Complexity :** length of input + Radix (R)

// apply move-to-front encoding, reading from standard input and writing to standard output

public static void encode()

**Time Complexity :** length of input \* Radix (R)

**Space Complexity :** length of input + Radix (R)

// apply move-to-front decoding, reading from standard input and writing to standard output

public static void decode()

}

**Circular Suffix Arrays :**

public class CircularSuffixArray {

**Time Complexity :** nlogn

**Space Complexity :** length of input + Radix (R)

// circular suffix array of s

public CircularSuffixArray(String s)

**Time Complexity :** Constant

**Space Complexity :** Constant

// length of s

public int length()

**Time Complexity :** Constant

**Space Complexity :** Constant

// returns index of ith sorted suffix

public int index(int i)

}

**Burrows Wheeler :**

public class BurrowsWheeler {

**Time Complexity :** length of input + Radix (R)

**Space Complexity :** length of input + Radix (R)

// apply Burrows-Wheeler transform,

// reading from standard input and writing to standard output

public static void transform()

**Time Complexity :** length of input + Radix (R)

**Space Complexity :** length of input + Radix (R)

// apply Burrows-Wheeler inverse transform,

// reading from standard input and writing to standard output

public static void inverseTransform()

}