**ADVANCED DATA STRUCTURES - PROJECT REPORT**

**216-A**

**BurrowsWheeler**

By

RANJITH KUMAR K(2019501102)

*Abstract:*

The Burrows–Wheeler data compression algorithm consists of three algorithmic components, which are applied in succession:

1. *Burrows–Wheeler transform.* Given a typical English text file, transform it into a text file in which sequences of the same character occur near each other many times.
2. *Move-to-front encoding.* Given a text file in which sequences of the same character occur near each other many times, convert it into a text file in which certain characters appear much more frequently than others.
3. *Huffman compression.* Given a text file in which certain characters appear much more frequently than others, compress it by encoding frequently occurring characters with short codewords and infrequently occurring characters with long codewords.
4. **Introduction:**

In this burrows wheeler project, we have implemented methods to encode and decode the strings,transform the string and then compress the sequence such that all the sequence of characters occur near each other many times or frequently. This is mainly for the data compression.It applies the data compression algorithm burrows wheeler which consists of methods such as burrowswheelertransform(), movetoFront(),and circularsuffixarray() methods.

**2.Terms and Methods Implemented:**

**Huffman compression and expansion.** [Huffman](https://algs4.cs.princeton.edu/code/javadoc/edu/princeton/cs/algs4/Huffman.html) (Program 5.10 in *Algorithms, 4th edition*) implements the classic Huffman compression and expansion algorithms.

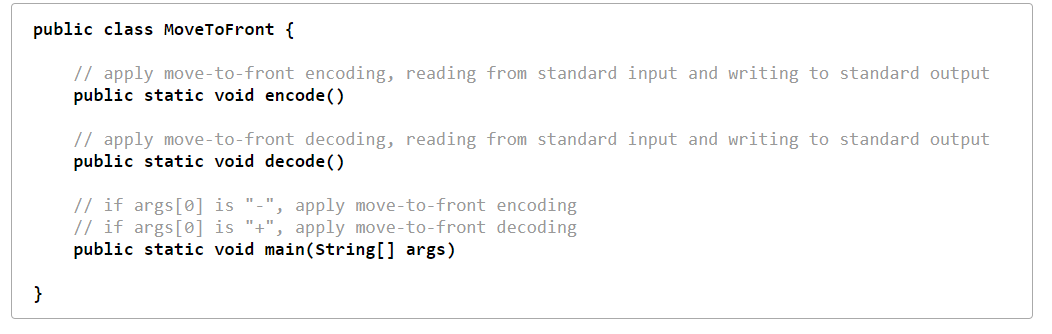
***Burrows–Wheeler transform****.* The Burrows–Wheeler transform of a string *s* of length *n* is defined as follows: Consider the result of sorting the *n* circular suffixes of *s*. The Burrows–Wheeler transform is the last column in the sorted suffixes array t[], preceded by the row number first in which the original string ends up.

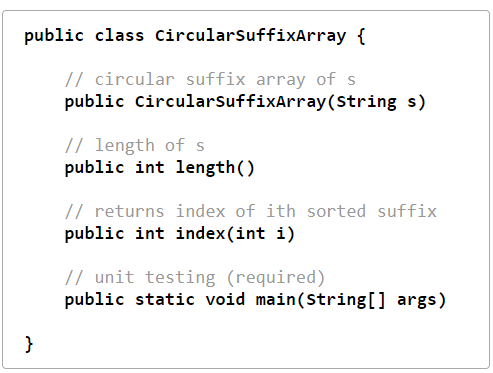
***Burrows–Wheeler inverse transform.*** Now, we describe how to invert the Burrows–Wheeler transform and recover the original input string. If the *jth* original suffix (original string, shifted *j* characters to the left) is the *ith* row in the sorted order, we define next[i] to be the row in the sorted order where the (*j* + 1)*st* original suffix appears. For example, if first is the row in which the original input string appears, then next[first] is the row in the sorted order where the 1*st* original suffix (the original string left-shifted by 1) appears; next[next[first]] is the row in the sorted order where the 2*nd* original suffix appears; next[next[next[first]]] is the row where the 3*rd* original suffix appears; and so forth.

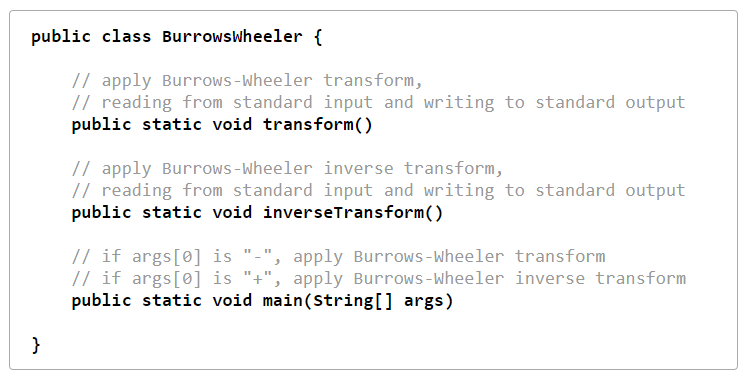
***Move-to-front encoding and decoding.*** *The main idea of move-to-front encoding is to maintain an ordered sequence of the characters in the alphabet by repeatedly reading a character from the input message; printing the position in the sequence in which that character appears; and moving that character to the front of the sequence. As a simple example, if the initial ordering over a 6-character alphabet is A B C D E F, and we want to encode the input CAAABCCCACCF.*

***Circular suffix array.*** *To efficiently implement the key component in the Burrows–Wheeler transform, you will use a fundamental data structure known as the circular suffix array, which describes the abstraction of a sorted array of the n circular suffixes of a string of length n. As an example, consider the string "ABRACADABRA!" of length 12. The table below shows its 12 circular suffixes and the result of sorting them.*

**3.API Implemented**







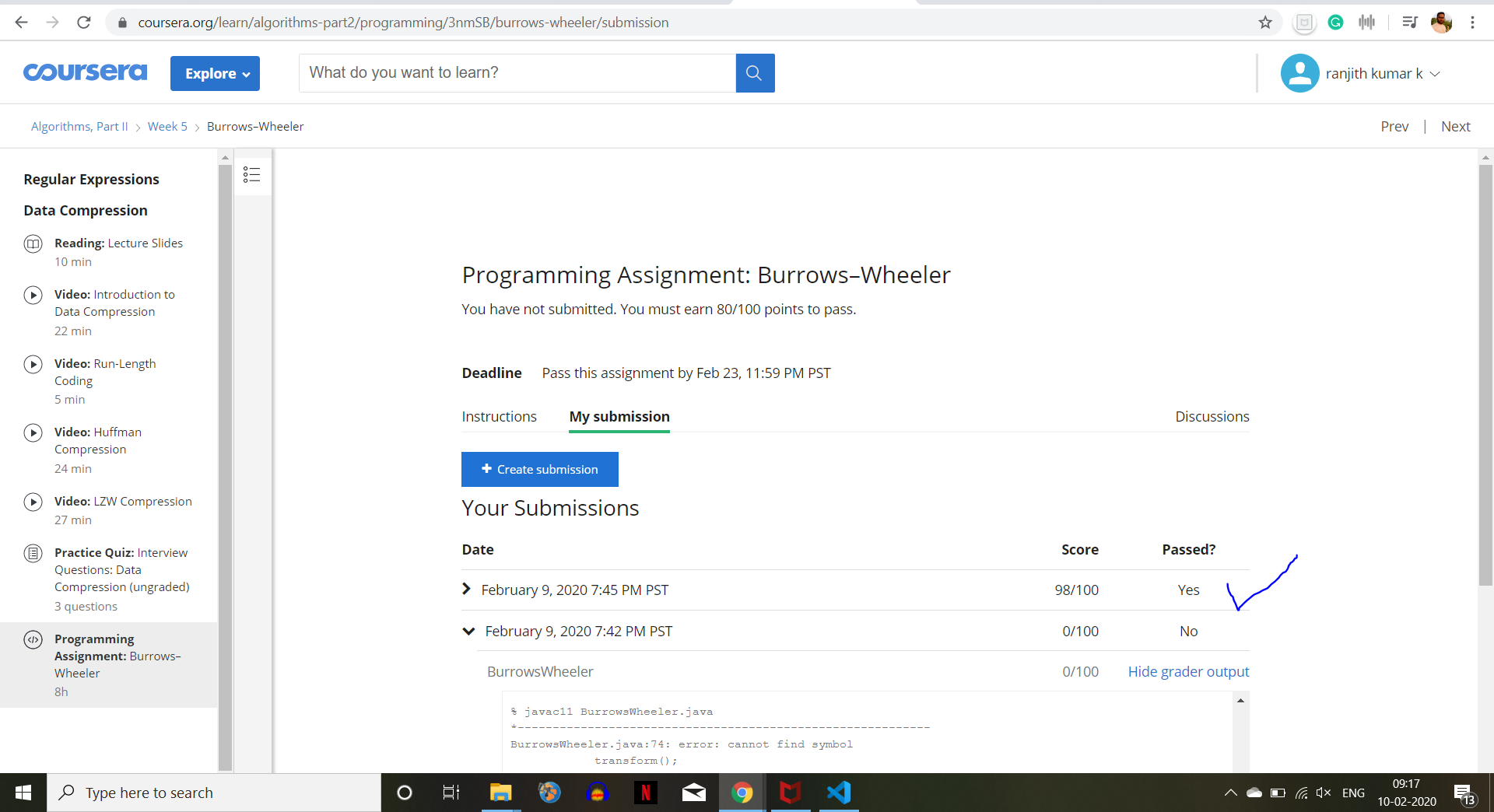
**4.Web Submission**

The project completed and submitted in the coursera for gardening.

All test cases passed obtaining 98% score overall.

The project files are submitted in the git and LMS last module along with the report.

The project git submission also been attached in the reference section below.



**REFERENCES:**

<https://github.com/ranjith210297/ADS-2_2019501102/tree/master/Project-4(BurrowsWheeler)>

<https://coursera.cs.princeton.edu/algs4/assignments/seam/specification.php>