Laboratory work #4

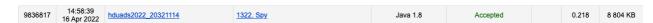
Student: HU Riqian Student ID: 20321114

Timus Name: hduads2022_20321114

Mail: jhlxhrq@163.com

Problem # 1322. Spy

Screenshot from Timus:



Explanation of algorithm:

- 1. We use the Burrows-Wheeler inverse transformation algorithm.
- 2. In the BWT, we use the Counting Sort to sort the list of the elements.

Computational complexity of algorithm:

Time Complexity: $O(N^3 \log N)$ Space Complexity: $O(N^2)$

Source code:

```
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Spy {
    public static void main(String[] args) {
        new Spy().run();
    }

    Scanner in;
    PrintWriter out;
    static int INDEX;
    static char[] CODES;

    int nextInt() {
        return Integer.parseInt(in.nextLine());
    }

    String nextLine() {
        return in.nextLine();
    }
}
```

```
void run() {
        in = new Scanner(System.in);
        out = new PrintWriter(System.out);
        solve();
        out.flush();
    void solve() {
        INDEX = nextInt();
        String codes = nextLine();
        int size = codes.length();
        CODES = codes.toCharArray();
        NODES = new Node[size + 1];
        for (int i = 0; i < size; i++) {</pre>
            Node temp = new Node(i, CODES[i]);
            NODES[i] = temp;
        }
        Arrays.sort(NODES, 0, size, Node::compare);
        int x = INDEX - 1;
        for (int i = 0; i < size; i++) {</pre>
            x = NODES[x].getNext();
            out.print(CODES[x]);
    }
class Node {
    private final int nextIndex;
    private final char c;
    public Node(int nextIndex, char c) {
        this.nextIndex = nextIndex;
        this.c = c;
    public int getNext() {
        return nextIndex;
    public char getChar() {
        return c;
    static int compare(Node n1, Node n2) {
        if (n1.getChar() < n2.getChar() || n1.getChar() == n2.getChar() &&</pre>
n1.getNext() < n2.getNext()) {</pre>
            return -1;
        } else {
            return 1;
   }
```