GOA COLLEGE OF ENGINEERING

"Bhausaheb Bandodkar Technical Education Complex"

Experiment No: 5

Aim: To implement Run Length technique

Theory:

Run-length encoding (RLE) is a form of lossless data compression in which runs of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. A stream of data is given as the input (i.e. "AAABBCCCC") and the output is a sequence of counts of consecutive data values in a row (i.e. "3A2B4C"). This type of data compression is lossless, meaning that when decompressed, all of the original data will be recovered when decoded. Its simplicity in both the encoding (compression) and decoding (decompression) is one of the most attractive features of the algorithm.

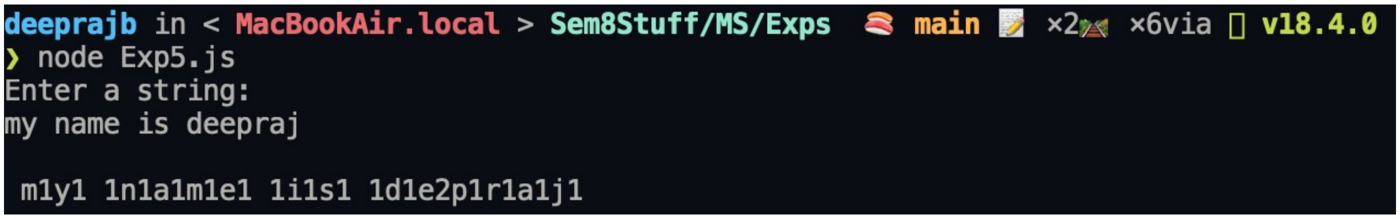
RLE works by reducing the physical size of a repeating string of characters. This repeat- ing string, called a run, is typically encoded into two bytes. The first byte represents the number of characters in the run and is called the run count. In practice, an encoded run may contain 1 to 128 or 256 characters; the run count usually contains as the number of characters minus one (a value in the range of 0 to 127 or 255). The second byte is the value of the character in the run, which is in the range of 0 to 255, and is called the run value.

It is supported by most bitmap file formats, such as TIFF, BMP, and PCX. RLE is suited for compressing any type of data regardless of its information content, but the content of the data will affect the compression ratio achieved by RLE. Although most RLE algorithms cannot achieve the high compression ratios of the more advanced compression methods, RLE is both easy to implement and quick to execute, making it a good alternative to either using a complex

Code:

```
const readline = require("readline").createInterface({
input: process.stdin,
output: process.stdout
})
function printRLE(str) {
let result = ""
let n = str.length;
for (let i = 0; i < n; i++) {
// Count occurrences of current character
let count = 1;
while (i < n - 1 && str[i] == str[i + 1]) {
count++;
i++; }
// character and its count
result += str[i] + count
}
console.log("\n",result)
}
readline.question("Enter a string:\n", str => {
printRLE(str);
readline.close()
})
```

Output:



```
deeprajb in < MacBookAir.local > Sem8Stuff/MS/Exps  🍣 main 📝 ×2 ×6via  v18.4.0
❯ node Exp5.js
Enter a string:
my name is deepraj

 m1y1 1n1a1m1e1 1i1s1 1d1e2p1r1a1j1
```

Conclusion: A program implementing Run Length technique was successfully implemented and executed.