Zachary Banducci

April 2, 2018

CSE 3353

Project 2

**Running instructions:**

encoder: *node encoder.js <file\_name>*

(will output the encoded file to a file named encoded.txt)

decoder: *node decoder.js <file\_name> <file\_dest(optional)>*

**Random generation**

[ { character: 'T', frequency: 3529 },

{ character: 'd', frequency: 3452 },

{ character: 'C', frequency: 3429 },

{ character: 'c', frequency: 3463 },

{ character: 'e', frequency: 3562 },

{ character: ' ', frequency: 27204 },

{ character: 'v', frequency: 3540 },

{ character: 'L', frequency: 3546 },

{ character: 'G', frequency: 3429 },

{ character: 'D', frequency: 3540 },

{ character: 'N', frequency: 3474 },

{ character: '?', frequency: 4252 },

{ character: 'P', frequency: 3467 },

{ character: 'M', frequency: 3507 },

{ character: 'h', frequency: 3464 },

{ character: 'X', frequency: 3507 },

{ character: 'I', frequency: 3500 },

{ character: 'U', frequency: 3517 },

{ character: 'n', frequency: 3456 },

{ character: 'A', frequency: 3527 },

{ character: 'l', frequency: 3470 },

{ character: '\n', frequency: 2796 },

{ character: 's', frequency: 3428 },

{ character: 'j', frequency: 3545 },

{ character: 'S', frequency: 3455 },

{ character: '!', frequency: 5798 },

{ character: 'J', frequency: 3411 },

{ character: 'g', frequency: 3543 },

{ character: 'm', frequency: 3351 },

{ character: 'B', frequency: 3430 },

{ character: 'u', frequency: 3476 },

{ character: 't', frequency: 3474 },

{ character: 'f', frequency: 3582 },

{ character: 'R', frequency: 3495 },

{ character: 'r', frequency: 3521 },

{ character: 'V', frequency: 3479 },

{ character: 'O', frequency: 3479 },

{ character: 'o', frequency: 3504 },

{ character: 'i', frequency: 3356 },

{ character: 'p', frequency: 3443 },

{ character: 'F', frequency: 3468 },

{ character: 'q', frequency: 3384 },

{ character: 'W', frequency: 3548 },

{ character: 'x', frequency: 3360 },

{ character: 'Q', frequency: 3553 },

{ character: '.', frequency: 2949 },

{ character: 'b', frequency: 3462 },

{ character: 'k', frequency: 3516 },

{ character: 'a', frequency: 3506 },

{ character: 'Z', frequency: 3463 },

{ character: 'E', frequency: 3430 },

{ character: 'H', frequency: 3449 },

{ character: 'K', frequency: 3500 },

{ character: 'Y', frequency: 3438 },

{ character: 'y', frequency: 3496 },

{ character: 'w', frequency: 3444 },

{ character: 'z', frequency: 3452 } ]

{ '!': '00000',

m: '000010',

i: '000011',

x: '000100',

q: '000101',

J: '000110',

s: '000111',

' ': '001',

C: '010000',

G: '010001',

B: '010010',

E: '010011',

Y: '010100',

p: '010101',

w: '010110',

H: '010111',

z: '011000',

d: '011001',

S: '011010',

n: '011011',

b: '011100',

c: '011101',

Z: '011110',

h: '011111',

P: '100000',

F: '100001',

l: '100010',

t: '100011',

N: '100100',

u: '100101',

O: '100110',

V: '100111',

R: '101000',

y: '101001',

K: '101010',

I: '101011',

o: '101100',

a: '101101',

X: '101110',

M: '101111',

k: '110000',

U: '110001',

r: '110010',

A: '110011',

T: '110100',

D: '110101',

v: '110110',

g: '110111',

j: '111000',

L: '111001',

W: '111010',

Q: '111011',

e: '111100',

f: '111101',

'?': '111110',

'\n': '1111110',

'.': '1111111' }

uncompressed file length:

body: 223819

compressed file length:

characters: 57

tree: 171

body: 1261249

Compression ratio:

Since there were many unique characters within the file, the tree size needed to be larger in order to accommodate the necessary bits to represent the data. As we will see, by decreasing the unique characters within the file, the compression ratio will begin to decrease.

|  |  |  |  |
| --- | --- | --- | --- |
| Most frequent | | Least frequent | |
| Character | Frequency | Character | Frequency |
| “ “ | 27204 | “x” | 3371 |
| “!” | 5798 | “s” | 3358 |
| “?” | 4252 | “z” | 3354 |
| “q” | 3601 | “\n” | 2796 |
| “f” | 3577 | “.” | 2949 |

**Decreased Character Pool**

[ { character: 'c', frequency: 30252 },

{ character: 'e', frequency: 30180 },

{ character: 'd', frequency: 30085 },

{ character: '.', frequency: 2778 },

{ character: ' ', frequency: 27080 },

{ character: 'f', frequency: 29711 },

{ character: 'a', frequency: 29981 },

{ character: 'b', frequency: 30110 },

{ character: '\n', frequency: 2920 },

{ character: '?', frequency: 4193 },

{ character: '!', frequency: 5726 } ]

{ '!': '0000',

'?': '00010',

'.': '000110',

'\n': '000111',

' ': '001',

f: '010',

a: '011',

d: '100',

b: '101',

e: '110',

c: '111' }

uncompressed file length:

body: 223016

compressed file length:

characters: 11

tree: 33

body: 700254

Compression ratio:

By decreasing the amount of characters that need to be represented, a smaller number of bits is needed in order to represent the data when the file is encoded.

**Weighted Data Pool**

[ { character: 'e', frequency: 5229 },

{ character: ' ', frequency: 5323 },

{ character: 'c', frequency: 3276 },

{ character: 'g', frequency: 7542 },

{ character: 'f', frequency: 6390 },

{ character: 'd', frequency: 4403 },

{ character: 'a', frequency: 1043 },

{ character: '.', frequency: 127 },

{ character: 'b', frequency: 2117 },

{ character: '\n', frequency: 130 },

{ character: '!', frequency: 119 },

{ character: '?', frequency: 118 } ]

{ c: '000',

'?': '0010000',

'!': '0010001',

'.': '0010010',

'\n': '0010011',

a: '00101',

b: '0011',

g: '01',

d: '100',

e: '101',

' ': '110',

f: '111' }

uncompressed file length:

body: 35817

compressed file length:

characters: 12

tree: 36

body: 106088

Compression ratio:

**Recap**

One major difficulty was determining the best way to structure the various components of the modules in order to increase efficiency and clarity. Another difficult thing was that it became difficult to encode the tree back into the encoded file in an efficient manner. I struggled with how I could conserve the most space in the file. I managed to overcome my hurdle and was able to encode my Huffman tree in a minimal number of bytes;