For the purposes of loading a dynamic huffman tree, this is a png file

that uses dynamic and is all red. The final values should be (255, 0, 0) and

is 50x50

/\*

0000024: xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  CIDATx

000002a: xxxxxxxx 11101101 11001111 00110001 00010001 00000000  ...1..

0000030: 00000000 00001000 00000000 10100001 11101111 01011111  .....\_

0000036: 01011010 00110011 10111000 01111010 00001100 00000100  Z3.z..

000003c: 10100000 10101001 11111001 00100000 00010001 00010001  ... ..

0000042: 00010001 00010001 00010001 00010001 00010001 00010001  ......

0000048: 00010001 00010001 00010001 00010001 00010001 00010001  ......

000004e: 00010001 00010001 00010001 00010001 00010001 00010001  ......

0000054: 00010001 00010001 00010001 00010001 00010001 00010001  ......

000005a: 00010001 00010001 00010001 00010001 00010001 00010001  ......

0000060: 00010001 00010001 00010001 00010001 00010001 10010001  ......

0000066: 10001011 00000101 10110000 00110011 01110101 10010110  ...3u.

000006c: 01111001 11000101 00011100 10110001 00000000 00000000  y.....

0000072: 00000000 00000000 01001001 01000101 01001110 01000100  ..

\*/

1 - Last block

10 - Dynamic

11101 - HLIT, ??? + 257 = 29 + 257 = 286

01111 - HDIST, ??? + 1 = 15 + 1 = 16

1110 - HCLEN, (??? + 4) \* 3 = (14 + 4) \* 3 = (18)\*3 = 54

///Read 3 bits per each

///Values appear in this order

16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15

16 – 000 – 0

17 – 011 – 3

18 – 010 – 2

0 – 100 – 4

8 – 000 - 0

7 – 000 – 0

9 – 000 – 0

6 – 000 – 0

10 – 000 – 0

5 – 000 - 0

11 – 000 – 0

4 – 010 – 2

12 – 000 – 0

3 – 000 – 0

13 – 000 – 0

2 – 100 – 4

14 – 000 – 0

1 – 010 – 2

15 – 0 (Because we only have 54 bits to read and the rest are zeroes) Not 111

15 – 111 – 7 (Incorrect)

What we should get

Value->Code Length Below

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 4 | 2 | 4 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 |

[0:4, 1:2, 2:4, 3:0, 4:2, 5:0, 6:0, 7:0, 8:0, 9:0, 10:0, 11:0, 12:0, 13:0, 14:0, 15:0, 16:0, 17:3, 18:2]

Count of the code length. (How many times does a code length occur)

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 0 | 3 | 1 | 2 |

//Do canonical Huffman coding

00

01

10

110

1110

1111

//Min codes for reference

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 0 | 0 | 4 | 10 |

TABLE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 4 | 18 | 17 | 0 | 2 |
| 00 | 01 | 10 | 110 | 1110 | 1111 |

0 - 15: Represent code lengths of 0 - 15

16: Copy the previous code length 3 - 6 times.

The next 2 bits indicate repeat length

(0 = 3, ... , 3 = 6)

Example: Codes 8, 16 (+2 bits 11),

16 (+2 bits 10) will expand to

12 code lengths of 8 (1 + 6 + 5)

17: Repeat a code length of 0 for 3 - 10 times.

(3 bits of length)

18: Repeat a code length of 0 for 11 - 138 times

(7 bits of length)

1111 – 2

10 –

1111 –

1111 –

01 –

00 –

10 –

110 –

10 –

110 –

01 –

10 –

00 -

//Testing using the example on the deflate specifications.

//ABCDEFGH with bit lengths (3, 3, 3, 3, 3, 2, 4, 4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 0 | 0 | 1 | 5 | 2 |

//Treat it like canonical Huffman coding to generate the codes. It should work the same exact way

2, 3, 3, 3, 3, 3, 4, 4

00

010

011

100

101

110

1110

1111

In Order of appearance in the list

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H |
| 010 | 011 | 100 | 101 | 110 | 00 | 1110 | 1111 |

11101101 11001111 00110001 00010001 00000000 00000000 00001000 00000000 10100001 11101111 01011111 01011010 00110011 10111000 01111010 00001100 00000100 10100000 10101001 11111001 00100000 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 00010001 10010001 10001011 00000101 10110000 00110011 01110101 10010110 01111001 11000101 00011100 10110001 00000000 00000000 00000000 00000000 01001001 01000101 01001110 01000100

1110110111001111001100010001000100000000000000000000100000000000101000011110111101011111010110100011001110111000011110100000110000000100101000001010100111111001001000000001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000110010001100010110000010110110000001100110111010110010110011110011100010100011100101100010000000000000000000000000000000001001001010001010100111001000100