huffman.py

```
import numpy as np

MAX_CLEN = 32 # assumed maximum initial code length

def getFreq(data):
    freq = [0] * 257
    for elem in data:
        freq[elem] += 1
    freq[256] = 1
    return freq

def jpegGenerateOptimalTable(freq):
    bits = [0] * (MAX_CLEN + 1)
    bitPos = [0] * (MAX_CLEN + 1)
    codesize = [0] * 257
    nzIndex = [0] * 257

    others = [-1] * 257

    numNzSymbols = 0
    for i in range(257):
        if freq[i]:
            nzIndex[numNzSymbols] = i
            freq[numNzSymbols] = freq[i]
            numNzSymbols += 1

    huffval = [0] * (numNzSymbols - 1)

    while True:
        c1 = -1
        c2 = -1
        v = 1000000000
        v2 = 1000000000
        for i in range(numNzSymbols):
            if freq[i] <= v2:
                if freq[i] <= v:
                    c2 = c1
```

```
                    v2 = v
                    v = freq[i]
                    c1 = i
                else:
                    v2 = freq[i]
                    c2 = i

        if (c2 < 0):
            break

        freq[c1] += freq[c2]
        freq[c2] = 1000000001

        codesize[c1] += 1
        while others[c1] >= 0:
            c1 = others[c1]
            codesize[c1] += 1

        others[c1] = c2

        codesize[c2] += 1
        while others[c2] >= 0:
            c2 = others[c2]
            codesize[c2] += 1

    for i in range(numNzSymbols):
        bits[codesize[i]] += 1

    p = 0
    for i in range(1, MAX_CLEN + 1):
        bitPos[i] = p
        p += bits[i]

    for i in range(MAX_CLEN, 16, -1):
        while bits[i] > 0:
            j = i - 2
            while bits[j] == 0:
                j -= 1
            bits[i] -= 2
            bits[i - 1] += 1
            bits[j + 1] += 2
```

```
79              bits[j] -= 1
80
81      i = MAX_CLEN
82      while bits[i] == 0:
83          i -= 1
84      bits[i] -= 1
85
86      for i in range(numNzSymbols - 1):
87          huffval[bitPos[codesize[i]]] = nzIndex[i]
88          bitPos[codesize[i]] += 1
89
90      return bits, huffval
91
92  def jpegGenerateHuffmanTable(bits, huffval):
93      huffsize = [0] * 257
94      huffcode = [0] * 257
95
96      p = 0
97      for l in range(1, 17):
98          i = bits[l]
99          while i:
100             huffsize[p] = l
101             p += 1
102             i -= 1
103
104     huffsize[p] = 0
105     lastp = p
106
107     code = 0
108     si = huffsize[0]
109     p = 0
110     while huffsize[p]:
111         while huffsize[p] == si:
112             huffcode[p] = code
113             code += 1
114             p += 1
115         code <<= 1
116         si += 1
117
118     ehufco = [0] * 257
119     ehufsi = [0] * 257
```

```
120
121     for p in range(lastp):
122         i = huffval[p]
123         ehufco[i] = huffcode[p]
124         ehufsi[i] = huffsize[p]
125
126     return ehufsi, ehufco
127
128 def jpegTransformTable(ehufsi, ehufco):
129     table = {}
130     for i in range(len(ehufco)):
131         if ehufsi[i] != 0:
132             endCode = bin(ehufco[i])[2:]
133             nbZeros = ehufsi[i] - len(endCode)
134             table[i] = '0' * nbZeros + endCode
135     return table
136
137 def jpegCreateHuffmanTable(arr):
138     freq = getFreq(arr)
139     bits, huffval = jpegGenerateOptimalTable(freq)
140     ehufsi, ehufco = jpegGenerateHuffmanTable(bits,
    huffval)
141     table = jpegTransformTable(ehufsi, ehufco)
142     return table
143
144 def convert_huffman_table(table):
145     """convert huffman table to count and weigh"""
146     # table[int] = string
147     pairs = sorted(table.items(), key=lambda x:
    (len(x[1]), x[1]))
148     weigh, codes = zip(*pairs)
149     weigh = np.array(weigh, dtype=np.uint8)
150     # count[i]: there are count[i] codes of length
    i+1
151     count = np.zeros(16, dtype=np.uint8)
152     for c in codes:
153         count[len(c)-1] += 1
154     return count, weigh
155
156
157 def read_huffman_code(table, stream):
158     prefix = ''
```

```
159     while prefix not in table:
160         prefix += str(stream.read_bit())
161     return table[prefix]
162 
163 
164 def reverse(table):
165     return {v: k for k, v in table.items()}
166 
167 # 4 recommended huffman tables in JPEG standard
168 # luminance DC
169 RM_Y_DC = {'00': 0, '010': 1, '011': 2, '100': 3,
    '101': 4, '110': 5,
170            '1110': 6, '11110': 7, '111110': 8,
    '1111110': 9, '11111110': 10,
171            '111111110': 11}
172 
173 # luminance AC
174 RM_Y_AC = {'00': 1, '01': 2, '100': 3, '1010': 0,
    '1011': 4, '1100': 17,
175            '11010': 5, '11011': 18, '11100': 33,
    '111010': 49, '111011': 65,
176            '1111000': 6, '1111001': 19, '1111010':
    81, '1111011': 97,
177            '11111000': 7, '11111001': 34, '11111010':
    113, '111110110': 20,
178            '111110111': 50, '111111000': 129,
    '111111001': 145,
179            '111111010': 161, '1111110110': 8,
    '1111110111': 35,
180            '1111111000': 66, '1111111001': 177,
    '1111111010': 193,
181            '11111110110': 21, '11111110111': 82,
    '11111111000': 209,
182            '11111111001': 240, '111111110100': 36,
    '111111110101': 51,
183            '111111110110': 98, '111111110111': 114,
    '111111111000000': 130,
184            '1111111110000010': 9, '1111111110000011':
    10,
185            '1111111110000100': 22,
    '1111111110000101': 23,
186            '1111111110000110': 24,
    '1111111110000111': 25,
187            '1111111110001000': 26,
    '1111111110001001': 37,
```

```
188             '1111111110001010': 38,
    '1111111110001011': 39,
189             '1111111110001100': 40,
    '1111111110001101': 41,
190             '1111111110001110': 42,
    '1111111110001111': 52,
191             '1111111110010000': 53,
    '1111111110010001': 54,
192             '1111111110010010': 55,
    '1111111110010011': 56,
193             '1111111110010100': 57,
    '1111111110010101': 58,
194             '1111111110010110': 67,
    '1111111110010111': 68,
195             '1111111110011000': 69,
    '1111111110011001': 70,
196             '1111111110011010': 71,
    '1111111110011011': 72,
197             '1111111110011100': 73,
    '1111111110011101': 74,
198             '1111111110011110': 83,
    '1111111110011111': 84,
199             '1111111110100000': 85,
    '1111111110100001': 86,
200             '1111111110100010': 87,
    '1111111110100011': 88,
201             '1111111110100100': 89,
    '1111111110100101': 90,
202             '1111111110100110': 99,
    '1111111110100111': 100,
203             '1111111110101000': 101,
    '1111111110101001': 102,
204             '1111111110101010': 103,
    '1111111110101011': 104,
205             '1111111110101100': 105,
    '1111111110101101': 106,
206             '1111111110101110': 115,
    '1111111110101111': 116,
207             '1111111110110000': 117,
    '1111111110110001': 118,
208             '1111111110110010': 119,
    '1111111110110011': 120,
209             '1111111110110100': 121,
    '1111111110110101': 122,
210             '1111111110110110': 131,
    '1111111110110111': 132,
```

```
211             '1111111110111000': 133,
    '1111111110111001': 134,
212             '1111111110111010': 135,
    '1111111110111011': 136,
213             '1111111110111100': 137,
    '1111111110111101': 138,
214             '1111111110111110': 146,
    '1111111110111111': 147,
215             '1111111111000000': 148,
    '1111111111000001': 149,
216             '1111111111000010': 150,
    '1111111111000011': 151,
217             '1111111111000100': 152,
    '1111111111000101': 153,
218             '1111111111000110': 154,
    '1111111111000111': 162,
219             '1111111111001000': 163,
    '1111111111001001': 164,
220             '1111111111001010': 165,
    '1111111111001011': 166,
221             '1111111111001100': 167,
    '1111111111001101': 168,
222             '1111111111001110': 169,
    '1111111111001111': 170,
223             '1111111111010000': 178,
    '1111111111010001': 179,
224             '1111111111010010': 180,
    '1111111111010011': 181,
225             '1111111111010100': 182,
    '1111111111010101': 183,
226             '1111111111010110': 184,
    '1111111111010111': 185,
227             '1111111111011000': 186,
    '1111111111011001': 194,
228             '1111111111011010': 195,
    '1111111111011011': 196,
229             '1111111111011100': 197,
    '1111111111011101': 198,
230             '1111111111011110': 199,
    '1111111111011111': 200,
231             '1111111111100000': 201,
    '1111111111100001': 202,
232             '1111111111100010': 210,
    '1111111111100011': 211,
233             '1111111111100100': 212,
    '1111111111100101': 213,
```

```
234             '1111111111100110': 214, '1111111111100111': 215,
235             '1111111111101000': 216, '1111111111101001': 217,
236             '1111111111101010': 218, '1111111111101011': 225,
237             '1111111111101100': 226, '1111111111101101': 227,
238             '1111111111101110': 228, '1111111111101111': 229,
239             '1111111111110000': 230, '1111111111110001': 231,
240             '1111111111110010': 232, '1111111111110011': 233,
241             '1111111111110100': 234, '1111111111110101': 241,
242             '1111111111110110': 242, '1111111111110111': 243,
243             '1111111111111000': 244, '1111111111111001': 245,
244             '1111111111111010': 246, '1111111111111011': 247,
245             '1111111111111100': 248, '1111111111111101': 249,
246             '1111111111111110': 250}
247
248 # chroma DC
249 RM_C_DC = {'00': 0, '01': 1, '10': 2, '110': 3, '1110': 4, '11110': 5,
250             '111110': 6, '1111110': 7, '11111110': 8, '111111110': 9,
251             '1111111110': 10, '11111111110': 11}
252
253 # chroma AC
254 RM_C_AC = {'00': 0, '01': 1, '100': 2, '1010': 3, '1011': 17, '11000': 4,
255             '11001': 5, '11010': 33, '11011': 49, '111000': 6, '111001': 18,
256             '111010': 65, '111011': 81, '1111000': 7, '1111001': 97,
257             '1111010': 113, '11110110': 19, '11110111': 34, '11111000': 50,
258             '11111001': 129, '111110100': 8, '111110101': 20, '111110110': 66,
259             '111110111': 145, '111111000': 161, '111111001': 177,
```

```
260             '111111010': 193, '1111110110': 9,
'1111110111': 35,
261             '1111111000': 51, '1111111001': 82,
'1111111010': 240,
262             '11111110110': 21, '11111110111': 98,
'11111111000': 114,
263             '11111111001': 209, '111111110100': 10,
'111111110101': 22,
264             '111111110110': 36, '111111110111': 52,
'11111111100000': 225,
265             '111111111000010': 37, '111111111000011':
241,
266             '1111111110001000': 23,
'1111111110001001': 24,
267             '1111111110001010': 25,
'1111111110001011': 26,
268             '1111111110001100': 38,
'1111111110001101': 39,
269             '1111111110001110': 40,
'1111111110001111': 41,
270             '1111111110010000': 42,
'1111111110010001': 53,
271             '1111111110010010': 54,
'1111111110010011': 55,
272             '1111111110010100': 56,
'1111111110010101': 57,
273             '1111111110010110': 58,
'1111111110010111': 67,
274             '1111111110011000': 68,
'1111111110011001': 69,
275             '1111111110011010': 70,
'1111111110011011': 71,
276             '1111111110011100': 72,
'1111111110011101': 73,
277             '1111111110011110': 74,
'1111111110011111': 83,
278             '1111111110100000': 84,
'1111111110100001': 85,
279             '1111111110100010': 86,
'1111111110100011': 87,
280             '1111111110100100': 88,
'1111111110100101': 89,
281             '1111111110100110': 90,
'1111111110100111': 99,
282             '1111111110101000': 100,
'1111111110101001': 101,
```

```
283             '1111111110101010': 102,
    '1111111110101011': 103,
284             '1111111110101100': 104,
    '1111111110101101': 105,
285             '1111111110101110': 106,
    '1111111110101111': 115,
286             '1111111110110000': 116,
    '1111111110110001': 117,
287             '1111111110110010': 118,
    '1111111110110011': 119,
288             '1111111110110100': 120,
    '1111111110110101': 121,
289             '1111111110110110': 122,
    '1111111110110111': 130,
290             '1111111110111000': 131,
    '1111111110111001': 132,
291             '1111111110111010': 133,
    '1111111110111011': 134,
292             '1111111110111100': 135,
    '1111111110111101': 136,
293             '1111111110111110': 137,
    '1111111110111111': 138,
294             '1111111111000000': 146,
    '1111111111000001': 147,
295             '1111111111000010': 148,
    '1111111111000011': 149,
296             '1111111111000100': 150,
    '1111111111000101': 151,
297             '1111111111000110': 152,
    '1111111111000111': 153,
298             '1111111111001000': 154,
    '1111111111001001': 162,
299             '1111111111001010': 163,
    '1111111111001011': 164,
300             '1111111111001100': 165,
    '1111111111001101': 166,
301             '1111111111001110': 167,
    '1111111111001111': 168,
302             '1111111111010000': 169,
    '1111111111010001': 170,
303             '1111111111010010': 178,
    '1111111111010011': 179,
304             '1111111111010100': 180,
    '1111111111010101': 181,
305             '1111111111010110': 182,
    '1111111111010111': 183,
```

```
            '1111111111011000': 184, '1111111111011001': 185,
            '1111111111011010': 186, '1111111111011011': 194,
            '1111111111011100': 195, '1111111111011101': 196,
            '1111111111011110': 197, '1111111111011111': 198,
            '1111111111100000': 199, '1111111111100001': 200,
            '1111111111100010': 201, '1111111111100011': 202,
            '1111111111100100': 210, '1111111111100101': 211,
            '1111111111100110': 212, '1111111111100111': 213,
            '1111111111101000': 214, '1111111111101001': 215,
            '1111111111101010': 216, '1111111111101011': 217,
            '1111111111101100': 218, '1111111111101101': 226,
            '1111111111101110': 227, '1111111111101111': 228,
            '1111111111110000': 229, '1111111111110001': 230,
            '1111111111110010': 231, '1111111111110011': 232,
            '1111111111110100': 233, '1111111111110101': 234,
            '1111111111110110': 242, '1111111111110111': 243,
            '1111111111111000': 244, '1111111111111001': 245,
            '1111111111111010': 246, '1111111111111011': 247,
            '1111111111111100': 248, '1111111111111101': 249,
            '1111111111111110': 250}

if __name__ == "__main__":
    arr = np.array([np.random.randint(-127, 128) for _ in range(64)])
    table = jpegCreateHuffmanTable(arr)
    print(table)
```