Huffman Report

Terminology

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code (sometimes called "prefix-free codes", that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol). Huffman coding is such a widespread method for creating prefix codes that the term "Huffman code" is widely used as a synonym for "prefix code".

Description of implementation:

1. Compression of Files/Folder

1)The program asks user for the file/folder name to be compressed after that the program check that the file/folder exist in **same directory**. if exists the program open the input file/folder and read the message (if folder the program read each file within folder and merge them in one string as whole message knowing the size of each message)

The construction algorithm uses a **priority queue** where the node with lowest frequency is given highest priority , where :

2) **Create** a **leaf node** for each symbol and add it to the heap.

3)While there is more than one node in the heap:

3.1)**Remove** the two nodes of highest priority **(lowest frequency)** from the min heap

3.2) **Create** a **new internal node** with these two nodes as children and with frequency equal to **the sum** of the two nodes' frequencies.

3.3)**Add** the new node to the heap.

4)**Traverse** tree and **make codes**

5)**encode** the message using this codes

6)**Add** padding

7)**return** compressed file/folder

2.Decompression Of File/Folder

1)The program asks user for the file/folder name to be decompressed after that the program check that the file/folder exist in **same directory**. if exists the program open the input file (if folder the program read file and un merge them then consider each of them as separate file) then:

1)**Read** headers from compressed file and convert them to **binary strings**

2)**Remove** padding

3)**Loop** through the **binary string** and check that bit is in the in reverse mapping of codes :

- if not in reverse mapping take 2 bits and check,
- if found return the char and start from third bit .....etc.

4)**return** decompressed File/Folder

Data structures used for implementing the encoding techniques:

Priority Queue/Min Heap(heapq library)/Dictionary(built-in python Hash-Maps)

The algorithms used and its complexity

Huffman :

heap data structures require $O(\log n)$ time per insertion, and a tree with n leaves has $2n-1$ nodes, **this algorithm operates *in $O(n \log n)$* time, where n is the number of symbols.**

Header format of Compression.

Compression Header Of File

- **First Byte** : Padding Size
- **Second Byte** : Indicator to know how many byte should be read to know the number of bytes the message written in (let it be n)
- **Third byte to n** : get the number of bytes should be read to get the message (let it be X)
- **Fourth byte to X** :message
- **X+1 to rest of file** :dictionary(Codes)

Compression Header Of Folder

- **First Byte** : Padding Size
- **Second Byte** : Number of files in folder (let it be n)
- **Third Byte to n*3** : Each file has three byte to know its length
- **(n*3)+1** : Indicator to know how many byte should be read to know the number of bytes the message written in (let it be x)
- **(n*3)+2 to x** : get the number of bytes should be read to get the message (let it be z)
- **Following byte to Z** :message
- **Z+1 to rest of file** :dictionary(Codes)

Sample run using given text file(test_huffman)

1)Ask user for compress/decompress file/folder

2)Ask for file name

```
$ python huffman.py
1)Compress
2)Decompress
1
1)CompressFile
2)CompressFolder
1
Please Enter File Name To Compress (Without Extention) : test_huffman
```

3)File get compressed with printing codes used to compress, compression rate ,running time

```
Please Enter File Name To Compress (Without Extention) : test_huffman
File Compressed
Compression Rate =  55.70608335309416 %
Char            Bytes           Code            NewCode
                32              100000          00
u               117             1110101         01000
b               98              1100010         010010
f               102             1100110         010011
n               110             1101110         0101
A               65              1000001         011000000
q               113             1110001         01100000100
C               67              1000011         01100000101
M               77              1001101         0110000011
H               72              1001000         011000010
:               58              111010          0110000110
O               79              1001111         0110000111
S               83              1010011         011000100
!               33              100001          0110001010
J               74              1001010         0110001011
V               86              1010110         0110001100000
X               88              1011000         0110001100001
*               42              101010          011000110001000
2               50              110010          0110001100010010
5               53              110101          0110001100010011
(               40              101000          01100011000101
)               41              101001          01100011000110
1               49              110001          0110001100011
G               71              1000111         01100011001
N               78              1001110         0110001101
W               87              1010111         011000111
c               99              1100011         011001
l               108             1101100         01101
a               97              1100001         0111
o               111             1101111         1000
,               44              101100          100100
I               73              1001001         1001010
.               46              101110          1001011
r               114             1110010         10011
t               116             1110100         1010
m               109             1101101         101100
y               121             1111001         101101
g               103             1100111         101110
'               39              100111          1011110
B               66              1000010         1011111000
F               70              1000110         101111100100
K               75              1001011         1011111001010
8               56              111000          1011111001011000O
9               57              111001          10111110010110001
Q               81              1010001         101111100101100l
0               48              110000          101111100101101
```

```
3                51              110011          1011111001011100
4                52              110100          10111110010111010
7                55              110111          10111110010111011
/                47              101111          1011111001011110
6                54              110110          10111110010111110
$                36              100100          101111100101111100
%                37              100101          101111100101111101000
#                35              100011          101111100101111101001
@                64              1000000         10111110010111110101
[                91              1011011         10111110010111110110
]                93              1011101         10111110010111110111
Z                90              1011010         1011111001011111111
R                82              1010010         10111110011
E                69              1000101         1011111010
j                106             1101010         1011111011
v                118             1110110         10111111
p                112             1110000         1100000
k                107             1101011         1100001

                10              1010            110001
i                105             1101001         11001
d                100             1100100         11010
s                115             1110011         11011
e                101             1100101         1110
P                80              1010000         11110000000
z                122             1111010         111100000010
U                85              1010101         111100000011
?                63              111111          1111000001
T                84              1010100         111100001
-                45              101101          11110001
;                59              111011          111100100
L                76              1001100         11110010100
x                120             1111000         11110010101
Y                89              1011001         11110010110
D                68              1000100         11110010111
"                34              100010          11110011
w                119             1110111         111101
h                104             1101000         11111
Time For Compress:  1.4615630999999993
```

Decompression

```
$ python huffman.py
1)Compress
2)Decompress
2
1)DecompressFile
2)DecompressFolder
1
Please Enter File Name To Decompress (Without Extention)  : test_huffman
File Decompressed
Time For Decompress:  2.357712499999999
```