

HUFFMAN CODING

Ms 140400147
SADIA YUNAS BUTT

Abstract

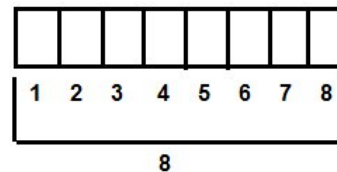
In 1952 David A. Huffman the student of MIT discovered this algorithm during work on his term paper assigned by his professor Robert M. Fano. The idea came into his mind that using a frequency sorted binary tree we can be able to find out the most efficient binary code [1]. Huffman denied the major flaw of Claude Shannon Fano coding by building the tree from bottom up instead of from the top down. There is a lot of compression algorithms but we focus on lossless compression so in this regard Huffman algorithm is so cool and efficient algorithm in sort of smallest application. Huffman is used for image compression for example PNG, JPG for simple picture of bird it also uses MPEG whenever you watch movie or video in computer chances are that its image will be in MPEG format both these PNG, MPEG and for text compression for example static and dynamic Huffman coding techniques

INTRODUCTION

Huffman coding is a great way to compress down and to be able to shrink it down by bites and chunks so give you example how it works write down on right an side the letter A B C D E if the document only have a range of these five letters and we want to do compress the data

Use of Huffman coding method to compress their content most of the time images and movies are also compress using lossless technique. so it is general technique which we often use in our computer like PKZIP in which WINZIP is one and other is GZIP these are use for general purpose algorithm which compress your text file and these use Huffman another application of Huffman is audio so there is MP3 and AAC format for audio whenever we use Ipad we use MP3 all these are encoded compress using Huffman there is lossy compression but there is lossless compression as well AAC is very widely use in U tube so rightly now we are using Huffman algorithm in your Iphone in Ipad these are use AAC. Huffman proof that compression can be easy using a frequency sorted binary tree in the bottom up fashion.

Popping could be the great way for doing it and the reason why? if we want

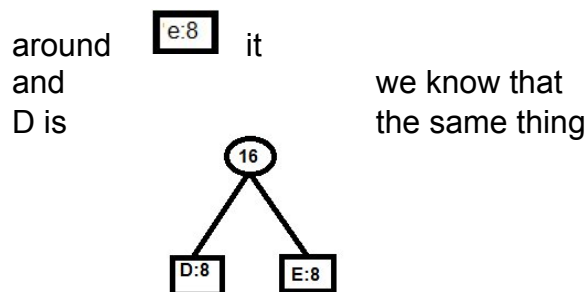


to stress the data in regular text format usually with store data you will after do like this you need set of your system

and each one would have to represent a bit is a two row at each -and so u say it you have to take a 4 bit for every letter but there is actually a lie u looking have been 8 bit for every letter but that is actually wrong, a better way for doing it and that was Huffman coding does and its only work on specific data so if you need 4 range data this is the something that would work for you but we can do actually is take care similar to something like this we can actually take care and stated acquiring two row 8 after you can do that just by using 3 bits and what can do have to do it in 3 form first then i am assure you have actual code soft works very first we

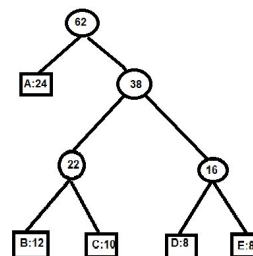
A	24
B	12
C	10
D	8
E	8

can do is . if you wondering when which actually is important certain strategy in computer science paring matrix that this is actually critical because you are doing with dynamic command of data and you need to be compress and Huffman coding is best way to doing that so the temporary tree generate is can be robed different then some the other we have seen as specifically can be different like binary tree because it cannot be use c associated with them and we can actually stood of with two lowest values we can give a E and we say it is equal to 8 e:8 and we draw a box



and we say they are equal 16

starting with the low value first so we can start of c:10 B:12 so they will be equal to 22 and now we connect it 22 and 16 it will make 38 and then we connect it to A :24 after sum up it root node contain which does not contain any node a sum of total data we looking for



why it will be better then ASCII

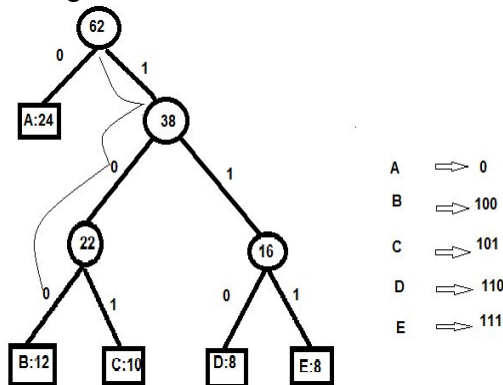
we will assign here the values 0 at the left hand side and 1 at the right hand side

we see the value 0 to A and if we see at the binary tree we will see loop here from root to B we will see binary value 100 and in the same way the all nodes get there values now we store all this in compress form

instead of 8 value here we give 1 and next we see three give value 8 same way do this all we are doing is the shrinking down the numbers of bits that required

if we look here is 5 time 8 so we need a total 40 as compare the new system Huffman coding

which is total 1+ 4 set of three so we looking 13



see adoptive Huffman coding how Huffman encode decode , image text compression finally we will discuss the conclusion and the future scope of it.

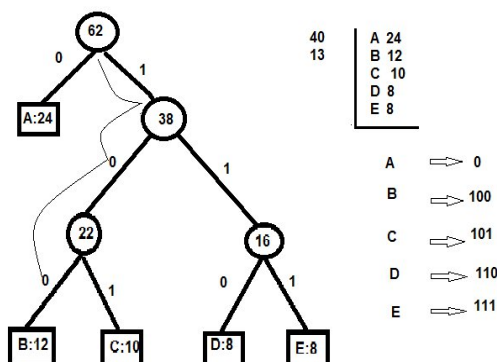
WHAT IS HUFFMAN CODING

There are many ways to store information. computer scientist always looking for new and better ways to store strings of data with as little space as possible. Huffman coding is a method of storing strings of data as binary code in an efficient manner .Huffman coding uses" lossless data compression" which means no information is lost which you are coded[3]

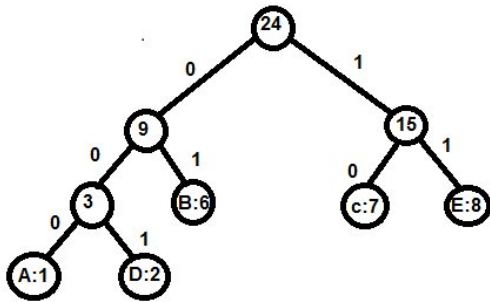
Huffman coding uses ' variable length [6] coding' which means that symbol in the data you are encoded or converted to binary symbol based on how often that symbol is used .for example if character 'a ' is used in your data a lot , the binary symbol representing it is shorter. If it is used rarely the symbol representing it is longer .this way all the data will be take less physical space when encoded. There is a way to decide what binary code to give each character using trees.

char	frequency
A	1
B	6
C	7
D	2
E	8

by Huffman coding we see well over half of the amount of space and that's one of the big reason that Huffman coding so powerful because by simply doing something by shrinking doing that are necessary on per letter base they much more for sequences or something where u can were the data sort like that



In this paper we may discuss first that what is Huffman coding and then how this algorithm works In 3 section we will



in this figure we have 5 different letter and also have frequency which often be use a and d in this example is the least , frequency represent how often character appear in a string of data now imagine these as 5 separate trees combine the two smallest tress in order we can combine them slowly bit by bit

These are what a ,b,c,d,e will each be converted to.

Huffman coding is to encode data to take up less space wouldn't it make sense to give some character binary code only 1 digit long ('0' or '1') or shouldn't more character be given code 2 digit long instead of 3? consider how you would read the code it is important for how each representation to be unique from each other such that you can easily tell which character that part is suppose to each represent .If '1' and '0' is representing a character any other representation containing 1 and 0 could be different character . now encode a,b,c,d,e using Huffman code so we will take each character in replace it with the binary

abcbe=000 01 1001 11-00001100111

decode 1011001000011101

we have to compare the representation above to the binary code bit by bit to fill

first we combine a and d which are smallest after combining them i have new tree with a greater frequency 3 now i will go for next least number which is here b:6 now i have new tree with greater frequency 9. next i have two more least character c and e and get the frequency 15 .finally i combine these two tress and get one large tree .now we have large tree containing all the characters we can now assign them a binary code to each symbol by going down the tree (each left branch receives a '0' and each right branch will receive '1' .if we go to the top we will get a binary string a=000,b=01 , c=10 , d=001 ,e=11

the only possible result

1011001000011101=c1001000011101(only c start with 1,then 0) so it must start with 'c'

after that it will be start with 'e'=c e 001000011101(only e start with 1 ,then 1)

then start with =c e d 000011101(only d start with 0 then 0 then 1)

= c d e a 011101(only a start with 0 then 0 then 0)

=c d e a b1101(only b start with 0 then 1)

=c d e a b e 01(only e start with 1 then 1)

=c d e a b e

b=cdeabeb

if we have to decode it you have to start either from back way front or front

suppose we have to accrue the average length via Huffman coding there is

special formula for that so average code word length = $1/f(T) \times \sum d(i) \times f(i)$, for $i=0$ to $i=n$

where n = number of character

$f(T)$ = total frequency

$f(i)$ = frequency of that character

$d(i)$ = length of that symbol

if we consider it in our example[2]

Average length

$$= (1/(1+6+7+2=8)) \times (3 \times 1 + 2 \times 6 + 2 \times 7 + 3 \times 2 + 2 \times 8)$$

$$= (1/24) \times (3 + 12 + 14 + 6 + 16)$$

$$= (1/24) \times (51)$$

e is encoded to 2.125 binary digit.

Adaptive Huffman coding

In adaptive Huffman coding we should remember two steps are . nodes should be increasing order and of the frequency and sum of the frequency in parent node greater than all the frequency of the side node .so taking these two facts in count it grow the Huffman tree using the adaptive Huffman coding [3],[4]

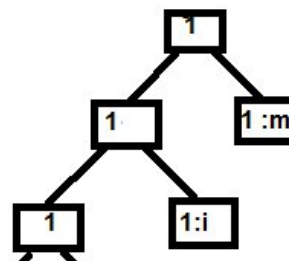
First of all in the for the string (MISSISSIPPI) we write an empty node always we first insert empty node after doing this we insert m

so we put two empty node here and put the frequency of m here that is m:1 we should always update the parent node so $1+0=1$ now we insert i so again we put two empty node and insert i in it and the frequency of i:1 and update

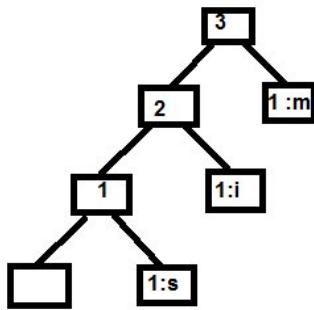
=2.125 digit long

(the average letter in the cod)[8]

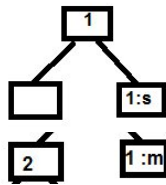
its parent node which was empty so after putting it becomes 1





now we can put s so we again make two empty node and insert s in the right node and insert the frequency of s which is s:1 and update its parent node we also update the i parent node which is now $1+1=2$

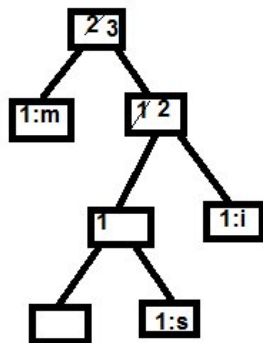


now we will see all left hand side nodes are greater than right hand side so in this case we will see that it is not in proper order so we will re-loop this



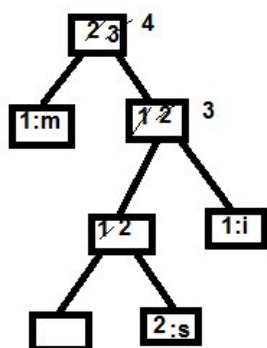
node( ) we go back and interchange this node .[5]

so in the string (MISSISSIPPI) MISS comes on right hand side and PPI go to left side) after interchange we will get



this tree

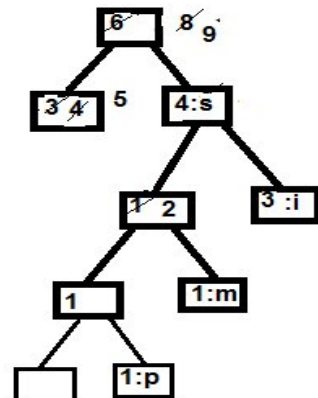
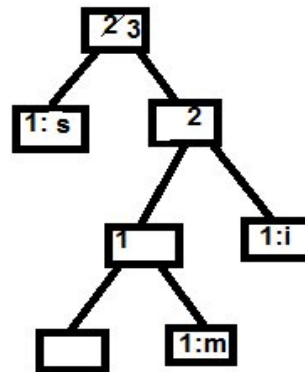
after each update in parent node we omit the previous frequency and write



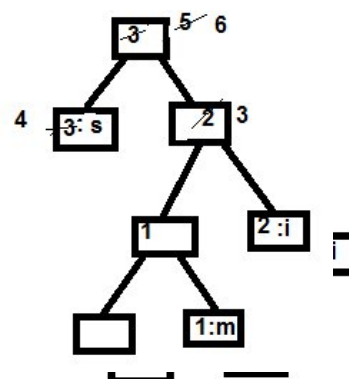
new after add in the previous.

now we have to add next s but there is already s so we just update the frequency but to do this we will see problem with updating see what will happened when we 1 more s in the previous s frequency

so we will interchange this 1:s with 1:m



so then we continue the operation s is already here so we update it and it becomes 2 now it look healthy tree so now we add new frequency of 1 here but there is already 1 so we just update the 1 so it will become 2 and if go to

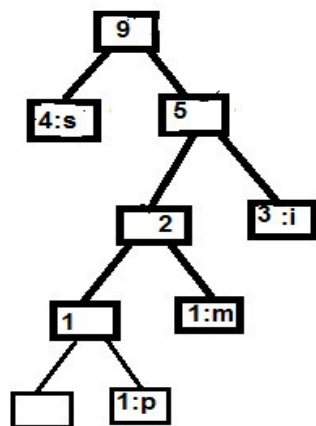


remaining parent node they all will add 1 more in them see

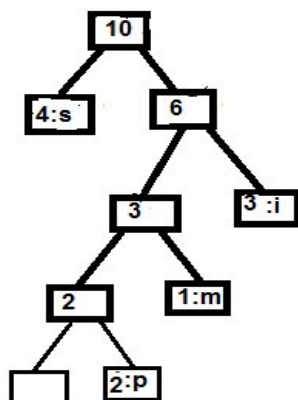
now we take next s from the string and update it ,now it becomes 3 now we do interchange these two nodes so s frequency will go to right side. So this is the update tree

next is again frequency i so we add new string it in it will be now 3 then we take string p there is no p that why we put this frequency in empty node so update here the parent node

but here we will see $5 > 4$ so we will interchange them with each other because they should be either less or equal to the right hand side.

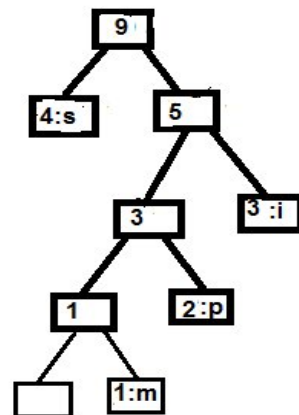


now we again insert next p and if we see here we have p so we update the frequency it become 2 and its parent node become $1+1=3$ and their



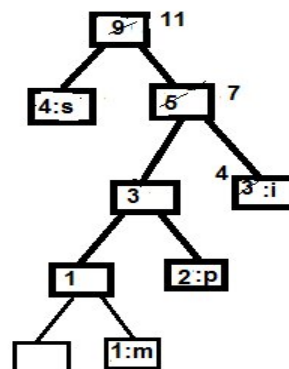
parent node will become $3+3=6$ and lastly $6+4=10$ at the root node

now if we look at the tree we will see it is again not a healthy tree because $2 > 1$ so we have to interchange it but now at this time we will interchange 1:m with 2:p



so due to this interchange whole tree will change the nodes frequency now it become healthy tree .now we add last character i we have i so we update its frequency it will become 4 then we update its parent node it will become

if we check the integrating of the tree it



is in increasing order so it is healthy tree all the parent node are greater then its child node

Huffman encode decode [2]

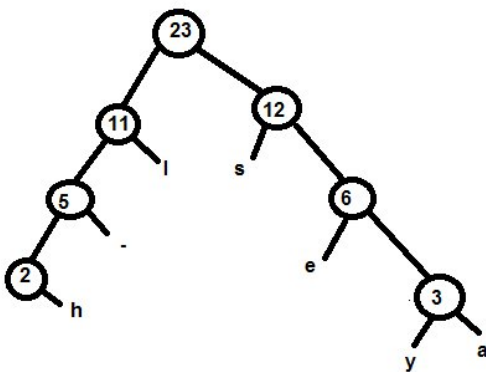
sally sell sea shells

we put the binary values to this string

s 10
a 1111
l 01
y 1110
_ 001
e 110
h 0001
! 0000

now we assign values to these characters

s a l l y - s e l l s - s e a - s h e l l !
10111101011110011011001011000110110111100110000111001011



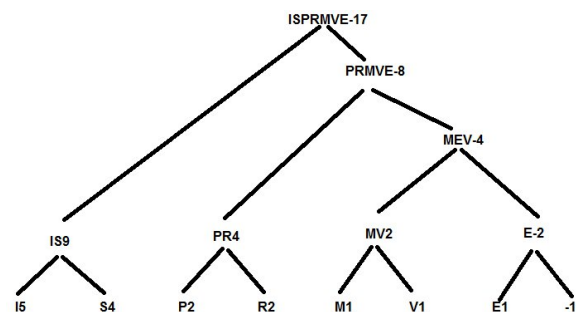
Text compression With Huffman Coding

Huffman coding technique use for compression .there is a idea that more common letter equal fewer bits, and similarly less common letter require more bits to represent them[7].in this case we use the string MISSISSIPPI RIVER=17characters .If we reduce the

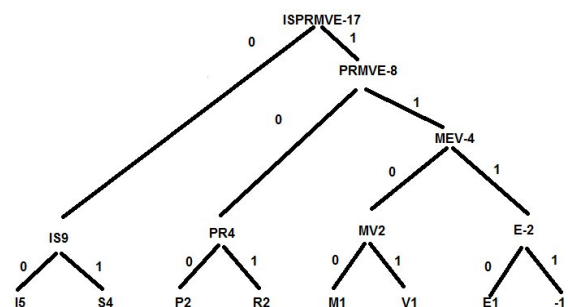
common ASCII we required 8 bit each for total of 136 bit, Now we will write that how much time each letter is present in the string. We will SEE

M	1
I	5
S	4
P	2
R	2
V	1
E	1
-	1

Now we sort this string by order I5 S4 P2 R2 M1 V1 E1 -1.now we make a tree and start with the small nodes

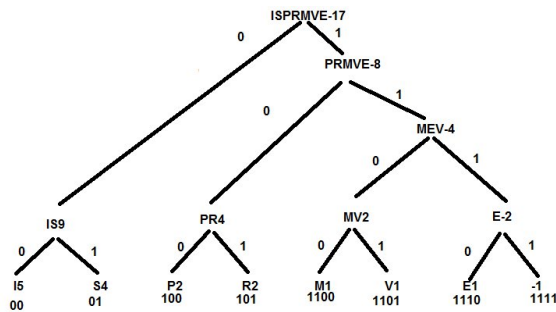


then we add them and make the root node so on. Now we write the branches as 0 and 1



this is the complete construction of tree. To find the binary tree the code of the letter just start the top of the tree and follow tree branches encoding of l is 00 and for S is 01 for P encoding is 100 do the same things for each other letter. We will see that letter with greatest

frequency have the lower presentation and lower frequency have the longest presentation that exactly we wanted.[9]



now we write down the bits for message

M I S S I S S I P P I R I V E R
1100 00 01 01 00 01 01 00 100 100 00 1111 101 00 1101 1110 101

this is Huffman coded binary message if we count this message is equal to 46 bits this is the lesser than the original 136 bits if we divide them we get 33% of the original one which save the 67 % of the original message in this way we can compress the text message through Huffman coding very easily.

CONCLUSION

In this paper we proof that how Huffman is cool and efficient algorithm in the filed of compression although there are more applications are available in the filed of compression like arithmetic coding and Lempel ziv, but mostly we see in our daily life Huffman algorithm work in our Iphone Ipods etc because it is not complex and more optimal among the family of symbol code. Huffman coding is a greedy algorithm so for the compression of data it is consider good that use longer bit instead of more frequent bits but

Arithmetic coding gives greater compression, is faster for adaptive models, and clearly separates the model from the channel encoding.

REFERENCES

[1].Gary Stix. Profile: David A. Huffman. Scientific American, 265(3):54, 58, September 1991.

[2]. Fundamentals of Information Theory and Coding Design

By Roberto Togneri, Christopher J.S deSilva

[3] Web Page Compression using

Huffman Coding Technique
Manjeet Gupta (Assistant professor)
Department of CSE JMIT Radaur
Brijesh Kumar (associate professor)
Department of IT Lingyaya's university.

[4]_Fundamental Data Compression

Author(s):Ida Mengyi Pu

[5] A fast adoptive Huffman coding algorithm

Communications, IEEE Transactions on (Volume:41 , Issue: 4)

[6] An efficient decoding technique for Huffman codes. Rezaul Alam Chowdhury(Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh)

Irwin King(Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, People's Republic of China)

[7] Evaluation of Huffman and Arithmetic

Algorithms for Multimedia Compression
Standards

Asadollah Shahbahrami, Ramin

Bahrampour

, Mobin Sabbaghi Rostami,

Mostafa Ayoubi Mobarhan,

Department of Computer Engineering,

Faculty of Engineering, University of

Guilan, Rasht, Iran

[8] Data Compression Scheme of
Dynamic Huffman Code for Different
Languages

1Shivani Pathak, 2Shradha Singh

, 3 Smita Singh, 4Mamta Jain

, 5Anand Sharma

[9] Huffman Data Compression

Joseph Lee

©May 23, 2007