Introduction

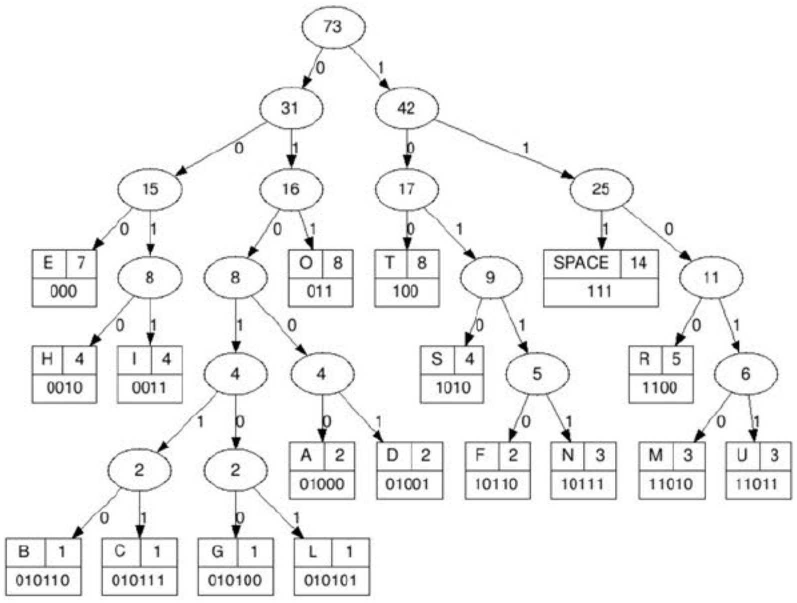
We in the [redacted] team are grateful for the opportunity to design a cryptosystem for OCRAI Creative Recursive Acronyms, Inc. We are glad to help you address the problem of security breaches. An ounce of prevention is truly worth a pound of cure, and we hope our cryptosystem will be a great help to you.

Our cipher system, modestly named the [redacted]-Huffman encryption, will help address your problem by encoding messages in a way that makes it very difficult to crack without knowing the key, which will help your company effectively protect it's messages and private information.

Cipher System

The cipher system that we've prepared for you is based on a system known as Huffman Compression. Typically, a text message is stored as a series of seven-digit codes of 0's and 1's, known as ASCII. For example, the letter A is represented as 1000001, while the character "^" is represented as 1011110. Huffman Compression recognizes that some things, like letters, are used far more frequently than other things like dollar signs and asterisks. In Huffman Compression, more frequent characters in a text are represented with shorter codes, while infrequent characters are represented by shorter codes.

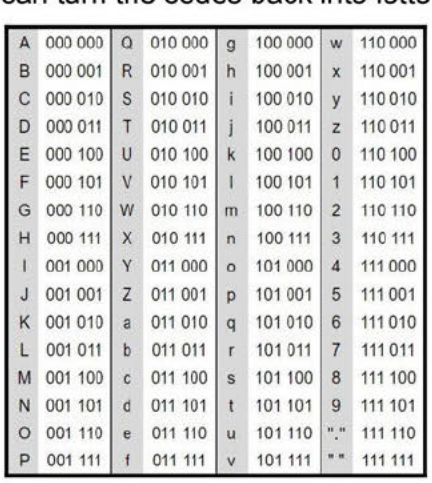
To create these shorter codes, Huffman Compression creates a frequency tree. To show an example of this kind of tree, we'll compress the phrase "OUR MEETING IS SCHEDULED FOR FOUR IN THE ROOM AT THE BOTTOM OF THE STAIRS". To start, we count the number



of times each character appears in this phrase. Some characters, like C and B, appear only once, while T appears 8 times and a space is used 14 times. Once we know the frequency all of the characters appear, we can combine low-frequency characters into higher frequency pairs. For example, while C and B only appear once each, there are two characters in the message that are either C or B, so they have a combined frequency of two. G and L appear once each, so they also have a combined frequency of two. C, B, G, and L all then have a combined frequency of 4. We can combine lower frequency characters or groups together until they've all been combined into a single tree, as shown in this image. In the case of ties for difference frequencies, the characters appear left to right in alphabetical order, as is the case with B, C, G, and L.

Once the frequency tree has been constructed, we can create new codes for each character by following the path to the character in the tree, with left branches being represented by 0's and right branches being represented by 1's. For example, the letter E has the code 000, because to find it we move left three times in the tree, while U has a code of 11011, because we move right twice, then left, then right two more times. We then replace all of the characters in our message with the codes; for example, "OUR" would be replaced with 011 11011 1100, as the codes for O, U, and R are 011, 11011, and 1100 respectively in this tree.

Once we have replaced the entire message with the codes from our frequency tree, we can turn the codes back into letters using the codes from this table for the final step of the cipher. The table gives each of the letters and numbers, as well as the space character, a six-digit code. We can use this table to break up our message in 6-digit codes, padding with 1's at the end so the total number of digits is divisible by six. So, for our combined code of 011110111100 for "OUR", the first 6 digits (011 110) would be replaced with the letter "e", as the code for "e" is 011 110. We would follow this pattern for the entirety of the message. As an example, the first 24 digits of our message after converting to 0's and 1's is the following: 011110 111100 111110 100000. When converting this using our table, we get "e80g". We would send this encoded message by text to the recipient via text, at which point the recipient would be able to turn this encoded message back into the regular text.



A	000 000	Q	010 000	g	100 000	w	110 000
B	000 001	R	010 001	h	100 001	x	110 001
C	000 010	S	010 010	i	100 010	y	110 010
D	000 011	T	010 011	j	100 011	z	110 011
E	000 100	U	010 100	k	100 100	0	110 100
F	000 101	V	010 101	l	100 101	1	110 101
G	000 110	W	010 110	m	100 110	2	110 110
H	000 111	X	010 111	n	100 111	3	110 111
I	001 000	Y	011 000	o	101 000	4	111 000
J	001 001	Z	011 001	p	101 001	5	111 001
K	001 010	a	011 010	q	101 010	6	111 010
L	001 011	b	011 011	r	101 011	7	111 011
M	001 100	c	011 100	s	101 100	8	111 100
N	001 101	d	011 101	t	101 101	9	111 101
O	001 110	e	011 110	u	101 110	"."	111 110
P	001 111	f	011 111	v	101 111	" "	111 111

In a regular Huffman Compression, the frequency tree is attached to the message, as there's no way to turn the text "011110111100" back into our original message of "OUR" without the frequency tree. In our encryption system, we will only send a list of character frequencies with the message instead of the tree, so that there's no way that someone intercepting the message can decipher the message. The recipient will have a key that tells them which frequencies belong to each character. For example, the message we send could begin with 4 5 3 2. If the recipient's key began with IRMD, it would signify that the frequencies of I, R, M, and D would be 4, 5, 3, and 2 respectively. Each person would have a unique key made of all of the characters a person could type shuffled together. Because this key wouldn't be sent with the message, anyone that intercepted the message would only see a series of numbers and the ciphertext, which no indication how to build the correct frequency tree.

Each key should be a randomized list of all of the valid characters that the people communicating want to be able to use. For example, the dollar sign can only be used if it is included in the key. For this reason, we recommend that all characters available on a standard keyboard be used in the key. However, because every character in the key requires more padding at the beginning of the message, the encryption system allows for smaller keys. One of the biggest advantages of having larger keys is the added security; most keyboards have 100+ characters, with the number of keys being equal to the number of characters raised to the power of the number of characters. In the case of 100 characters, there are 100^100 keys, or 10^200. In addition, because each pair of keys can include any characters, including emoticons, it is very difficult to break this code by guessing what characters are based on their frequencies, as no hacker can know what possible characters are included in the code.

Conclusion

The [redacted]-Huffman encryption system, which works by building a character frequency tree for each message and encoding the characters by their position on the tree, will help your company encrypt messages. For longer messages, because it is based on a compression algorithm, it may even have the benefit of compressing your data as well. The encryption will help secure company data and prevent accidental data leakage from employees. You can rest secure knowing that your data is protected and secure.

Notes: you may find the tool: https://www.csfieldguide.org.nz/en/interactives/huffman-tree/ helpful in developing a huffman tree from a given text string