Binary tree applications

- Huffman coding
- Shannon-Fano coding

Huffman Coding

- Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters.
- The most frequent character gets the smallest code and the least frequent character gets the largest code.
- The variable-length codes assigned to input characters are Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.

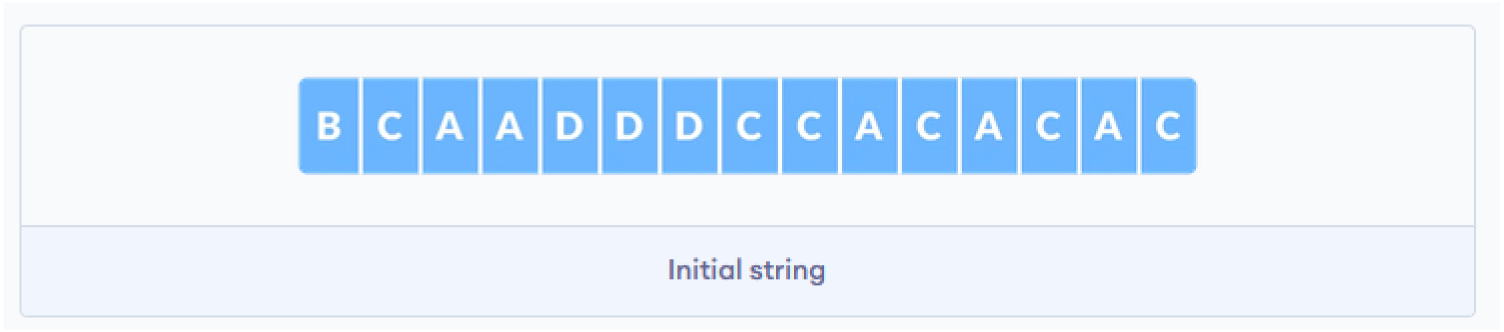
Source:https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

Huffman Coding

- Let us understand prefix codes with a counter example.
- Let there be four characters a, b, c and d, and their corresponding variable length codes be
- a:00
- b:01
- c:0
- d:1
- This coding leads to ambiguity because code assigned to c is the prefix of codes assigned to a and b.
- If the compressed bit stream is 0001, the decompressed output may be
- “cccd” or “ccb” or “acd” or “ab”.

Source:https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/c

Huffman Coding Example



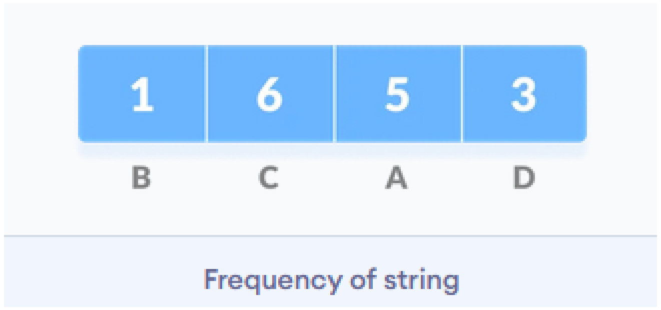
- Suppose the string above is to be sent over a network. Each character occupies 8 bits.
- There are a total of 15 characters in the above string. Thus, a total of 8 * 15 = 120 bits are required to send this string.
- Using the Huffman Coding technique, we can compress the string to a smaller size.
- Huffman coding first creates a tree using the frequencies of the character and then generates code for each character.
- Once the data is encoded, it has to be decoded. Decoding is done using the same tree.
- Huffman Coding prevents any ambiguity in the decoding process using the concept of prefix code ie. a code associated with a character should not be present in the prefix of any other code. The tree created above helps in maintaining the property.

https://www.programiz.com/dsa/huffman-coding

Huffman Coding Example

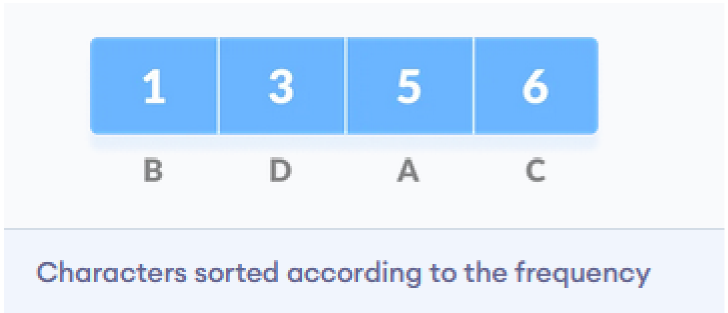
- Huffman coding is done with the help of the following steps.

1- Calculate the frequency of each character in the string.



Frequency of string

2- Sort the characters in increasing order of the frequency. These are stored in a priority queue Q.

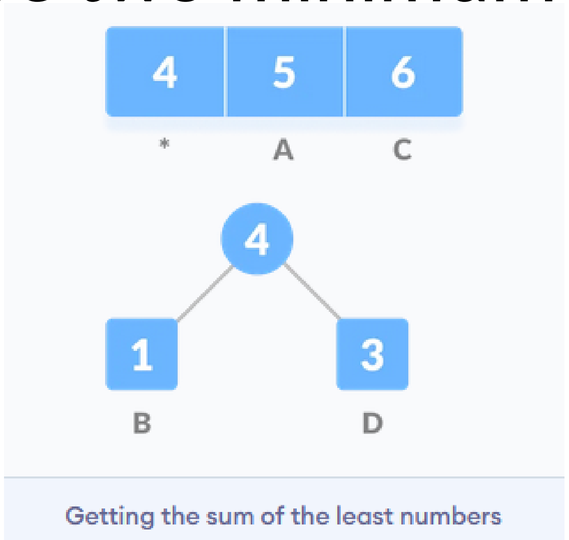


Characters sorted according to the frequency

Huffman Coding Example

3-Make each unique character as a leaf node.

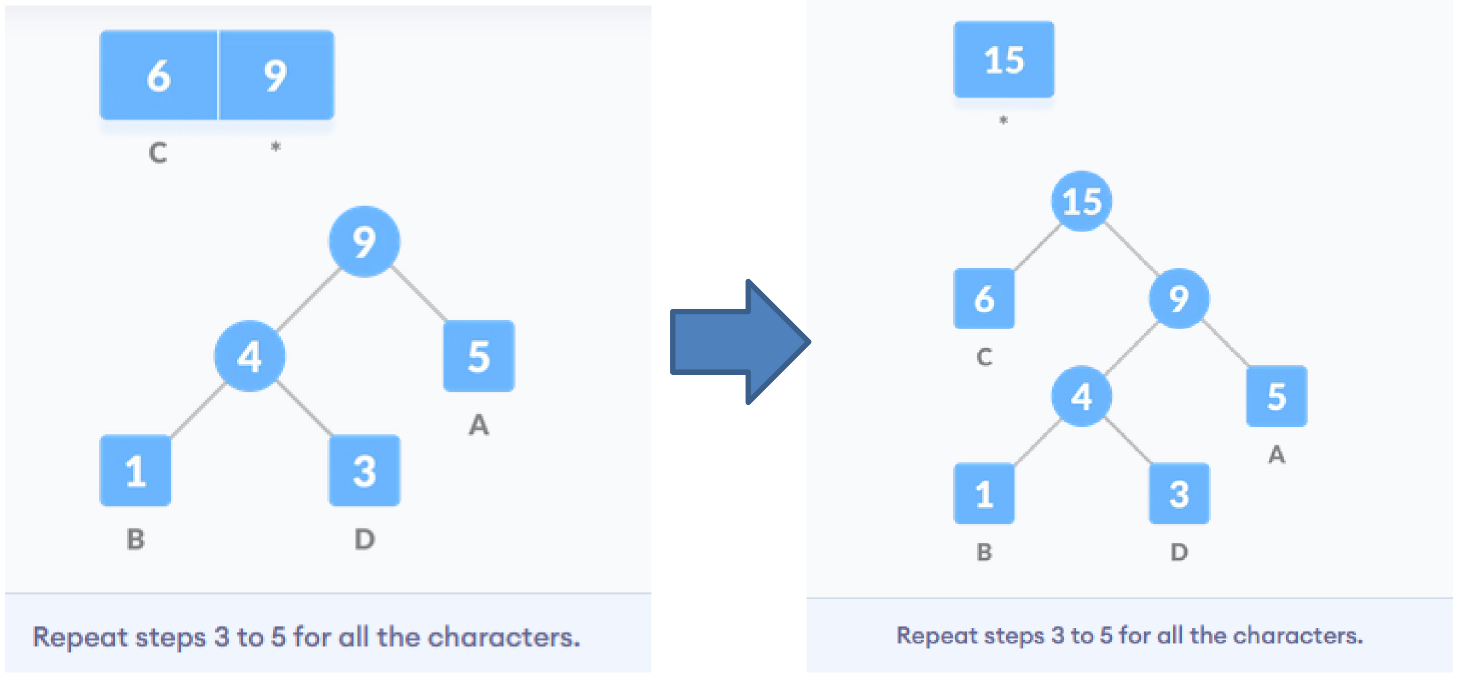
4-Create an empty node. Assign the minimum frequency to the left child of new node and assign the second minimum frequency to the right child. Set the value of the *new node* as the sum of the above two minimum frequencies.



Getting the sum of the least numbers

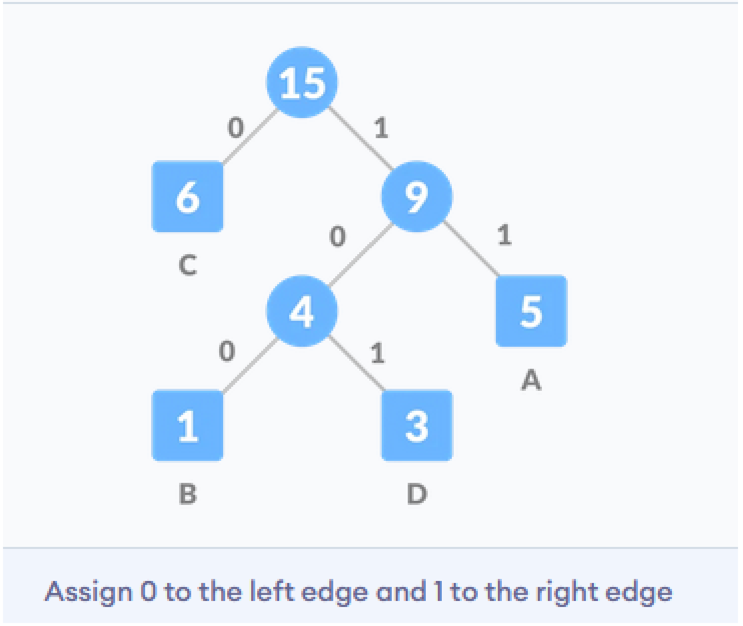
Huffman Coding Example

- Remove these two minimum frequencies from Q and add the sum into the list of frequencies (* denote the internal nodes in the figure above).
- Insert a new node into the tree.
- Repeat steps 3 to 5 for all the characters.

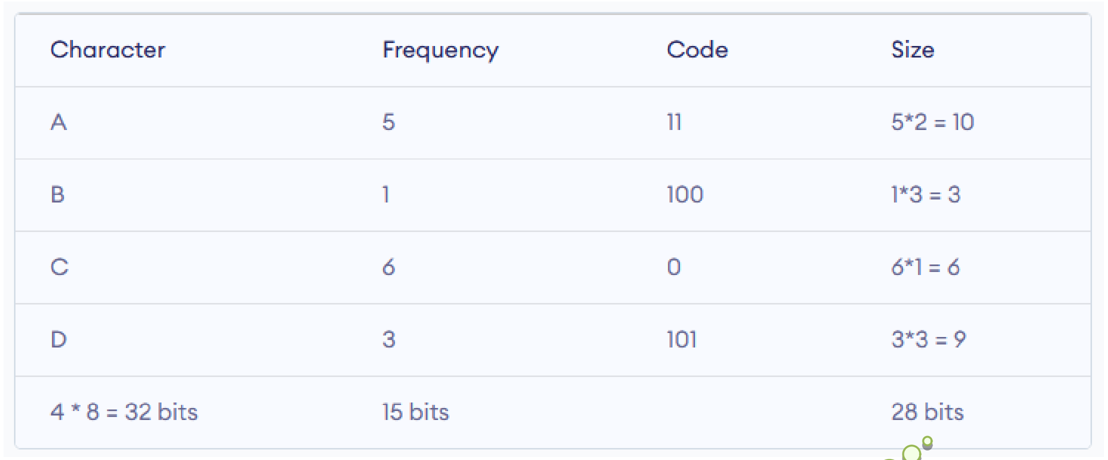

6
9
C
*
9
4
5
1
3
A
B
D
Repeat steps 3 to 5 for all the characters.
15
*
15
6
9
C
4
5
A
1
3
B
D
Repeat steps 3 to 5 for all the characters.

Huffman Coding Example

- For each non-leaf node, assign 0 to the left edge and 1 to the right edge.
- For sending the above string over a network, we have to send the symbol codes as well as the compressed-code. The total size is given by the table below.



Assign 0 to the left edge and 1 to the right edge

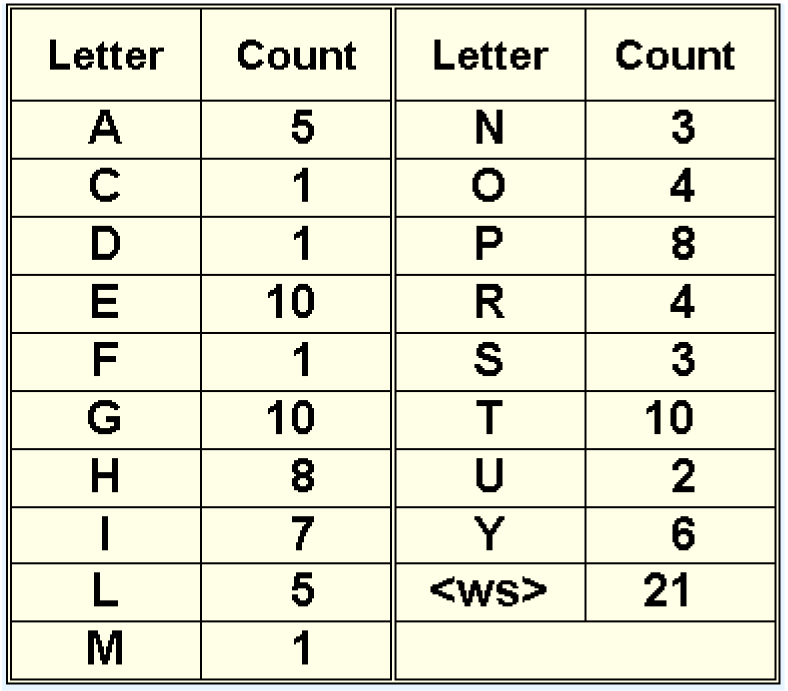


Character	Frequency	Code	Size
A	5	11	5*2 = 10
B	1	100	1*3 = 3
C	6	0	6*1 = 6
D	3	101	3*3 = 9
4 * 8 = 32 bits	15 bits		28 bits

Before compression:120 bits

Huffman Coding Example 2

```
HIGGLETY PIGGLTY POP
THE DOG HAS EATEN THE MOP
THE PIGS IN A HURRY
   THE CATS IN A FLURRY
HIGGLETY PIGGLTY POP
```

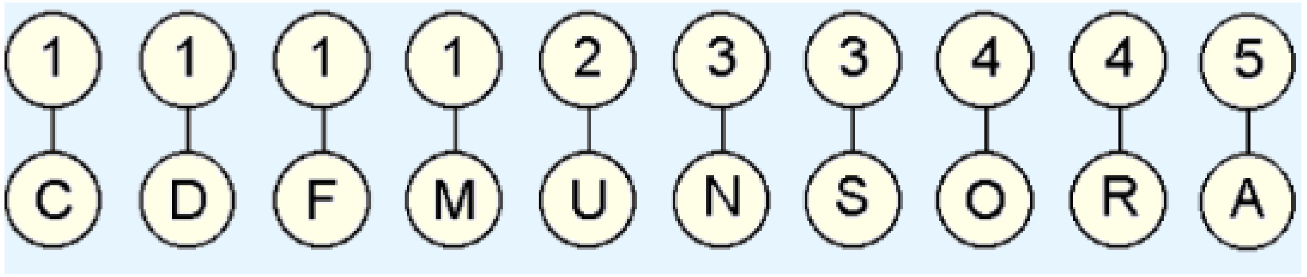


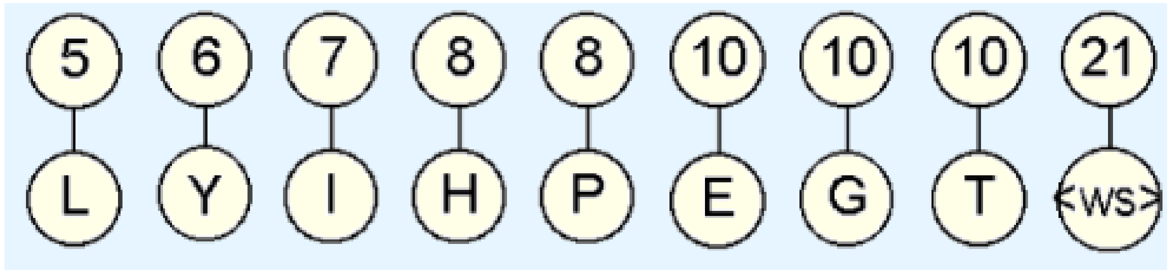
Letter	Count	Letter	Count
A	5	N	3
C	1	O	4
D	1	P	8
E	10	R	4
F	1	S	3
G	10	T	10
H	8	U	2
I	7	Y	6
L	5	<ws>	21
M	1		

Kaynak: *web2.aabu.edu.jo › tool › course_file › lec_notes*

Huffman Coding Example 2

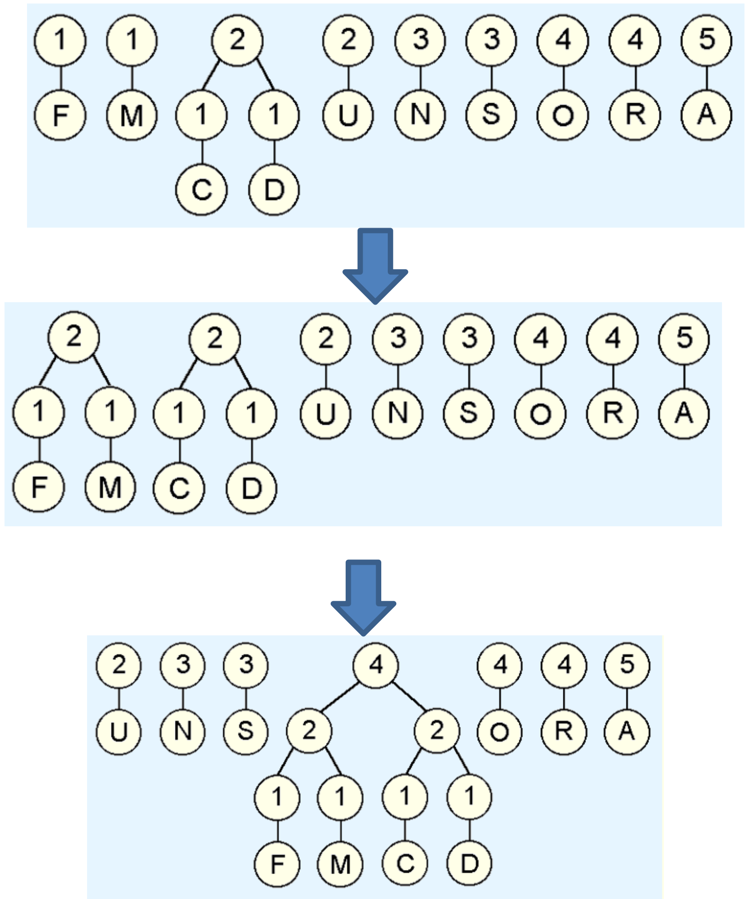
- Next, place the letters and their frequencies into a forest of trees that each have two nodes: one for the letter, and one for its frequency.



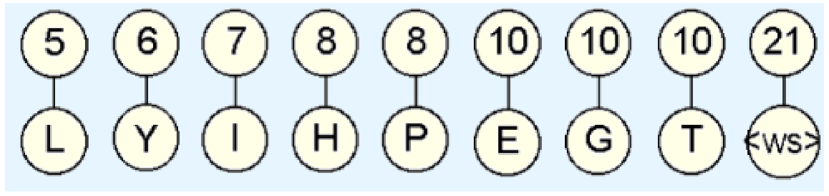


Huffman Coding Example 2

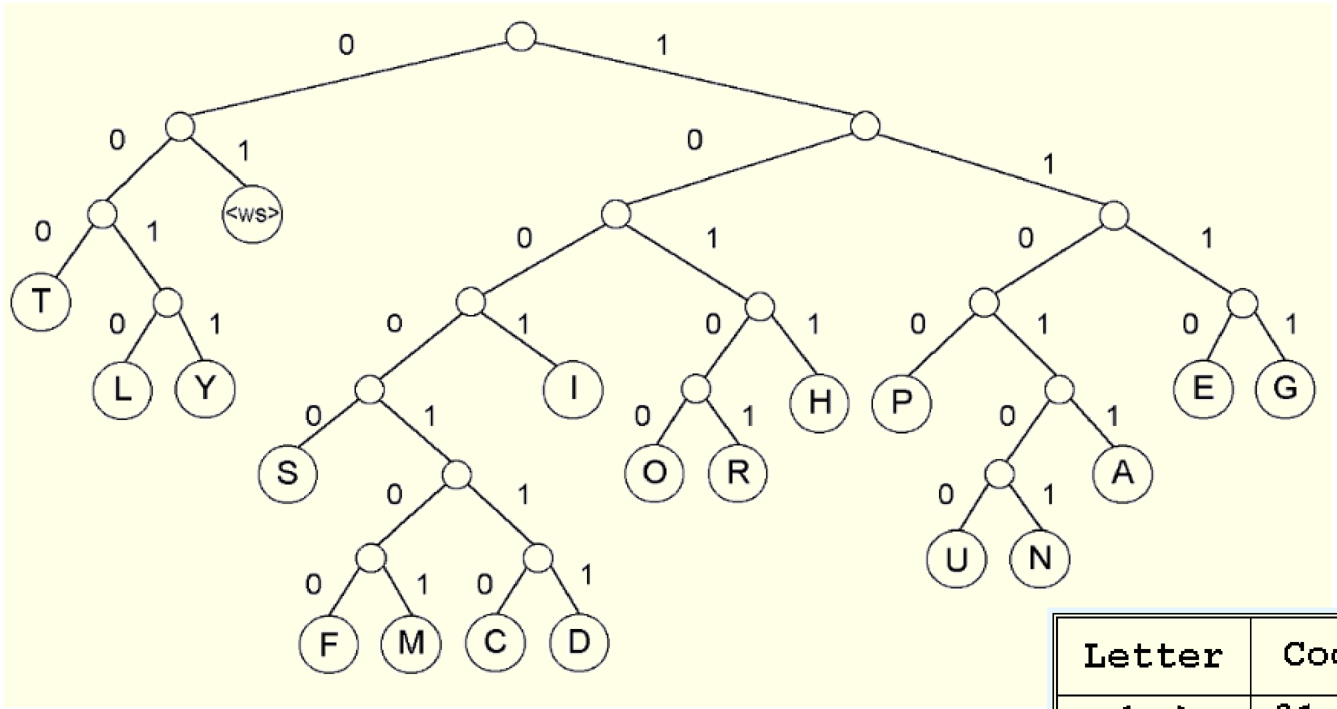
- We start building the tree by joining the nodes having the two lowest frequencies.



1 F, 1 M, 2 (1 C, 1 D), 2 U, 3 N, 3 S, 4 O, 4 R, 5 A
2 (1 F, 1 M), 2 (1 C, 1 D), 2 U, 3 N, 3 S, 4 O, 4 R, 5 A
2 U, 3 N, 3 S, 4 (2 (1 F, 1 M), 2 (1 C, 1 D)), 4 O, 4 R, 5 A



Huffman Coding Example 2



Letter	Code	Letter	Code
<ws>	01	O	10100
T	000	R	10101
L	0010	A	11011
Y	0011	U	110100
I	1001	N	110101
H	1011	F	1000100
P	1100	M	1000101
E	1110	C	1000110
G	1111	D	1000111
S	10000		

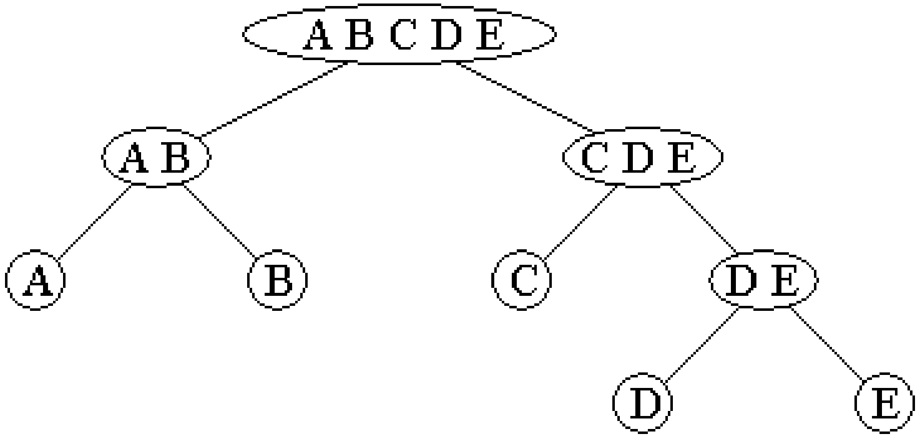
This is the code derived from this tree.

Shannon-Fano Coding

- According to Shannon and Fano, an ordered table providing the frequency of any symbol is required to construct a code tree.
- Each part of the table will be divided into two parts.
- The algorithm should ensure that the segment has a close sum of the top and bottom.
- These steps are repeated until only one symbol remains.

Symbol	Frequency	Code Length	Code	Total Length
A	24	2	00	48
B	12	2	01	24
C	10	2	10	20
D	8	3	110	24
E	8	3	111	24

total: 62 symbols   SF coded: 140 Bit


A B C D E
A B
C D E
A
B
C
D E
D
E