*Ahmed Adel Samir 20102650*

*Hossam Shiha 20109088*

Computer Algorithm’s Project

[Year]

**Explanation:**

-We imported the heapq library

-We defined some variables

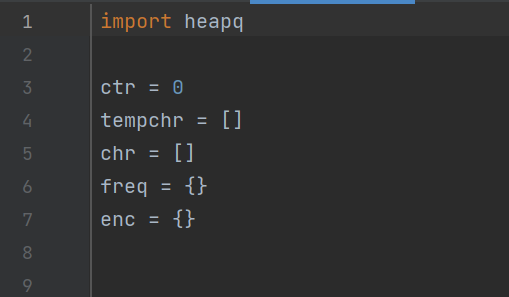
Ctr to count the number of lines in the input text file.

Tempchr list to store the characters of each line.

Chr array to store each character from each line.

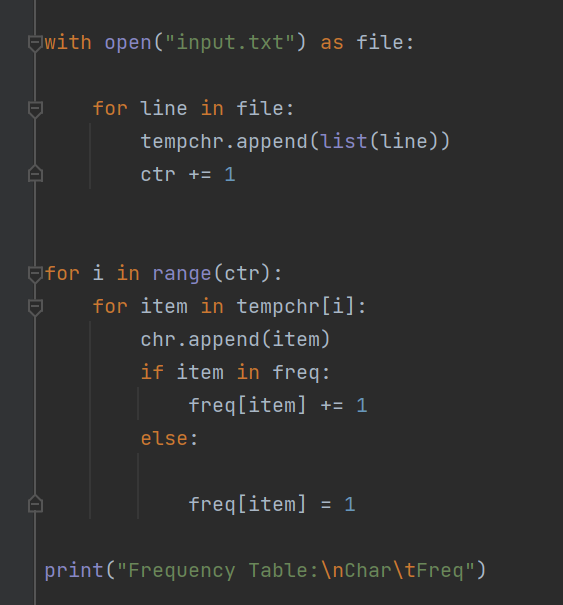
Feq dictionary to store each character with its frequency.

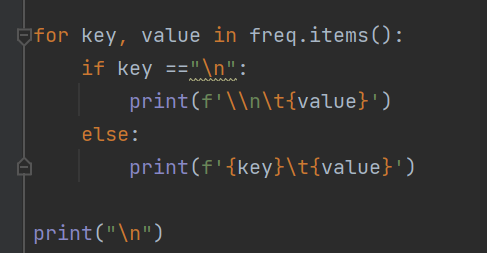
Enc dictionary to store each character with its encoding.



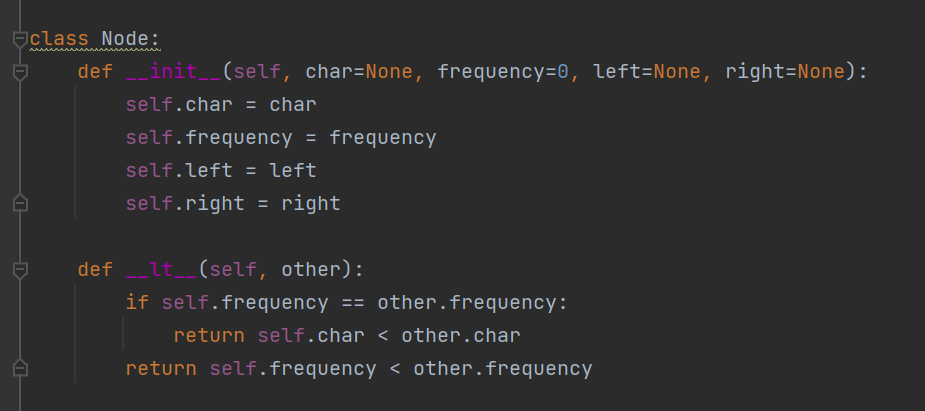
-We read the input from the input file text and stored it in tempchr while incrementing the counter ctr

-We made a for loop to store the character and its frequency in the dictionary freq and printed the frequency table

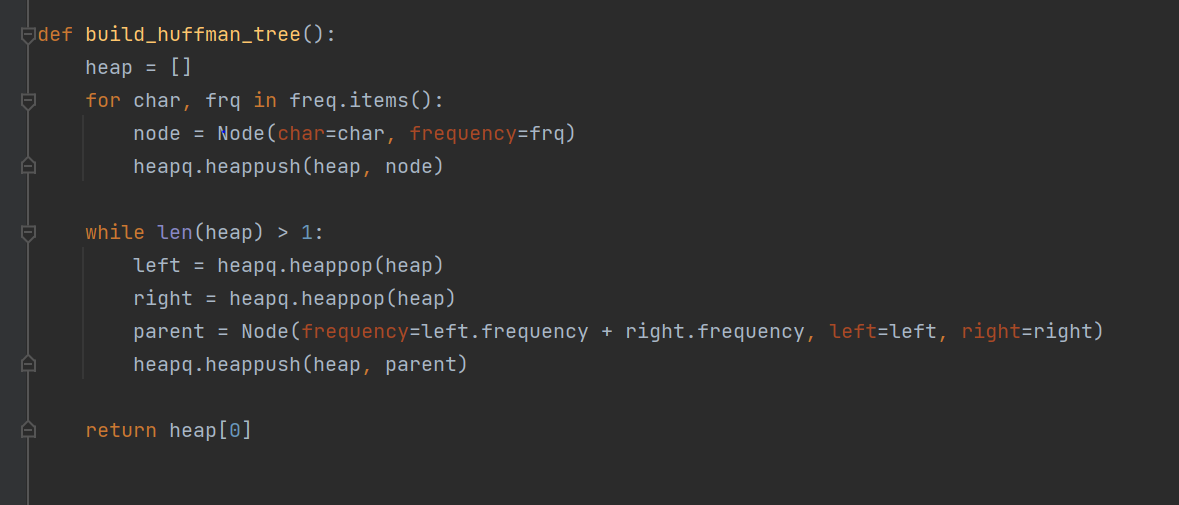




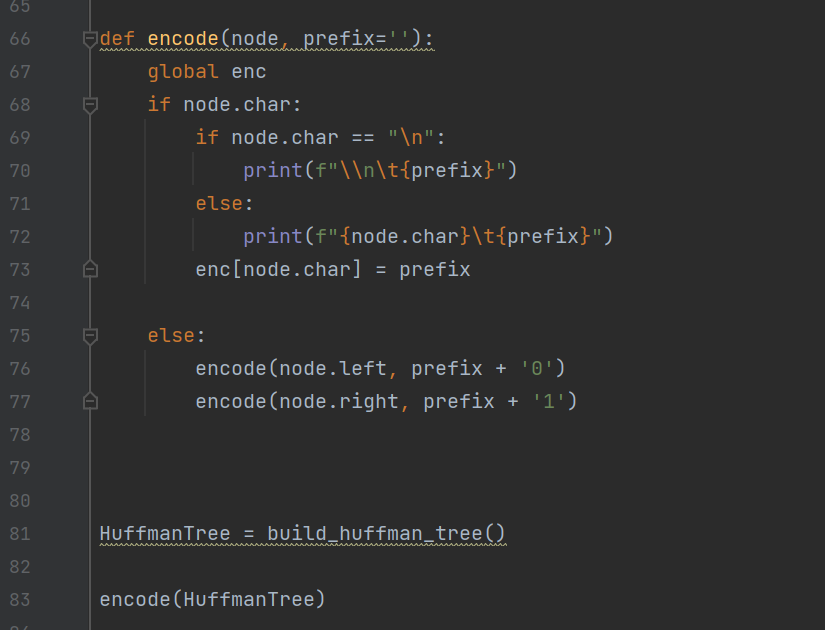
-We created a class called node that holds the character, frequency, left node, and right node as attributes and defined a method that sorts the nodes according to their frequency



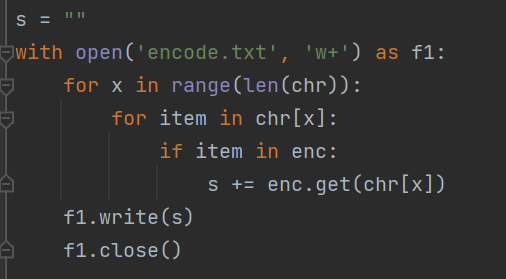
-We created a function called build\_huffman\_tree to build the tree by using the heap built-in functions such as push and pop and also used the node class to create instances (nodes) to build the tree and it returns the parent of the tree as a heap array that holds all left and right nodes including its characters and frequencies and stored in a variable called HuffmanTree by calling the function while assigning the variable.



-We printed the encode table by using the encode function that takes node (parent from build\_huffman\_tree function) and prefix as parameters as it checks if the current node has a character and if not, it recalls the encode function twice with new arguments one for the left node with prefix = 0 and the other for the right node with prefix = 1

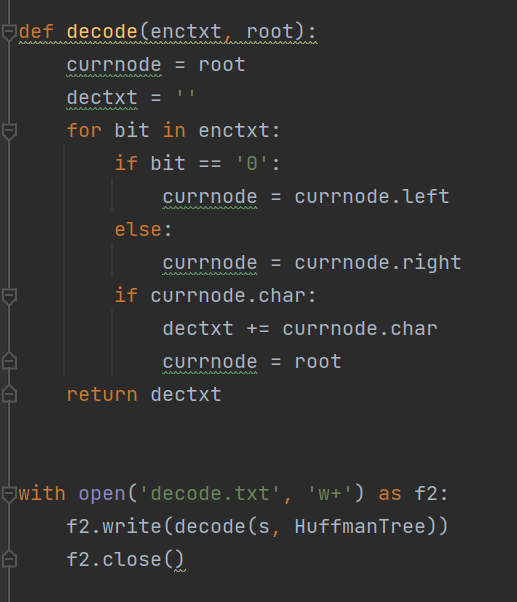


-We created an empty string variable called s and then opened the encode.txt file to write in it the encoded text so we created a nested for loop to concatenate the encoded text in the variable s and then we used the built-in functions to write in the file encode.txt



-We finally created a function to decode called decode that takes enctxt and root as parameters and stores the root in a variable called currnode which means the current node and declares an empty decode string variable as dectxt as it decodes the text in the variable s by using a for loop that searches in every bit (0 or 1) in the encoded text (enctxt) and if condition that checks if the bit is 0 and it changes the currnode to the left node of the currnode and if it is 1 it changes it to the right node of it and it checks with an if condition the current node has a character or not and if it has a character it concatenates the character in the string variable dectxt and it changes the currnode back to the root node (parent) and returns the dectxt

- Finally, we wrote in the decode.txt file the decoded output from the decode function by calling it through the write function and passing the arguments s (encoded text) and HuffmanTree (that calls the build\_huffman\_tree function)



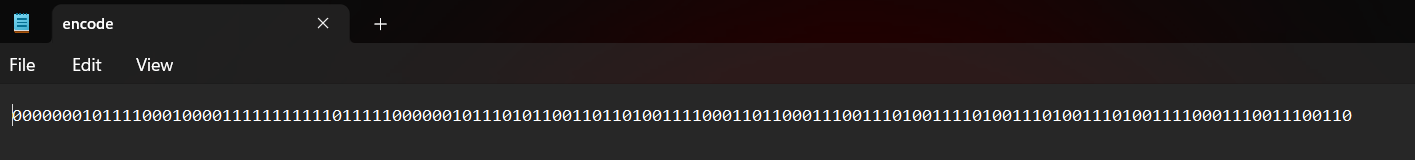
**A screenshot of a computer program

Description automatically generated with medium confidence** **Screenshots:**

**Output:**

**Files:**

**A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer program

Description automatically generated with medium confidence**