

Experiment No -3

Name: HUZAIB MULLA

RollNo:18ET15

Aim: To find Compression ratio using LZ78

Requirments: Python 3.4

Theory:

LZ78-based schemes work by entering phrases into a dictionary and then, when a repeat occurrence of that particular phrase is found, outputting the dictionary index instead of the phrase.

Every step LZ78 will send a pair (i,a) to the output, where i is an index of the phrase into the dictionary and a is the next symbol following immediately after the found phrase. The dictionary is represented like the trie with numbered nodes. If we go from the root to a certain node, we will get phrase from the input text.

In each step we look for the longest phrase in dictionary, that would correspond to the unprocessed part of the input text. Index of this phrase together with the symbol, which follows the found part in input text, are then send to the output. The old phrase extended by the new symbol is then put into dictionary. This new phrase is numbered by the smallest possible number.

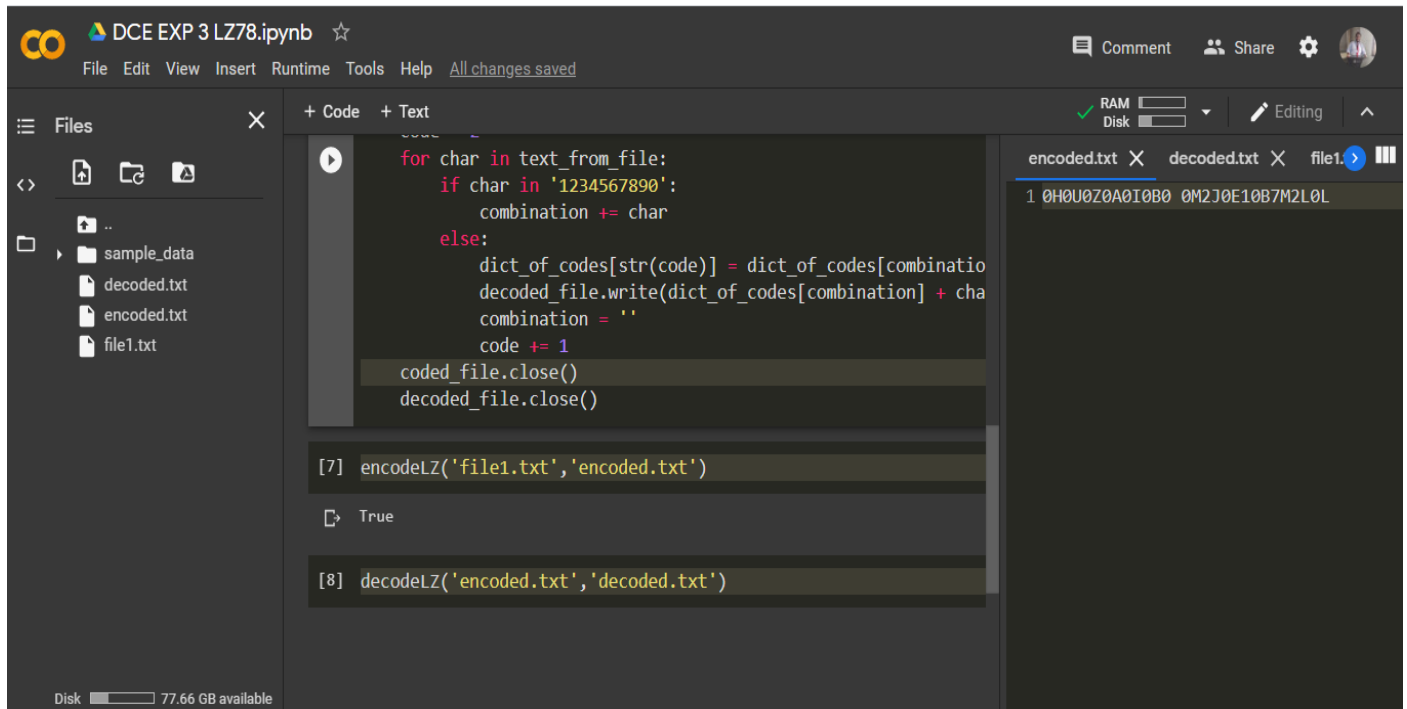
The coding will start with tree, that has only one node, which represents empty string.

Encoder and Decoder Program:

```
def encodeLZ(FileIn, FileOut):
    input_file = open(FileIn, 'r')
    encoded_file = open(FileOut, 'w')
    text_from_file = input_file.read()
    dict_of_codes = {text_from_file[0]: '1'}
    encoded_file.write('0' + text_from_file[0])
    text_from_file = text_from_file[1:]
    combination = ''
    code = 2
    for char in text_from_file:
        combination += char
        if combination not in dict_of_codes:
            dict_of_codes[combination] = str(code)
            if len(combination) == 1:
                encoded_file.write('0' + combination)
            else:
                encoded_file.write(dict_of_codes[combination[0:-
1]] + combination[-1])
            code += 1
            combination = ''
    input_file.close()
    encoded_file.close()
    return True

def decodeLZ(FileIn, FileOut):
    coded_file = open(FileIn, 'r')
    decoded_file = open(FileOut, 'w')
    text_from_file = coded_file.read()
    dict_of_codes = {'0': '', '1': text_from_file[1]}
    decoded_file.write(dict_of_codes['1'])
    text_from_file = text_from_file[2:]
    combination = ''
    code = 2
    for char in text_from_file:
        if char in '1234567890':
            combination += char
        else:
            dict_of_codes[str(code)] = dict_of_codes[combination] + char
            decoded_file.write(dict_of_codes[combination] + char)
            combination = ''
            code += 1
    coded_file.close()
    decoded_file.close()
```

Output:



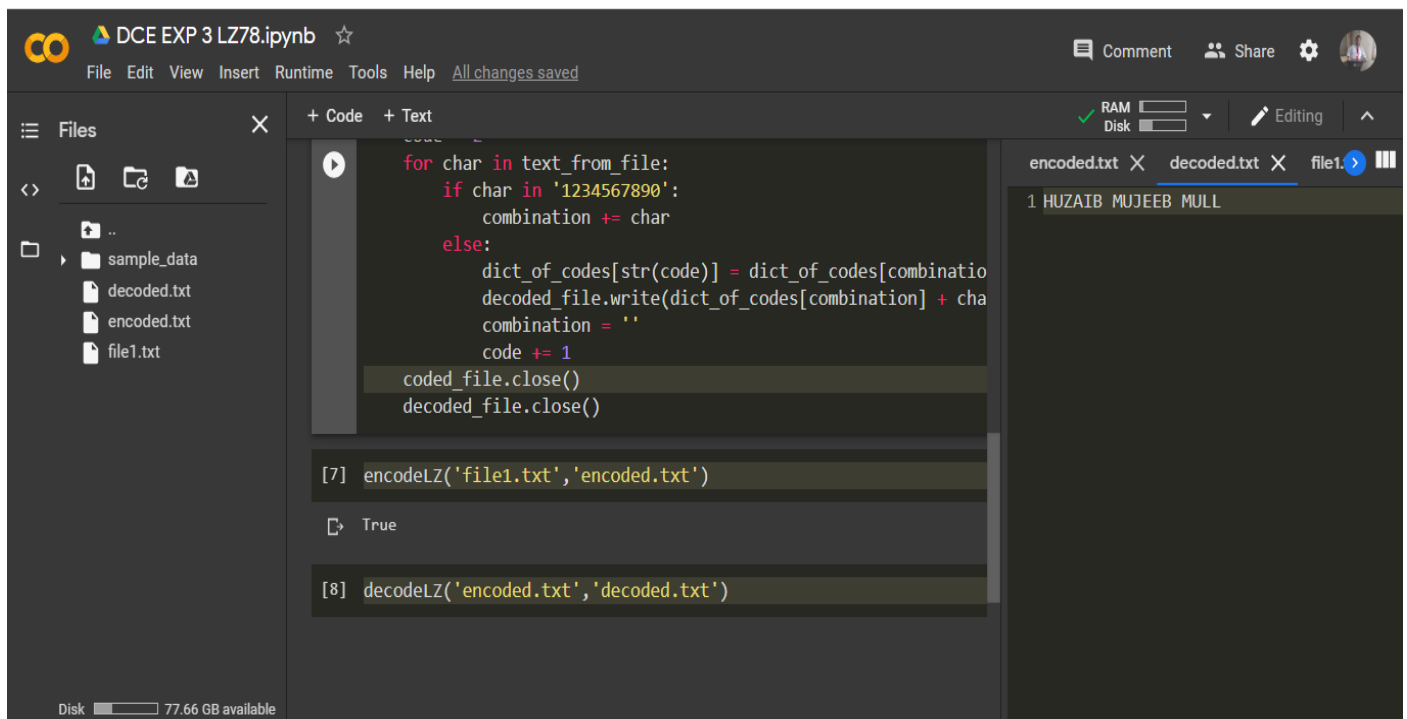
This screenshot shows a Jupyter Notebook titled "DCE EXP 3 LZ78.ipynb". The interface includes a top menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a left sidebar with a file explorer showing "sample_data", "decoded.txt", "encoded.txt", and "file1.txt", and a right sidebar with tabs for "encoded.txt", "decoded.txt", and "file1.txt". The main code area contains a Python function for LZ78 encoding and decoding, followed by two code cells. The first code cell runs the encoding function, and the second code cell runs the decoding function. The output of the first cell is a hex string, and the output of the second cell is the decoded text.

```
for char in text_from_file:
    if char in '1234567890':
        combination += char
    else:
        dict_of_codes[str(code)] = dict_of_codes[combination] + char
        combination = ''
        code += 1
coded_file.close()
decoded_file.close()

[7] encodeLZ('file1.txt','encoded.txt')
True

[8] decodeLZ('encoded.txt','decoded.txt')
```

RAM: 77.66 GB available



This screenshot shows the same Jupyter Notebook as above, but with the output of the decoding function visible. The output of the first cell is the same hex string as before. The output of the second cell is the decoded text "HUZAIB MUJEEB MULL".

```
for char in text_from_file:
    if char in '1234567890':
        combination += char
    else:
        dict_of_codes[str(code)] = dict_of_codes[combination] + char
        combination = ''
        code += 1
coded_file.close()
decoded_file.close()

[7] encodeLZ('file1.txt','encoded.txt')
True

[8] decodeLZ('encoded.txt','decoded.txt')
```

RAM: 77.66 GB available