CHAITRA SAMANT

231070055

DAA LAB 5B

1. **AIM:**
   Download books from the website in html, text, doc, and pdf format. Compress these books using Hoffman coding technique. Find the compression ratio.

2. **PROGRAM:**

```
import heapq
from collections import Counter
import PyPDF2
import docx
from bs4 import BeautifulSoup


class Node:
    def __init__(self, char, freq):
        self.char = char
        self.freq = freq
        self.left = None
        self.right = None

    def __lt__(s1, s2):
        return s1.freq < s2.freq

# Read Functions

def read_pdf(file_path):
    pdf_text = ""
    with open(file_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        for page_num in range(len(reader.pages)):
            pdf_text += reader.pages[page_num].extract_text()
    return pdf_text


def read_docx(file_path):
    doc = docx.Document(file_path)
    doc_text = ""
    for paragraph in doc.paragraphs:
        doc_text += paragraph.text
    return doc_text
```

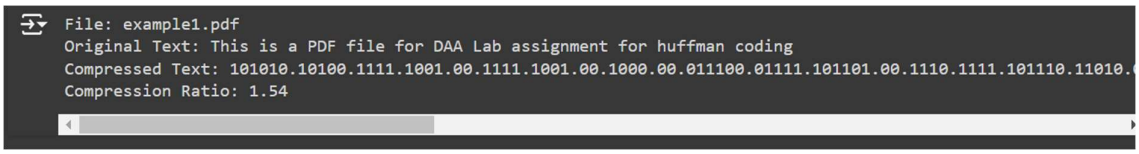
```
def read_html(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        soup = BeautifulSoup(file, 'html.parser')
        html_text = soup.get_text()
    return html_text

def read_txt(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        txt_text = file.read()


    if not txt_text.strip():
        print("File is empty")
        exit()

    return txt_text




def read_file(file_path):
    if file_path.endswith('.pdf'):
        return read_pdf(file_path)
    elif file_path.endswith('.docx'):
        return read_docx(file_path)
    elif file_path.endswith('.html'):
        return read_html(file_path)
    elif file_path.endswith('.txt'):
        return read_txt(file_path)
    else:

        raise ValueError("Unsupported file type. Please provide a file
in PDF, DOCX, HTML, or TXT format.")


# Build Huffman tree
def build_huffman_tree(frequency):
    heap = [Node(char, freq) for char, freq in frequency.items()]
    heapq.heapify(heap)

    while len(heap) > 1:
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)
        merged = Node(None, left.freq + right.freq)
        merged.left = left
        merged.right = right
        heapq.heappush(heap, merged)

    return heap[0]
```

```
# Generate Huffman codes
def generate_codes(root, prefix='', codebook={}):
    if root is None:
        return
    if root.char is not None:
        codebook[root.char] = prefix
    generate_codes(root.left, prefix + '0', codebook)
    generate_codes(root.right, prefix + '1', codebook)
    return codebook


def compress(text):
    frequency = Counter(text)
    if frequency == 0:
      print("Empty File")
      exit(1)
    huffman_tree = build_huffman_tree(frequency)
    codebook = generate_codes(huffman_tree)

    compressed_text = ''.join(codebook[char] for char in text)
    return compressed_text, huffman_tree, codebook


def calculate_compression_ratio(original_text, compressed_text):
    original_size = len(original_text) * 8
    compressed_size = len(compressed_text)
    return round(original_size / compressed_size, 2)

# Driver code

file_path = 'example1.pdf'
text = read_file(file_path)

compressed_text, huffman_tree, codebook = compress(text)

compression_ratio = calculate_compression_ratio(text, compressed_text)
print(f'File: {file_path}')
print("Original Text:", text.replace('\n', ' '))
print("Compressed Text:", compressed_text)
print("Compression Ratio:", compression_ratio)
```

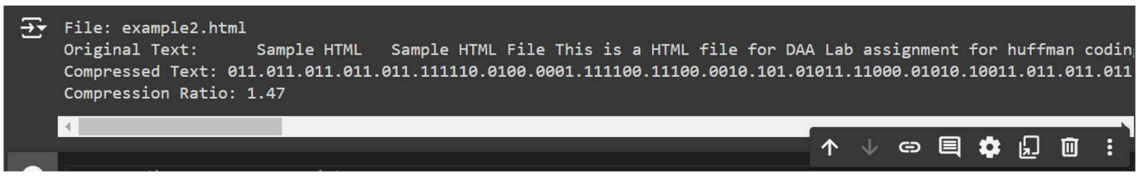
3. TESTCASES:

Positive Testcases:

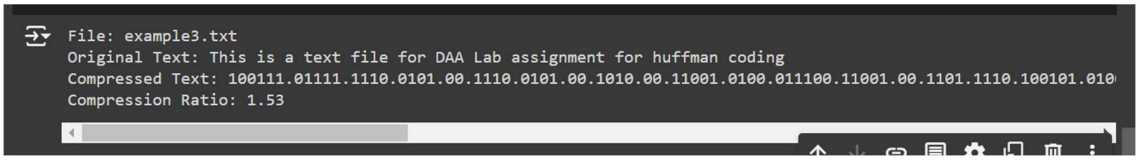
1. PDF File:



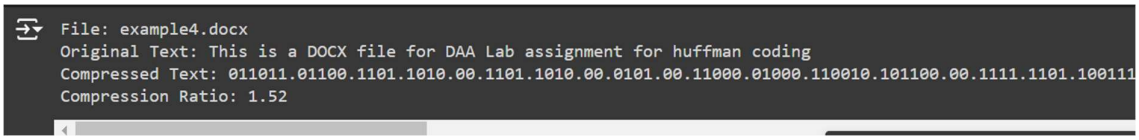
2. HTML File:



3. Text File:

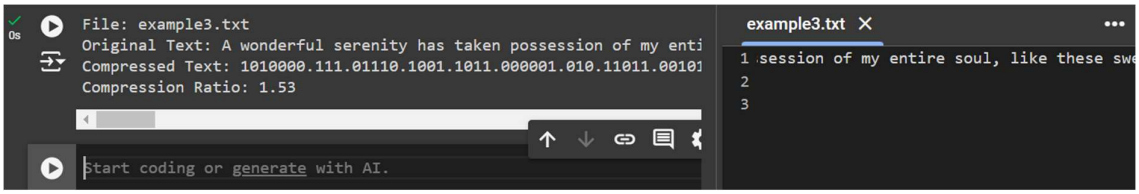


4. Doc File



5. Only one character in the file

6. MST Question



Negative Testcases:

7. Incorrect File Path

```
FileNotFoundError: [Errno 2] No such file or directory: 'example9.pdf'
```

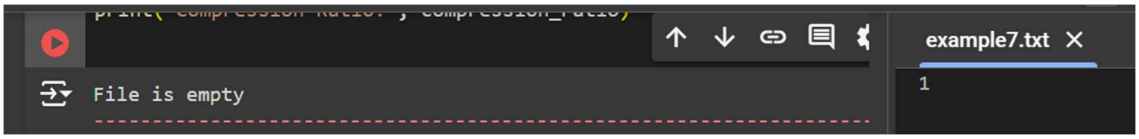
8. Libraries/Frameworks not installed

```
...  ---------------------------------------------------------------------------
     ModuleNotFoundError                       Traceback (most recent call last)
     Cell In[1], line 4
           2 import heapq
           3 from collections import Counter
     ----> 4 import PyPDF2
           5 import docx
           6 from bs4 import BeautifulSoup

     ModuleNotFoundError: No module named 'PyPDF2'
```

9. Not supported file type

```
ValueError: Unsupported file type. Please provide a file in PDF, DOCX, HTML, or TXT format.
```

10. File is empty



CONCLUSION

Hence, we successfully implemented Huffman coding algorithm to compress text from various file formats, including PDF, DOCX, HTML, and TXT. The program handles different input formats by identifying the file type and extracting text accordingly, then compresses the extracted content using Huffman coding, which generates a unique binary code for each character based on its frequency. The compression ratio is calculated to evaluate the effectiveness of the process.