Programme	:	**B.Tech** Semester : **Win Sem 21-22**
Course	:	**Web Mining Lab** Code : **CSE3024**
Faculty	:	**Dr.Bhuvaneswari A** Slot : **L7+L8**
Date	:	**11-02--2022** Marks : **10 Points**

Vaibhav Agarwal

19BCE1413

1. Apply run length encoding for the following string and compress it.

"eeeeeeerrrrrrrrrrrrrrrrttttttttttttttttiiiiiiiiffffffeft"

2. Consider the following Inverted Index File with Terms, Occurrences and Posting List

Term	**Occurrences**	**Posting List (Doc ids)**
Samsung	233	2, 12, 34544, 34574, 35569, …
Airtel	12	12, 17, 25, 148, 156, 159, 172, …
Mercury	15	1, 2, 3, 7, 9, 10, …
Venus	12	23, 45, 78, 122, 145, …
Fiber	6	1, 3, 5, 7, 19, 20

i. Apply Binary coding for term "Mercury" (apply for all doc ids)

ii. Apply Unary coding for term "Fiber"

iii. Apply Elias Gamma Encoding for term "Airtel"

iv. Apply Elias Delta Decoding for "000010000"

v. Apply Elias Delta Encoding for term "Venus"

vi. Apply Elias Delta Decoding for "00101001"

vii. Apply Variable Byte Encoding for “Samsung”. (Use doc ids gap)

19BCE1413_Lab_5.ipynb

Users > vaibhavagarwal > sem 6 > web mining > lab > lab 5 > 19BCE1413_Lab_5.ipynb > Q1. Apply run length encoding for the following string and compress it. "eeeeeeerrrrrrrrrrrrrrrrttttttttttttttttiiiiiiiiffffffeft"

+ Code + Markdown | Run All | Clear Outputs of All Cells | Outline ··· Python 3.8.9 64-bit

Q1. Apply run length encoding for the following string and compress it. "eeeeeeerrrrrrrrrrrrrrrrttttttttttttttttiiiiiiiiffffffeft"

```
def lengthEncoding(st):
    n = len(st)
    i = 0
    while i < n- 1:
        count = 1
        while (i < n - 1 and
            st[i] == st[i + 1]):
            count += 1
            i += 1
        i += 1
        print(st[i - 1] + str(count), end = "")

st = "eeeeeeerrrrrrrrrrrrrrrrttttttttttttttttiiiiiiiiffffffeft"
print("The Run Length Encoding of string ", st, " = ", end=" ")
lengthEncoding(st)
```

[2] Python

```
... The Run Length Encoding of string  eeeeeeerrrrrrrrrrrrrrrrttttttttttttttttiiiiiiiiffffffeft  =  e7r16t14i8f5e1f1
```

19BCE1413_Lab_5.ipynb

Users > vaibhavagarwal > sem 6 > web mining > lab > lab 5 > 19BCE1413_Lab_5.ipynb > Q1. Apply run length encoding for the following string and compress it. "eeeeeeerrrrrrrrrrrrrrrrttttttttttttttttiiiiiiiiffffffeft"

+ Code + Markdown | Run All | Clear Outputs of All Cells | Outline ··· Python 3.8.9 64-bit

Q2. Consider the following Inverted Index File with Terms, Occurrences and Posting List

i. Apply Binary coding for term “Mercury” (apply for all doc ids)

```
def Binary(n):
    num = bin(n)
    num = num[2:]
    return num

print("The Binary Encoding for Mercury is ", end=" ")
Mercury = [1, 2, 3, 7, 9, 10]
for N in Mercury:
    print(Binary(N), end=" ")
```

Python

```
... The Binary Encoding for Mercury is  1 10 11 111 1001 1010
```

ii. Apply Unary coding for term “Fiber”

```
def unaryCoding(arr):
    for N in arr:
        A = []
        for i in range(N):
            A.append(1)
        A.append(0)
        B = [str(k) for k in A]
        C = "".join(B)
        print(C, end=" ")

Fiber = [1,3,5,7,19,20]
print("The Unary Coding for Fiber is ", end=" ")
unaryCoding(Fiber)
```

[4] Python

```
... The Unary Coding for Fiber is  10 1110 111110 11111110 11111111111111111110 11111111111111111110
```

iii. Apply Elias Gamma Encoding for term “Airtel”

```python
from math import log
log2 = lambda x: log(x, 2)

def Unary(x):
    return (x-1)*'0'+'1'

def Binary(x, l = 1):
    s = '{0:0%db}' % l
    return s.format(x)

def Elias_Gamma(x):
    if(x == 0):
        return '0'
    n = 1 + int(log2(x))
    b = x - 2**(int(log2(x)))
    l = int(log2(x))
    return Unary(n) + Binary(b, l)

Airtel = [12, 17, 25, 148, 156, 159, 172]
print("The Elias Delta Encoding for Airtel is ", end=" ")
for N in Airtel:
    print(Elias_Gamma(N), end=" ")
```

[5] Python

```
... The Elias Delta Encoding for Airtel is  0001100 000010001 000011001 000000010010100 000000010011100 000000010011111 000000010101100
```

iv. Apply Elias Delta Decoding for “000010000”

```python
import math

def Elias_Delta_Decoding(x):
    x = list(x)
    L = 0
    while True:
        if not x[L] == '0':
            break
        L = L + 1
    x = x[2*L+1:]
    x.reverse()
    x.insert(0, '1')
    n = 0

    # Converting binary to integer
    for i in range(len(x)):
        if x[i] == '1':
            n = n+math.pow(2, i)
    return int(n)

x = '000010000'
print("The Elias Delta Decoding for 000010000 is ", Elias_Delta_Decoding(x))
```

[6] Python

```
... The Elias Delta Decoding for 000010000 is  1
```

v. Apply Elias Delta Encoding for term “Venus”

```python
from math import log
from math import floor

def Binary_Representation_Without_MSB(x):
    binary = "{0:b}".format(int(x))
    binary_without_MSB = binary[1:]
    return binary_without_MSB

def EliasGammaEncode(k):
    if (k == 0):
        return '0'
    N = 1 + floor(log(k, 2))
    Unary = (N-1)*'0'+'1'
    return Unary + Binary_Representation_Without_MSB(k)

def EliasDeltaEncode(x):
    Gamma = EliasGammaEncode(1 + floor(log(x, 2)))
    binary_without_MSB = Binary_Representation_Without_MSB(x)
    return Gamma+binary_without_MSB

print("The Elias Delta Encoding for Venus is ", end=" ")
Venus = [23, 45, 78, 122, 145]
for N in Venus:
    print(EliasDeltaEncode(N), end=" ")
```

[16] Python

```
... The Elias Delta Encoding for Venus is  001010111 0011001101 00111001110 00111111010 00010000010001
```

vi. Apply Elias Delta Decoding for “00101001”

```python
import math

def Elias_Delta_Decoding(x):
    x = list(x)
    L = 0
    while True:
        if not x[L] == '0':
            break
        L = L + 1
    x = x[2*L+1:]
    x.reverse()
    x.insert(0, '1')
    n = 0
    for i in range(len(x)):
        if x[i] == '1':
            n = n+math.pow(2, i)
    return int(n)

x = '00101001'
print("The Elias Delta Decoding for 00101001 is ", Elias_Delta_Decoding(x))
```

[12] Python

```
... The Elias Delta Decoding for 00101001 is  3
```

vii. Apply Variable Byte Encoding for “Samsung”. (Use doc ids gap)

```python
def toBinary(number):
    bin_num = bin(number)
    bin_num = bin_num[2:]
    return bin_num

def variableByteEncoding(number):
    s = toBinary(number)
    result = ""

    while len(s) > 0:
        if len(s) > 7:
            term = s[-7:]
            s = s[:-7]
        else:
            term = s
            s = ""
            term = ("0" * (7 - len(term))) + term

        if len(result) == 0:
            result = term + "0"
        else:
            result = term + "1" + result
    return result

print("The Variable Byte Encoding for Samsung : ", end=" ")
Samsung = [2, 12, 34544, 34574, 35569]
for i in range(len(Samsung)-1):
    if i == 0:
        print(variableByteEncoding(Samsung[i]), end=" ")
    else:
        x = int(variableByteEncoding(Samsung[i+1]))
        y = int(variableByteEncoding(Samsung[i]))
        print(x - y, end=" ")
```

[11] Python

```
... The Variable Byte Encoding for Samsung :  00000100 10100011011110890000 8988911100 8991011088910
```