VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

**Name: Tanmay Mahajan Reg. No.-19BCE1735**

| | | | | | |
|---|---|---|---|---|---|
| Programme | : | **B.Tech** | Semester | : | **Win Sem 21-22** |
| Course | : | **Web Mining** | Code | : | **CSE3024** |
| Faculty | : | **Dr.Bhuvaneswari A** | Slot | : | **L7+L8** |
| Date | : | **08-02-2022** | Marks | : | **10 Points** |

## Exercise 5: Index Compression Techniques

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

"eeeeeeerrrrrrrrrrrrrrrrrttttttttttttttttiiiiiiiiffffffeft"

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)

---

Reference:

https://gist.github.com/utahta/740609/fa7270b0674b6815f9c760570377988707f36717

https://nlp.stanford.edu/IR-book/html/htmledition/variable-byte-codes-1.html

https://nlp.stanford.edu/IR-book/html/htmledition/postings-file-compression-1.html

```
def binary(n) :
    """
    Given an integer number returns the equivalent binary string.
    """

    # Convert to binary string
    num = bin(n)

    # Remove the `0b` which is present in the front of the string
    num = num[2:]

    return num
```

```
def eliasDeltaEncoding(n) :
    """
    Given an integer number `n`, we encode the number using the `Elias Delta Enc
oding` scheme, and return the compressed value as a string.
    """

    # Zero is already encoded
    if n == 0 :
        return "0"

    # Find the gamma code for (1 + log2(n))
    num1 = 1 + int(math.log2(n))
    num1 = eliasGammaEncoding(num1)

    # Number in binary form after removing the MSB
    num2 = binary(n)
    num2 = str(num2)[1:]

    # Combine the gamma code and the other code value
    num = num1 + num2

    return num
```

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

```
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 = "eeeeeeerrrrrrrrrrrrrrrttttttttttttttiiiiiiiiffffeft"
print("Run Length Encoding of", st, " = ", end=" ")
lengthEncoding(st)
```

```
Run Length Encoding of eeeeeeerrrrrrrrrrrrrrrttttttttttttttiiiiiiiiffffeft  =  e7r16t1
```

## Q2. 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)

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

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

```
Binary Encoding for Mercury =  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("Unary Coding for Fiber = ", end=" ")
unaryCoding(Fiber)
```

```
Unary Coding for Fiber =  10 1110 111110 11111110 1111111111111111110 11111111111111111
```

## iii. Apply Elias Gamma Encoding for term “Airtel”

```
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("Elias Delta Encoding for Airtel = ", end=" ")
for N in Airtel:
    print(Elias_Gamma(N), end=" ")
```

```
Elias Delta Encoding for Airtel =  0001100 000010001 000011001 000000010010100 00000001
```

## iv. Apply Elias Delta Decoding for “000010000”

```
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("Elias Delta Decoding for 000010000 = ", Elias_Delta_Decoding(x))
```

```
 Elias Delta Decoding for 000010000 =  1
```

## v. Apply Elias Delta Encoding for term “Venus”

```
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(k, 2)))
    binary_without_MSB = Binary_Representation_Without_MSB(k)
```

```
    return Gamma+binary_without_MSB

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

```
 Elias Delta Encoding for Venus =  00100110 00100110 00100110 00100110 00100110
```

## vi. Apply Elias Delta Decoding for “00101001”

```
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("Elias Delta Decoding for 00101001 = ", Elias_Delta_Decoding(x))
```

```
 Elias Delta Decoding for 00101001 =  3
```

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

```
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("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=" ")
```

```
Variable Byte Encoding for Samsung =  00000100 1010001101111089000 8988911100 899101108
```