CSE3024 - WEB MINING (L41 + L42)

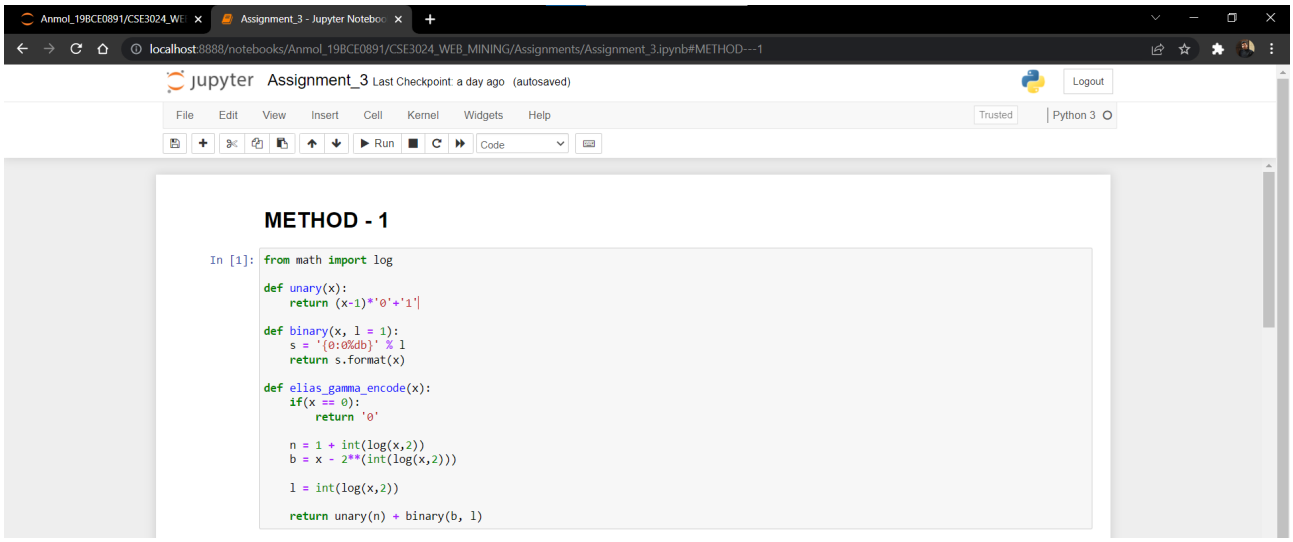
NAME – ANMOL
REG. NO. - 19BCE0891

DIGITAL ASSIGNMENT – 3

METHOD - 1

Algorithm (encoding)

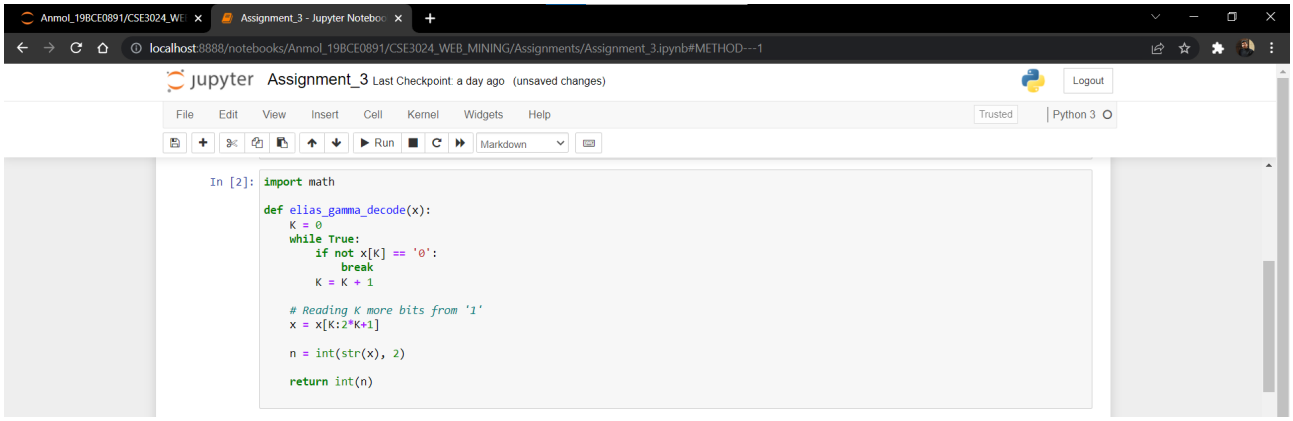
1. Find the largest N, with 2^N <= **X**(greater power of 2).
2. Encode N using Unary coding (i.e N zeroes followed by a one).
3. Append the integer ($X - 2^N$) using N digits in Binary.



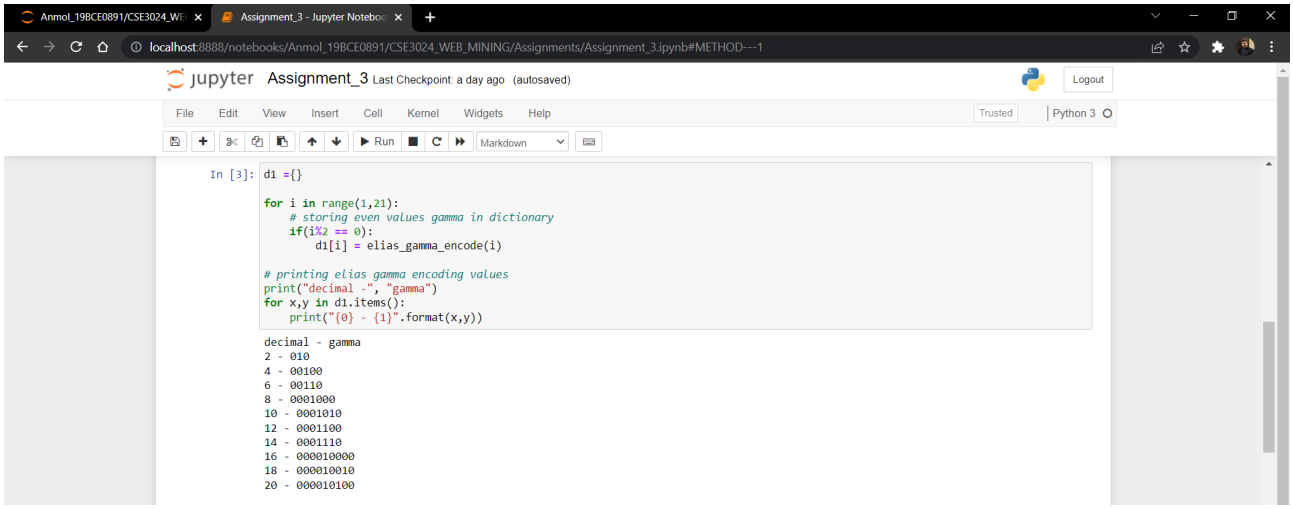
Algorithm (decoding)

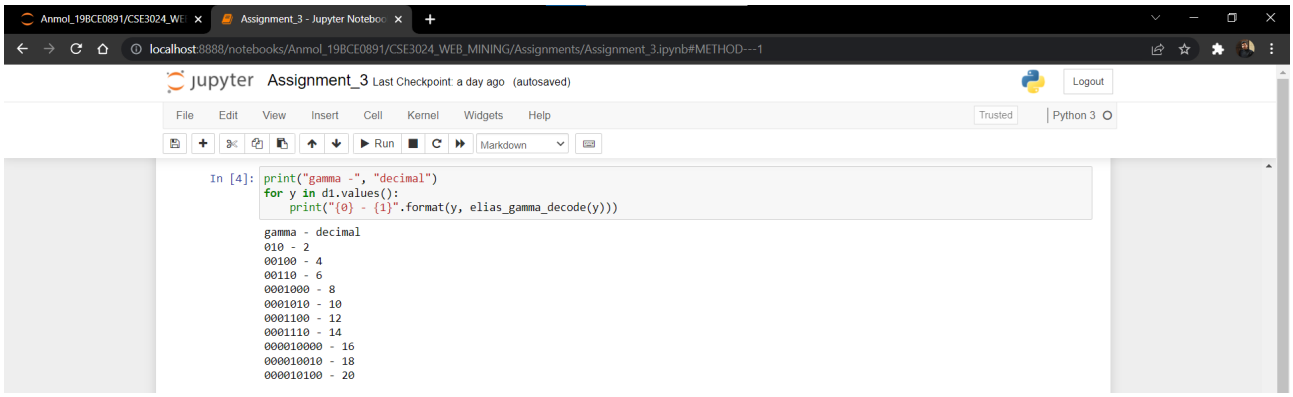
We decode an Elias gamma-coded integer in two steps:

1.Read and count zeroes from the stream until we reach the first one. Call this count of zeroes K.
2.Consider the one that was reached to be the first digit of the integer, with a value of 2K, read the remaining K bits of the integer.



OUTPUT (Method – 1)



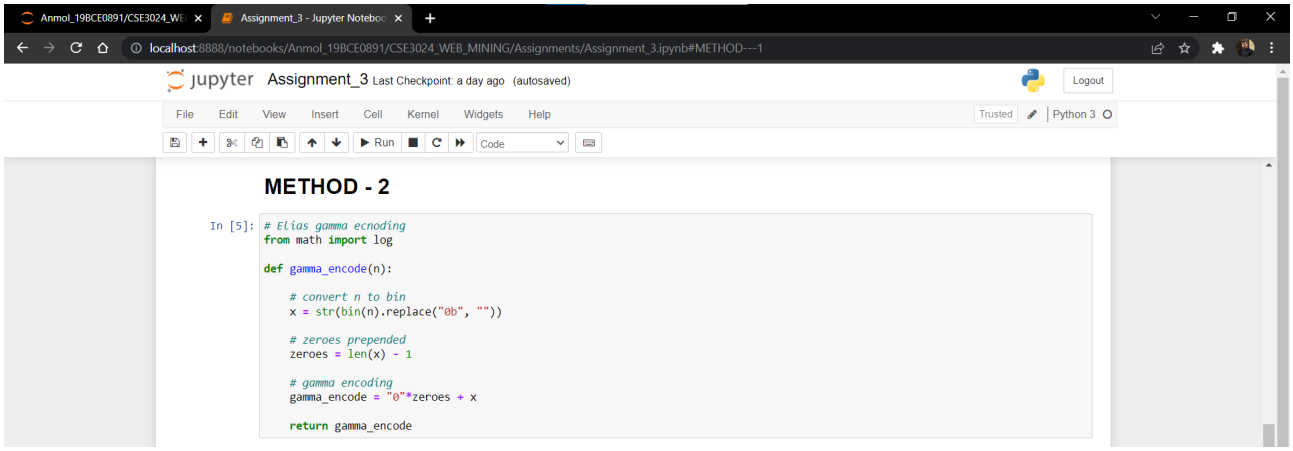


METHOD – 2

Algorithm (encoding)

The coding can also be described with the following two steps:

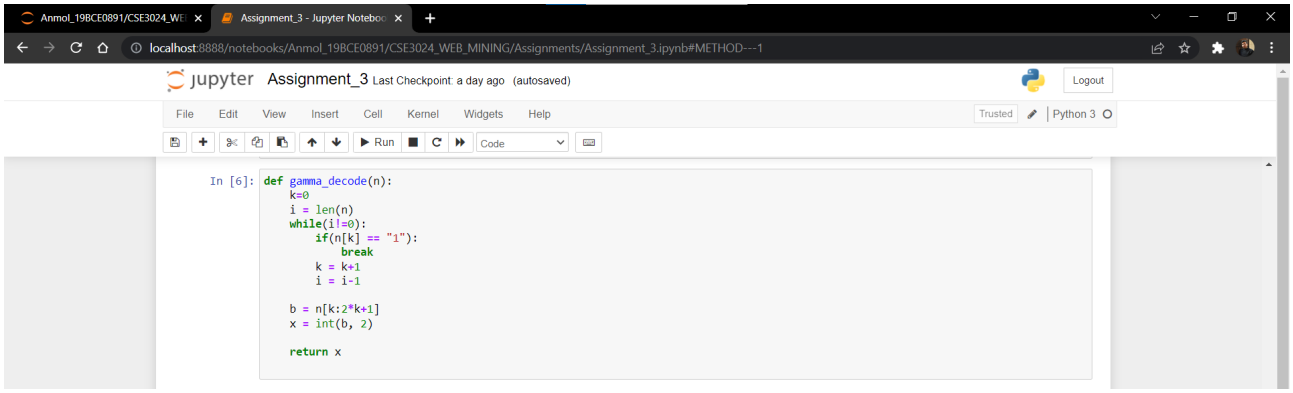
1. Write x in binary.
2. Subtract 1 from the number of bits written in step 1 and prepend that many zeros.

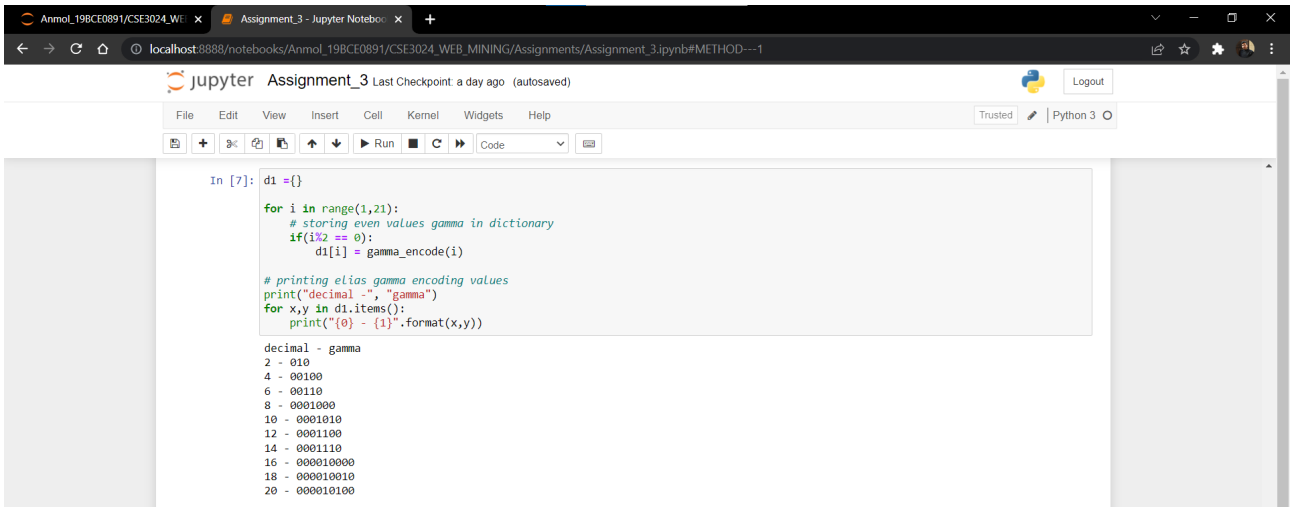


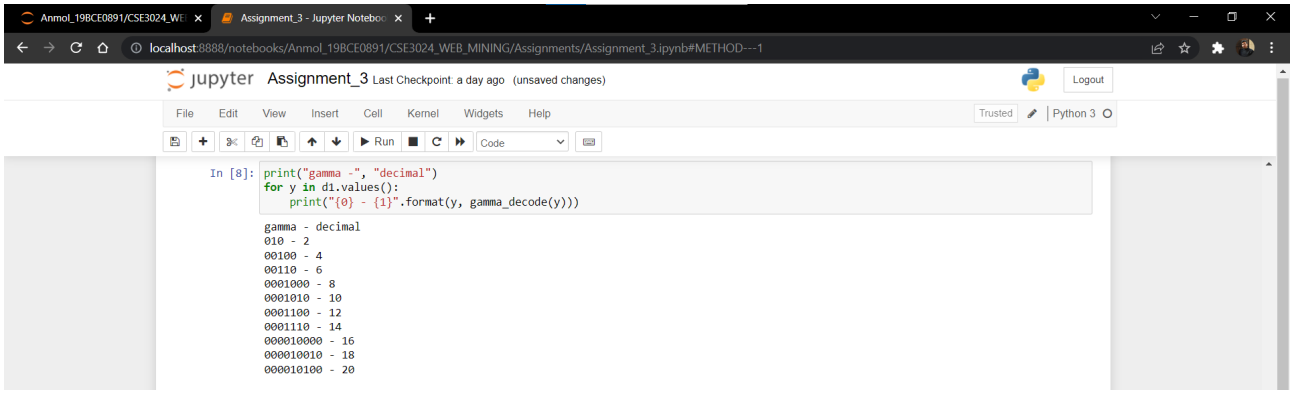
Algorithm (decoding)

We decode an Elias gamma-coded integer in two steps:

1.Read and count zeroes from the stream until we reach the first one. Call this count of zeroes K.
2.Consider the one that was reached to be the first digit of the integer, with a value of 2K, read the remaining K bits of the integer.







CODE (Method – 1)

```
from math import log

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_encode(x):
    if(x == 0):
        return '0'

    n = 1 + int(log(x,2))
    b = x - 2**(int(log(x,2)))

    l = int(log(x,2))

    return unary(n) + binary(b, l)
```

```
import math

def elias_gamma_decode(x):
    K = 0
    while True:
        if not x[K] == '0':
            break
        K = K + 1

    # Reading K more bits from '1'
    x = x[K:2*K+1]

    n = int(str(x), 2)

    return int(n)

d1 ={}

for i in range(1,21):
    # storing even values gamma in dictionary
    if(i%2 == 0):
        d1[i] = elias_gamma_encode(i)

# printing elias gamma encoding values
print("decimal -", "gamma")
for x,y in d1.items():
    print("{0} – {1}".format(x,y))

print("gamma -", "decimal")
for y in d1.values():
    print("{0} - {1}".format(y, elias_gamma_decode(y)))
```

CODE (Method – 2)

```
# Elias gamma ecnoding
from math import log

def gamma_encode(n):

    # convert n to bin
    x = str(bin(n).replace("0b", ""))

    # zeroes prepended
    zeroes = len(x) - 1

    # gamma encoding
```

```
    gamma_encode = "0"*zeroes + x

    return gamma_encode



def gamma_decode(n):
    k=0
    i = len(n)
    while(i!=0):
        if(n[k] == "1"):
            break
        k = k+1
        i = i-1

    b = n[k:2*k+1]
    x = int(b, 2)

    return x


d1 ={}

for i in range(1,21):
    # storing even values gamma in dictionary
    if(i%2 == 0):
        d1[i] = gamma_encode(i)

# printing elias gamma encoding values
print("decimal -", "gamma")
for x,y in d1.items():
    print("{0} – {1}".format(x,y))


print("gamma -", "decimal")
for y in d1.values():
    print("{0} - {1}".format(y, gamma_decode(y)))
```