

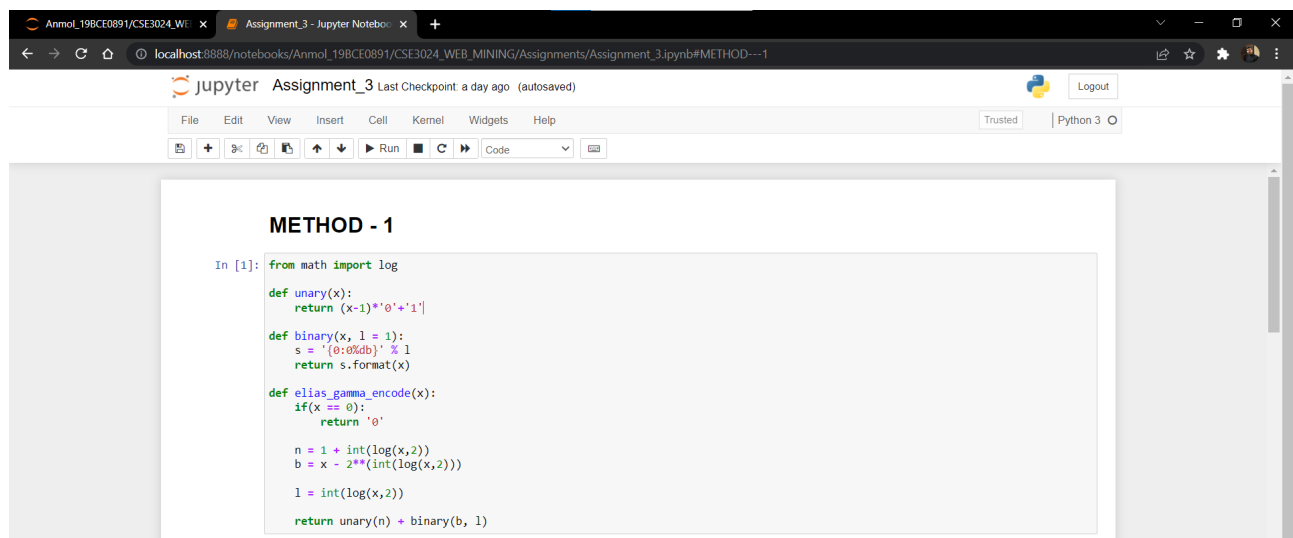
NAME – ANMOL
REG. NO. - 19BCE0891

DIGITAL ASSIGNMENT – 3

METHOD - 1

Algorithm (encoding)

1. Find the largest N , with $2^N \leq X$ (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.



The screenshot shows a Jupyter Notebook titled 'Assignment_3' running on a local host. The notebook contains a code cell with the following Python code:

```
In [1]: from math import log

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

def binary(x, l = 1):
    s = '{0:0{ld}}'.format(x, l)
    return s

def elias_gamma_encode(x):
    if(x == 0):
        return '0'

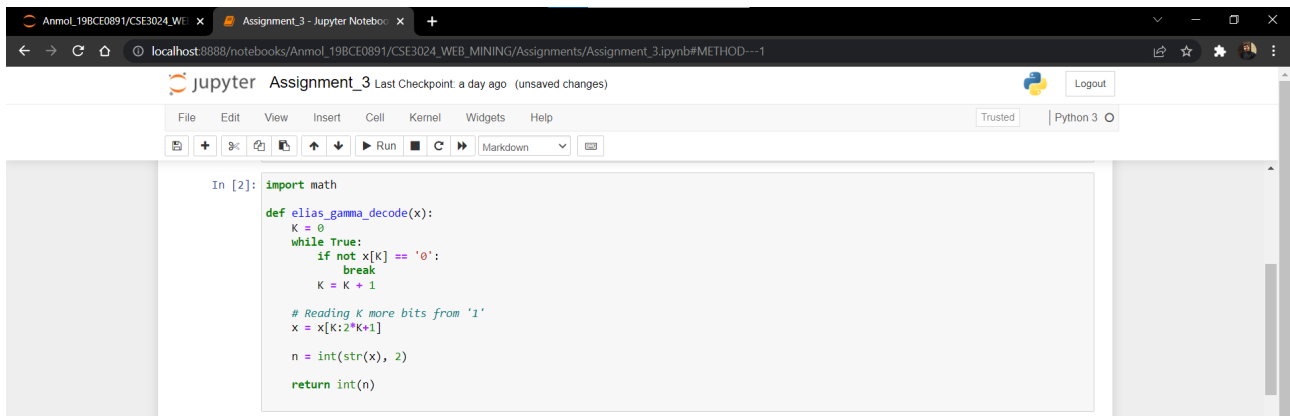
    n = 1 + int(log(x,2))
    b = x - 2**(int(log(x,2)))
    l = int(log(x,2))
    return unary(n) + binary(b, l)
```

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.

CSE3024 - WEB MINING (L41 + L42)



A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_3.ipynb#METHOD---1'. The notebook title is 'Assignment_3' with a status of 'Last Checkpoint: a day ago (unsaved changes)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar shows icons for file operations, running, and output. The code cell contains the following Python code:

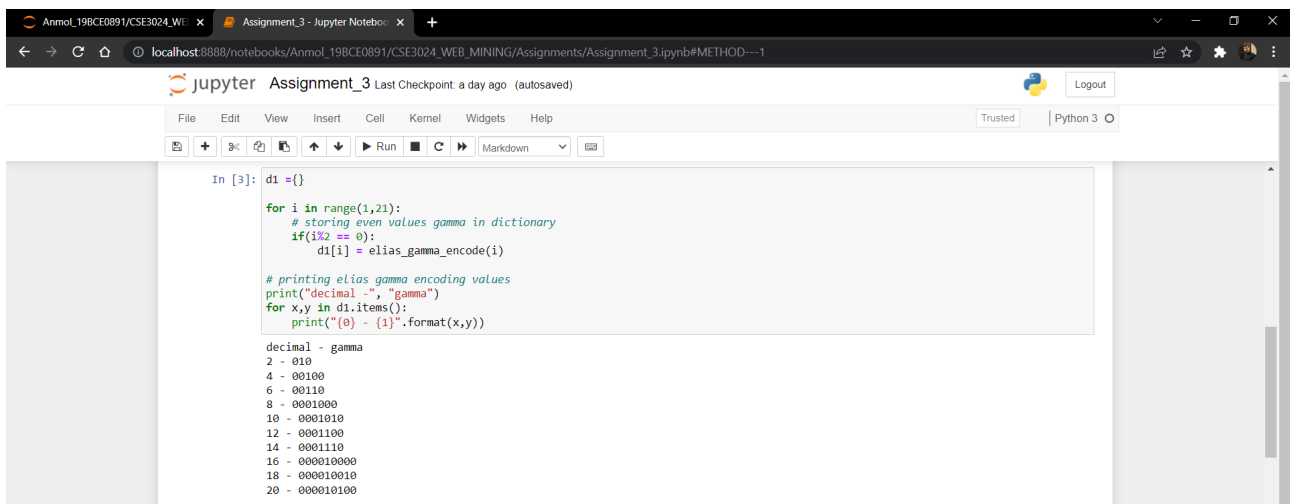
```
In [2]: import math

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)
```

OUTPUT (Method – 1)



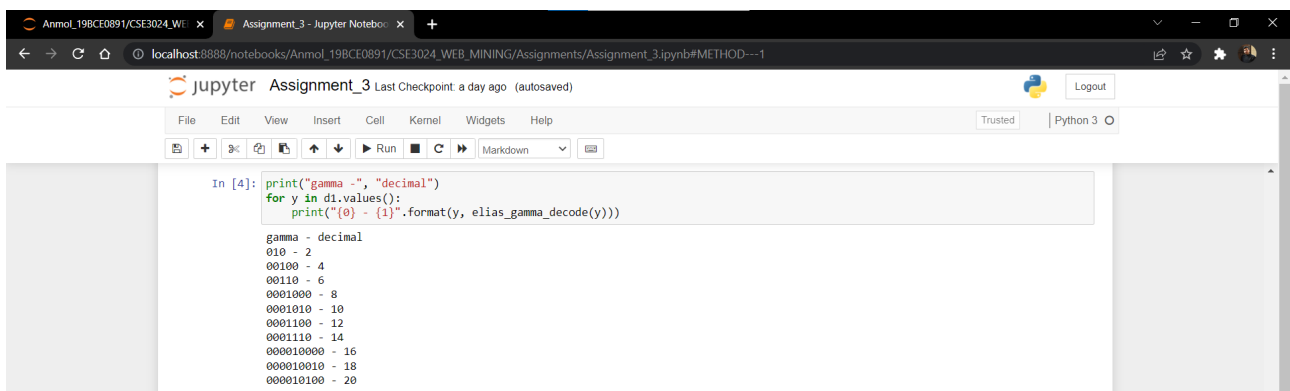
A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_3.ipynb#METHOD---1'. The notebook title is 'Assignment_3' with a status of 'Last Checkpoint: a day ago (autosaved)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar shows icons for file operations, running, and output. The code cell contains the following Python code:

```
In [3]: 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))

decimal - gamma
2 - 010
4 - 00100
6 - 00110
8 - 0001000
10 - 0001010
12 - 0001100
14 - 0001110
16 - 000010000
18 - 000010010
20 - 000010100
```



A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_3.ipynb#METHOD---1'. The notebook title is 'Assignment_3' with a status of 'Last Checkpoint: a day ago (autosaved)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar shows icons for file operations, running, and output. The code cell contains the following Python code:

```
In [4]: print("gamma -", "decimal")
for y in d1.values():
    print("{0} - {1}".format(y, elias_gamma_decode(y)))

gamma - decimal
010 - 2
00100 - 4
00110 - 6
0001000 - 8
0001010 - 10
0001100 - 12
0001110 - 14
000010000 - 16
000010010 - 18
000010100 - 20
```

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.



The screenshot shows a Jupyter Notebook interface with a single code cell. The cell is titled 'METHOD - 2' and contains the following Python code:

```
In [5]: # Elias gamma encoding
from math import log

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

    # zeroes prepended
    zeroes = len(x) - 1

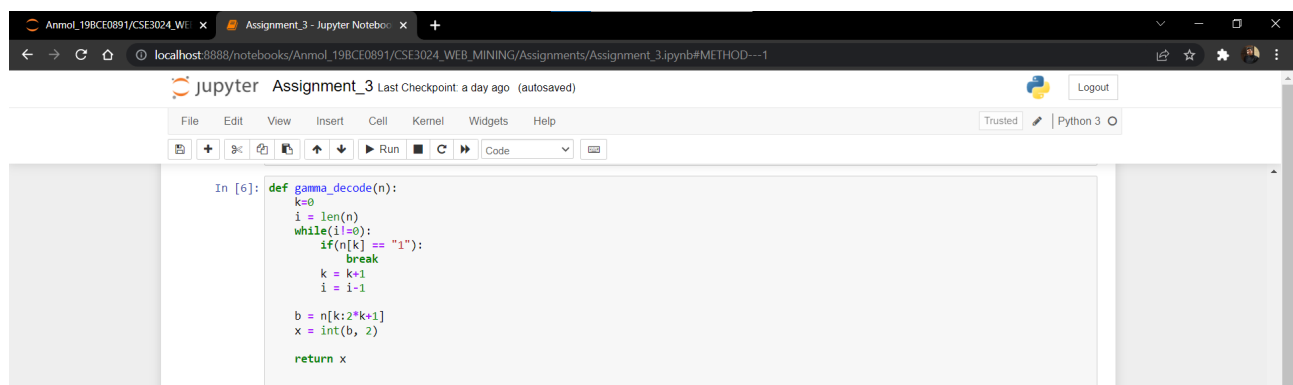
    # gamma encoding
    gamma_encode = "0"*zeroes + x

    return gamma_encode
```

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.



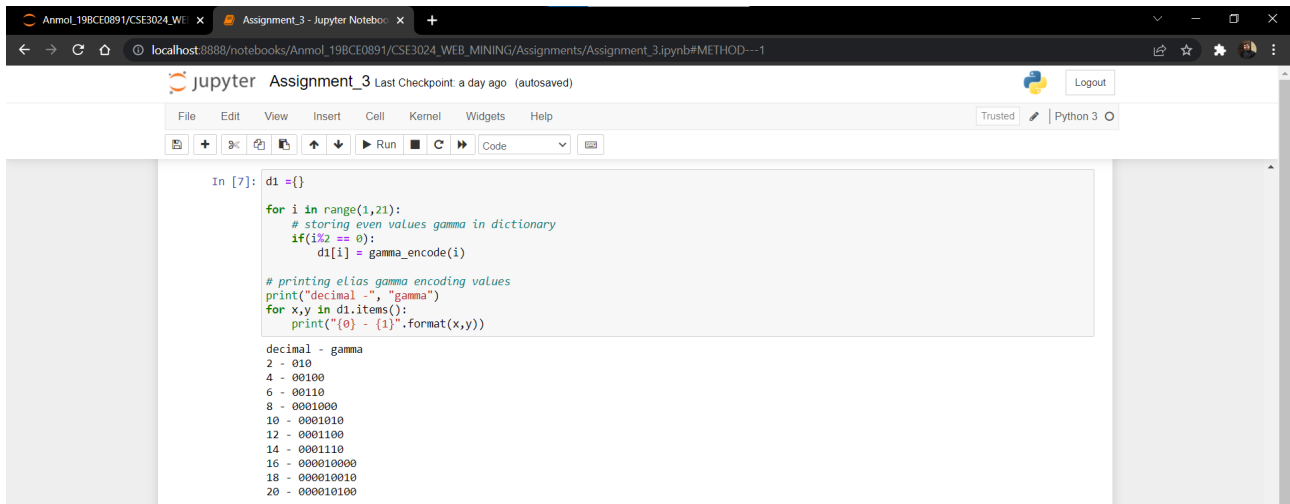
The screenshot shows a Jupyter Notebook interface with a single code cell. The cell is titled 'METHOD - 2' and contains the following Python code:

```
In [6]: 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
```

CSE3024 - WEB MINING (L41 + L42)



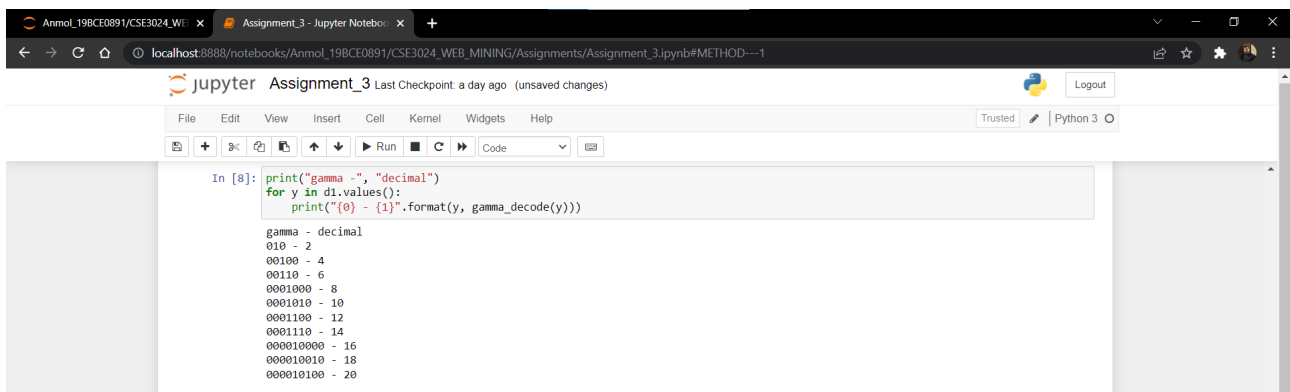
A Jupyter Notebook interface showing a Python script that encodes decimal values into gamma (binary) format. The script iterates over a range of 1 to 21, storing even values in a dictionary. The output shows a mapping of decimal values to their corresponding gamma values.

```
In [7]: 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))

decimal - gamma
2 - 010
4 - 00100
6 - 00110
8 - 0001000
10 - 0001010
12 - 0001100
14 - 0001110
16 - 000010000
18 - 000010010
20 - 000010100
```



A Jupyter Notebook interface showing a Python script that decodes gamma values back to decimal format. The script iterates over the values in the dictionary, printing the gamma value and its corresponding decimal value.

```
In [8]: print("gamma -", "decimal")
for y in d1.values():
    print("{0} - {1}".format(y, gamma_decode(y)))

gamma - decimal
010 - 2
00100 - 4
00110 - 6
0001000 - 8
0001010 - 10
0001100 - 12
0001110 - 14
000010000 - 16
000010010 - 18
000010100 - 20
```

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 encoding
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

```

CSE3024 - WEB MINING (L41 + L42)

```
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)))
```