INSTITUTO POLITECNICO NACIONAL

Escuela Superior de Cómputo

Programa primos

Teoría de la computación

Alumno:
Reyes Garnelo Uziel Bruno

Profesor: Genaro Juárez Martinez

Grupo: 5BM1

17 de abril de 2023

1. Introducción:

Se dice que el universo que componen todos los números es un universo infinito, y dentro de éste universo existe un subconjunto de números importantes, a este subconjunto de números se les conoce "números primos".

Estos números tienen la caracteristica de solo ser divisibles exactamente entre uno y si mismos. Encontrar estos números es una tarea cara computacionalmente, uno de los usos más comúnes y con más provecho e importancia es para cuestiones de cifrado en seguridad informatica.

La técnica que se uso en el desarrollo de la presente práctica fue "La criba de Eratóstenes", que parte del conocimiento que el número "2.es el primer número primo. Eliminando así todos los números multiplos de "2", posteriormente el siguiente número en la lista será un número primo, y el algoritmo elimina de la lista todos los multiplos del siguiente primo, y de esta forma consecutivamente.

2. Desarrollo:

Como primer paso, se utilizaron algunas librerias auxiliares que ayudaron al desarrollo, la optimización y la graficación del programa.

```
#Importando librerias
import matplotlib.pyplot as plt
import numpy as np
import re
```

- matplotlib: Libreria que proporciona herramientas de graficación.
- numpy: libreria que proporciona estructuras de datos y funciones para el manejo de numeros optimizadas en lenguajes de bajo nivel.
- re: Función brinda herramientas para el uso de expresiones regulares.

Primeramente se crea un bucle que ejecuta una vez el las funciones que integran el programa.

```
print('¡Practica 2!')
bandera = True
while bandera:
    n = inicializacion()
    calcularPrimos(n)
    graficar()
    menu = int(input('Desea calcular otra \'n\'? \n 0.No\n1.Si'))
    if menu == 0:
        bandera = False
```

Una vez iniciado el programa, se le pregunta al usuario si el número "n", se usará de forma manual o automática. Si el valor de "n"será de forma automática se le informa cual se decidió.

```
¡Practica 2!
El universo se creará de forma:
0. Automática
1. Manual
El valor de 'n' es: 10000000
```

```
def inicializacion():
    # Establecimiento de parámetros

    # Tipo de universo
    print('El universo se creará de forma: ')
    print('0. Automática')
    print('1. Manual')

    modo = int(input('?. '))

    # Asignación de n
    n = -1

    if modo == 0:
        n = np.randint(3, 10000001)
    else:
        while n == -1:
            n_temporal = int(input('¿Cuál será el valor de \'n\'? (Rango: 0-10,000,000): '))
            if n_temporal >= 0 and n_temporal <= 10000000:
                n = n_temporal

    print('El valor de \'n\' es: '+str(n))
    return n
```

La función 'calcularPrimos', se encarga de encontrar los números primos de 2 hasta n y escribirlas en un archivo.

Primero crea una lista de números consecutivos de 3 a n de cada dos números Ej: con n = 10 la lista que crea es: [3,5,7,9] así, se evitan todos los multiplos de 2 que ya sabemos que no serán primos.

Posteriormente se realiza la criba de Eratostenes, haciendo recorriendo una vez toda la lista buscando los múltiplos del primo a iterar, y así sucesivamente hasta vaciar la lista, agregando el primo encontrado al archivo y eliminando los múltiplos de éste mismo de la lista.

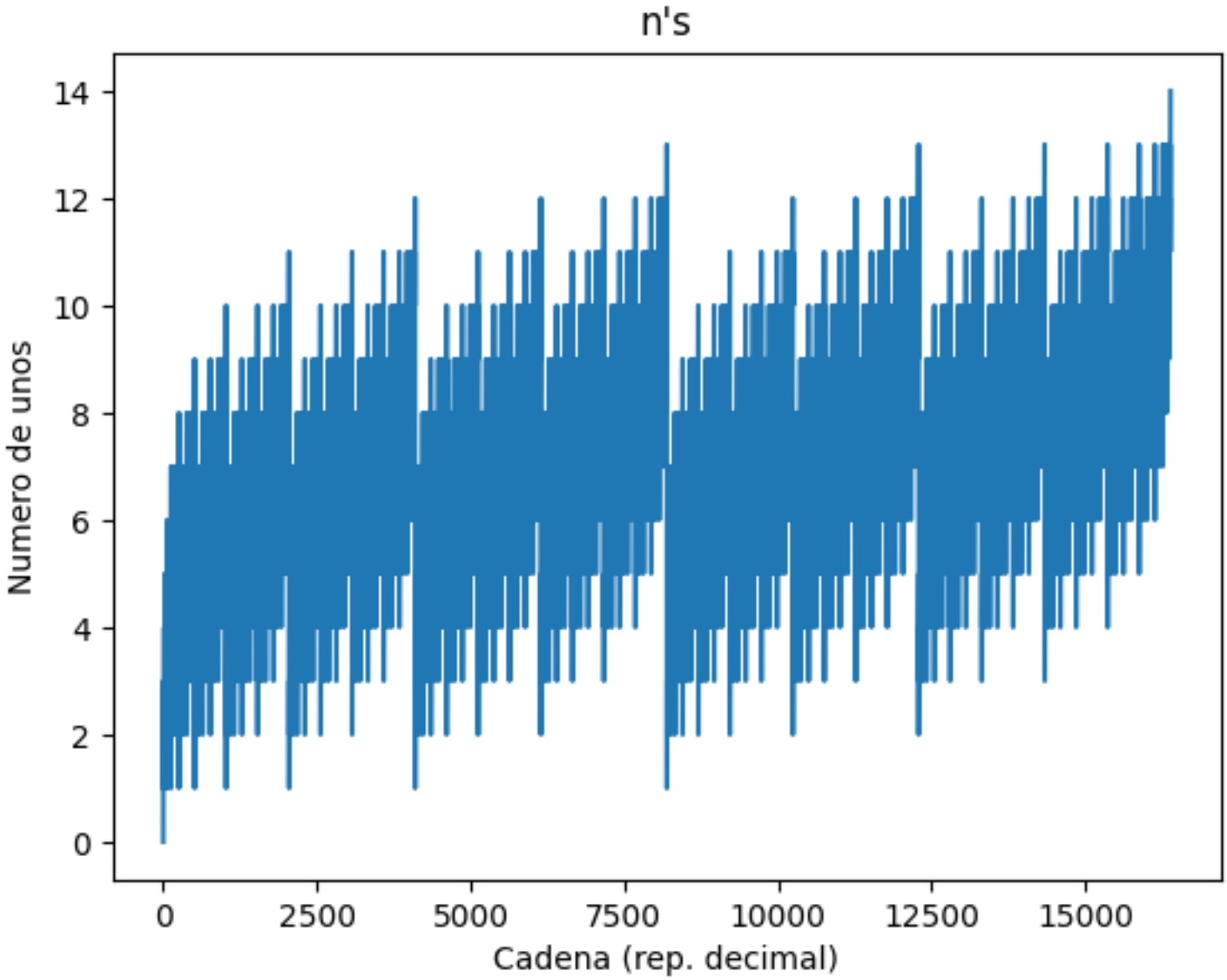
```
def calcularPrimos(n):
    # Creacion de la lista de números
    universo = np.arange(3,n+1,2)

    # Criba de Eratostenes
    primo = 3
    banderaPrimos = True

    with open('./Primos.txt', mode='w') as archivo:
        archivo.write('{')
        archivo.write('2')
        while banderaPrimos:
            universo = np.delete(universo, 0)
            universo_temporal = np.copy(universo)
            archivo.write(str(primo)+',')
            print('Eliminando multiplos de '+str(primo))
            for i in range(universo.shape[0]):
                if universo[i]%primo == 0:
                    universo_temporal = np.delete(universo_temporal, np.where(universo_temporal==unive
            universo = np.copy(universo_temporal)

            if universo.shape[0] == 0:
                banderaPrimos = False
            else:
                primo = universo[0]
        archivo.write('}')
    archivo.close()
```

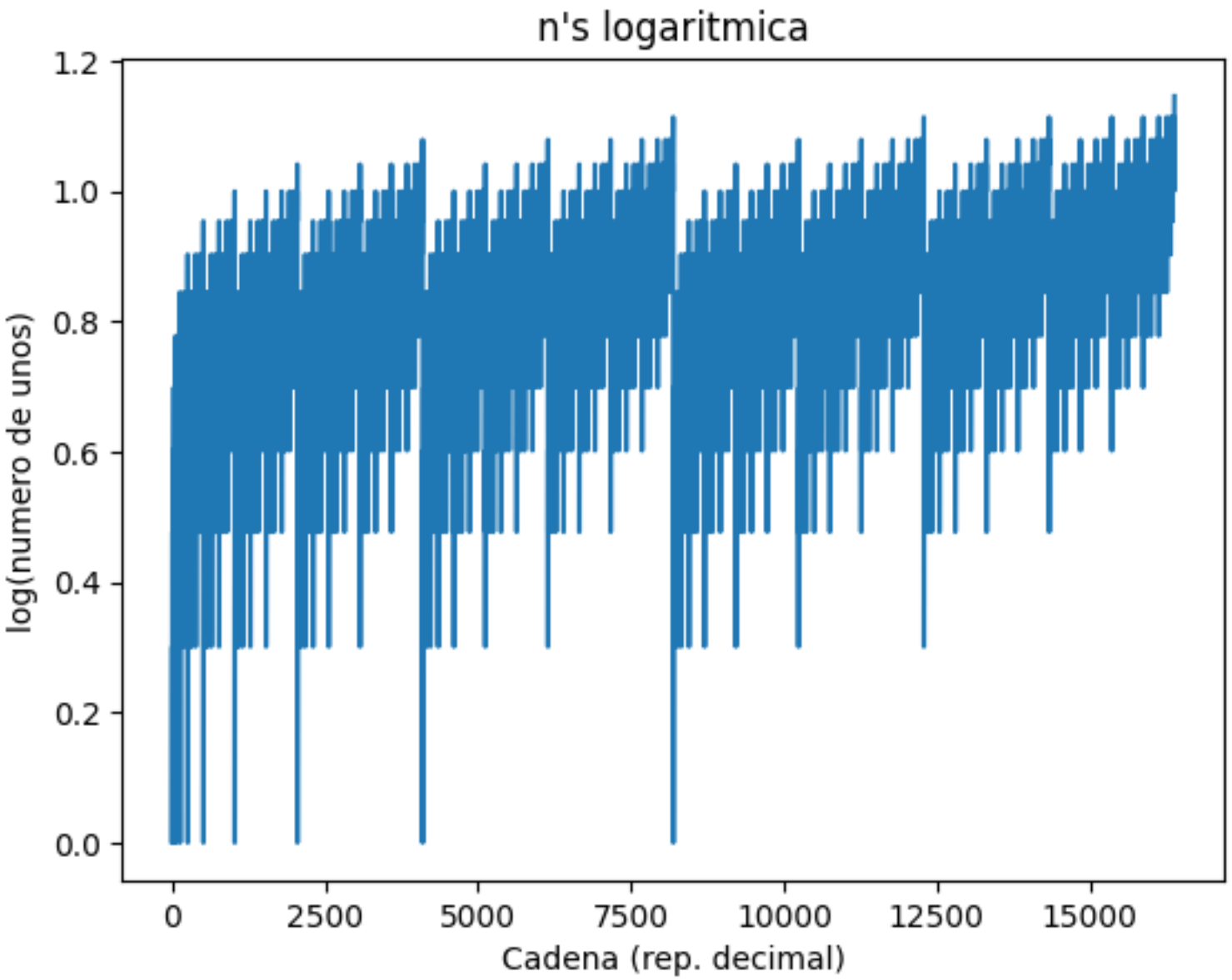
La función 'graficar', se encarga de leer el archivo anteriormente generado, y usando una expresión regular, encuentra todos los números y los almacena en una lista de números decimales, posteriormente toma esta lista y usando una función auxiliar llamada "decimal$_{ab}$*inaria*"*creaotralistadedecimalesparaposteriormentealmece*

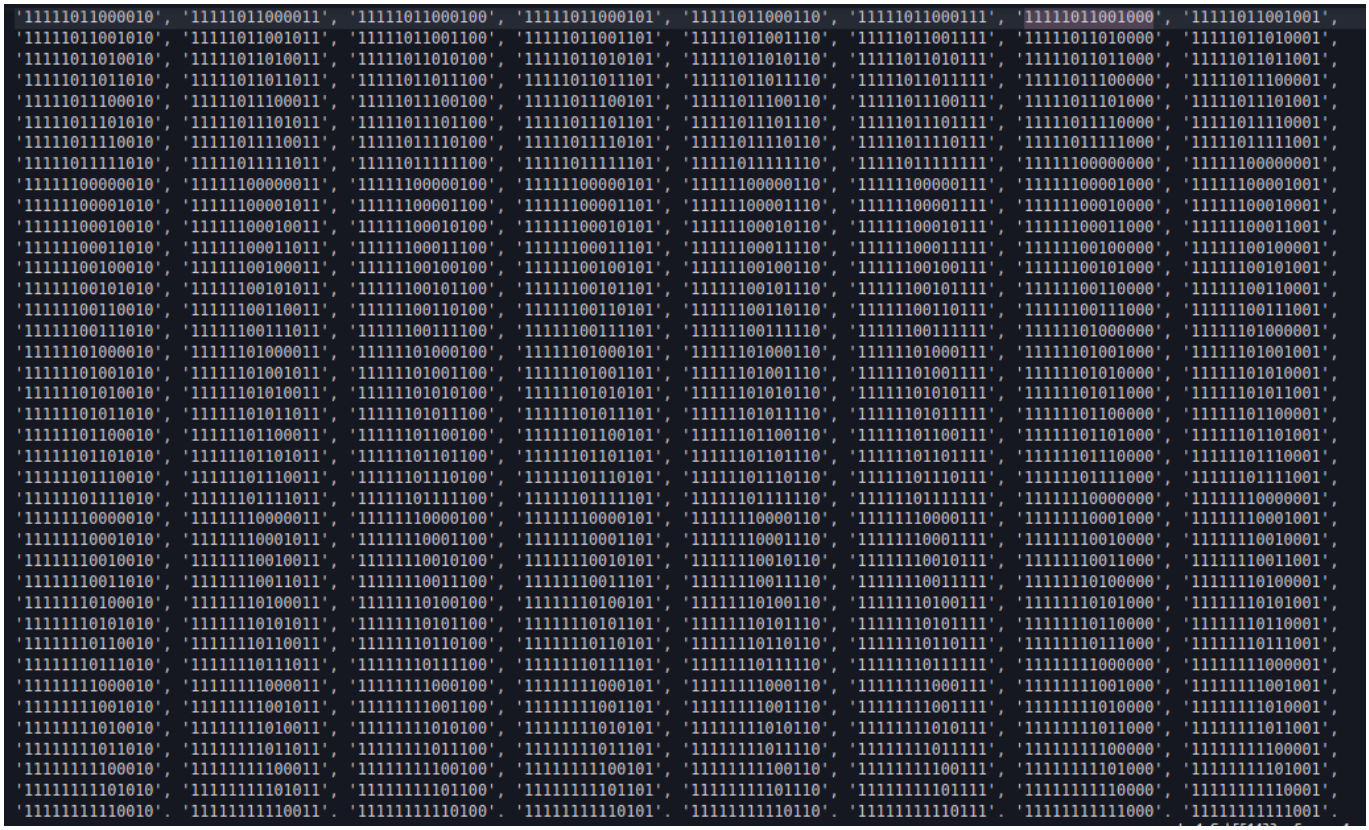
```
def graficar(n):
    with open("Practical.txt", "r") as archivo_universo:
        for lines in archivo_universo:
            line = lines.replace('{','').replace('}','').replace('\'','').replace(' ','').split(',')
    archivo_universo.close()
    line.pop()

    numeros = np.array([])

    for item in line[::-1]:
        if len(item) == n:
            numeros = np.append(numeros , item)

    n_unos = np.array([contar_unos(binario) for binario in numeros])

    np.set_printoptions(precision=4)
    n_unos_log = np.emath.log10(n_unos)
    # Eje x :'Cadena'
    decimal = np.array([binario_a_decimal(binario) for binario in numeros])

    plt.plot(decimal, n_unos)
    plt.title('n\'s')
    plt.xlabel('Cadena (rep. decimal)')
    plt.ylabel('Numero de unos')
    plt.show()

    plt.plot(decimal, n_unos_log)
    plt.title('n\'s logaritmica')
    plt.xlabel('Cadena (rep. decimal)')
    plt.ylabel('log(numero de unos)')
    plt.show()
```

```python
def contar_unos(cadena):
    # Inicializar el contador de unos
    contador_unos = 0

    # Recorrer cada dígito de la cadena
    for digito in cadena:
        # Si el dígito es igual a "1", sumar 1 al contador de unos
        if digito == '1':
            contador_unos += 1

    # Devolver el contador de unos
    return contador_unos

def binario_a_decimal(numero_binario):
    numero_decimal = 0

    for posicion, digito_string in enumerate(numero_binario[::-1]):
        numero_decimal += int(digito_string) * 2 ** posicion

    return numero_decimal
```

La salida del programa son las siguientes gráficas y el archivo de texto.




n's logaritmica
log(numero de unos)
Cadena (rep. decimal)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
2500
5000
7500
10000
12500
15000


'11111011000010', '11111011000011', '11111011000100', '11111011000101', '11111011000110', '11111011000111', '11111011001000', '11111011001001',
'11111011001010', '11111011001011', '11111011001100', '11111011001101', '11111011001110', '11111011001111', '11111011010000', '11111011010001',
'11111011010010', '11111011010011', '11111011010100', '11111011010101', '11111011010110', '11111011010111', '11111011011000', '11111011011001',
'11111011011010', '11111011011011', '11111011011100', '11111011011101', '11111011011110', '11111011011111', '11111011100000', '11111011100001',
'11111011100010', '11111011100011', '11111011100100', '11111011100101', '11111011100110', '11111011100111', '11111011101000', '11111011101001',
'11111011101010', '11111011101011', '11111011101100', '11111011101101', '11111011101110', '11111011101111', '11111011110000', '11111011110001',
'11111011110010', '11111011110011', '11111011110100', '11111011110101', '11111011110110', '11111011110111', '11111011111000', '11111011111001',
'11111011111010', '11111011111011', '11111011111100', '11111011111101', '11111011111110', '11111011111111', '11111100000000', '11111100000001',
'11111100000010', '11111100000011', '11111100000100', '11111100000101', '11111100000110', '11111100000111', '11111100001000', '11111100001001',
'11111100001010', '11111100001011', '11111100001100', '11111100001101', '11111100001110', '11111100001111', '11111100010000', '11111100010001',
'11111100010010', '11111100010011', '11111100010100', '11111100010101', '11111100010110', '11111100010111', '11111100011000', '11111100011001',
'11111100011010', '11111100011011', '11111100011100', '11111100011101', '11111100011110', '11111100011111', '11111100100000', '11111100100001',
'11111100100010', '11111100100011', '11111100100100', '11111100100101', '11111100100110', '11111100100111', '11111100101000', '11111100101001',
'11111100101010', '11111100101011', '11111100101100', '11111100101101', '11111100101110', '11111100101111', '11111100110000', '11111100110001',
'11111100110010', '11111100110011', '11111100110100', '11111100110101', '11111100110110', '11111100110111', '11111100111000', '11111100111001',
'11111100111010', '11111100111011', '11111100111100', '11111100111101', '11111100111110', '11111100111111', '11111101000000', '11111101000001',
'11111101000010', '11111101000011', '11111101000100', '11111101000101', '11111101000110', '11111101000111', '11111101001000', '11111101001001',
'11111101001010', '11111101001011', '11111101001100', '11111101001101', '11111101001110', '11111101001111', '11111101010000', '11111101010001',
'11111101010010', '11111101010011', '11111101010100', '11111101010101', '11111101010110', '11111101010111', '11111101011000', '11111101011001',
'11111101011010', '11111101011011', '11111101011100', '11111101011101', '11111101011110', '11111101011111', '11111101100000', '11111101100001',
'11111101100010', '11111101100011', '11111101100100', '11111101100101', '11111101100110', '11111101100111', '11111101101000', '11111101101001',
'11111101101010', '11111101101011', '11111101101100', '11111101101101', '11111101101110', '11111101101111', '11111101110000', '11111101110001',
'11111101110010', '11111101110011', '11111101110100', '11111101110101', '11111101110110', '11111101110111', '11111101111000', '11111101111001',
'11111101111010', '11111101111011', '11111101111100', '11111101111101', '11111101111110', '11111101111111', '11111110000000', '11111110000001',
'11111110000010', '11111110000011', '11111110000100', '11111110000101', '11111110000110', '11111110000111', '11111110001000', '11111110001001',
'11111110001010', '11111110001011', '11111110001100', '11111110001101', '11111110001110', '11111110001111', '11111110010000', '11111110010001',
'11111110010010', '11111110010011', '11111110010100', '11111110010101', '11111110010110', '11111110010111', '11111110011000', '11111110011001',
'11111110011010', '11111110011011', '11111110011100', '11111110011101', '11111110011110', '11111110011111', '11111110100000', '11111110100001',
'11111110100010', '11111110100011', '11111110100100', '11111110100101', '11111110100110', '11111110100111', '11111110101000', '11111110101001',
'11111110101010', '11111110101011', '11111110101100', '11111110101101', '11111110101110', '11111110101111', '11111110110000', '11111110110001',
'11111110110010', '11111110110011', '11111110110100', '11111110110101', '11111110110110', '11111110110111', '11111110111000', '11111110111001',
'11111110111010', '11111110111011', '11111110111100', '11111110111101', '11111110111110', '11111110111111', '11111111000000', '11111111000001',
'11111111000010', '11111111000011', '11111111000100', '11111111000101', '11111111000110', '11111111000111', '11111111001000', '11111111001001',
'11111111001010', '11111111001011', '11111111001100', '11111111001101', '11111111001110', '11111111001111', '11111111010000', '11111111010001',
'11111111010010', '11111111010011', '11111111010100', '11111111010101', '11111111010110', '11111111010111', '11111111011000', '11111111011001',
'11111111011010', '11111111011011', '11111111011100', '11111111011101', '11111111011110', '11111111011111', '11111111100000', '11111111100001',
'11111111100010', '11111111100011', '11111111100100', '11111111100101', '11111111100110', '11111111100111', '11111111101000', '11111111101001',
'11111111101010', '11111111101011', '11111111101100', '11111111101101', '11111111101110', '11111111101111', '11111111110000', '11111111110001',
'11111111110010', '11111111110011', '11111111110100', '11111111110101', '11111111110110', '11111111110111', '11111111111000', '11111111111001',