import sys

import matplotlib.pyplot as plt

import random

import numpy as np

def nrz\_l(inp):

inp1=list(inp)

res = []

for i in inp1:

if i=='0':

res.append(-1)

else:

res.append(1)

return res

def nrz\_i(inp):

nrzi=[]

if inp[0]=='1':

nrzi.append(1)

else:

nrzi.append(-1)

for x in range(len(inp[1:])):

if inp[x]=='0':

nrzi.append(-nrzi[x-1])

else:

nrzi.append(nrzi[x-1])

return nrzi

def manchester(inp):

inp1=list(inp)

manchester = []

for i in inp1:

if i=='0':

manchester.append(-1)

manchester.append(1)

else:

manchester.append(1)

manchester.append(-1)

return manchester

def Diff\_manchester(inp):

li=[]

if inp[0]=='1':

li.append(1)

li.append(-1)

else:

li.append(-1)

li.append(1)

for i in range(len(inp[1:])):

if inp[i]=='1':

li.append(li[2\*i-2])

li.append(li[2\*i-1])

else:

li.append(-li[2\*i-2])

li.append(-li[2\*i-1])

print(li)

return li

def AMI(inp):

pulse = True

ami = []

for x in inp:

if x=='1':

ami.append(1) if pulse else ami.append(-1)

pulse = not(pulse)

else:

ami.append(0)

return ami

def scr\_b8zs(array):

pulse = True

zeros = 0

b8zs = []

for x in array:

if x=='1':

b8zs.append(1) if pulse else b8zs.append(-1)

pulse = not(pulse)

zeros = 0

else:

b8zs.append(0)

zeros += 1

if zeros >= 8:

b8zs[-5] = 1 if not(pulse) else -1

b8zs[-4] = 1 if pulse else -1

b8zs[-2] = 1 if pulse else -1

b8zs[-1] = 1 if not(pulse) else -1

zeros = 0

return b8zs

def scr\_hdb3(array):

pulse = True

zeros, ones = 0 , 0

hdb3 = []

for x in array:

if x=='1':

hdb3.append(1) if pulse else hdb3.append(-1)

pulse = not(pulse)

zeros = 0

ones += 1

else:

hdb3.append(0)

zeros += 1

if zeros >= 4:

if ones%2 == 0:

hdb3[-4] = 1 if pulse else -1

hdb3[-1] = 1 if pulse else -1

pulse = not(pulse)

else:

hdb3[-1] = 1 if not(pulse) else -1

zeros = 0

ones = 0

return hdb3

def gen\_z(n):

bs=gen(n)

temp=random.randint(1,2)

if temp==1:

sub='0000'

else:

sub='00000000'

bs += sub

return bs

def gen(n):

bs=""

for i in range(n):

temp= str(random.randint(0, 1))

bs+=temp

return bs

def Palindrome(X, Y, m, n, lookup):

if m == 0 or n == 0:

return ""

if X[m - 1] == Y[n - 1]:

return Palindrome(X, Y, m - 1, n - 1, lookup) + X[m - 1]

if lookup[m - 1][n] > lookup[m][n - 1]:

return Palindrome(X, Y, m - 1, n, lookup)

return Palindrome(X, Y, m, n - 1, lookup)

if \_\_name\_\_=='\_\_main\_\_':

print("1-random data sequence\t\t2-random with consecutive zeros\nEnter your choice: ")

c1=int(input())

if c1==1:

print("Enter the size of Encoded Data : ")

size=int(input())

ip= gen(size)

else:

n=4

ip=gen\_z(n)

print("DATA STREAM: " ,ip)

print("\n1-NRZ-L\t\t2-NRZ-I\t\t3-Manchester\n4-Diff Manchester\t5-AMI\t6-Scrambled B8ZS(for 8-consecutive zeros)\t7-Scrambled HDB3(for 4-consecutive zeros)\nEnter your choice of encoding: ")

c=int(input())

if c==1:

encoded=nrz\_l(ip)

elif c==2:

encoded=nrz\_i(ip)

elif c==3:

encoded=manchester(ip)

elif c==4:

encoded=Diff\_manchester(ip)

elif c==5:

encoded=AMI(ip)

elif c==6:

encoded=scr\_b8zs(ip)

elif c==7:

encoded=scr\_hdb3(ip)

else:

print("Invalid choice\n")

# PRINT LONGEST PALINDROME

ip1= ip[::-1]

lookup = [[0 for x in range(len(ip) + 1)] for y in range(len(ip) + 1)]

print('\nThe longest palindromic subsequence is',Palindrome(ip, ip1, len(ip), len(ip), lookup))

#PLOTING THE ENCODED DATA STREAM

if c==3 or c==4: #manchester encoding has mid transitions

x = np.arange(0,len(ip)+0.5, 0.5)

x2 = np.arange(0,len(ip)+1)

else:

x = np.arange(0,len(encoded)+1)

x2 = x

plt.subplot(2,1,1)

plt.xlim(0, len(ip))

plt.ylim(-0.5, 1.5)

plt.ylabel('Value')

plt.title('Original')

array = [int(z) for z in ip]

for i in range(len(ip)):

plt.text(i+0.4, 1.2, array[i])

plt.grid()

plt.xticks(x2)

plt.step(x2, [array[0]]+array)

plt.subplot(2,1,2)

plt.xlim(0, len(array))

plt.ylim(-1.5, 1.5)

plt.ylabel('Value')

plt.title('Encoding')

for i in range(len(array)):

plt.text(i+0.4, 1.2, array[i])

plt.grid()

plt.xticks(x2)

plt.step(x, [encoded[0]]+encoded)

plt.show()