Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Do there exists some prefix binary code containing words of the following lengths?

[3,3,3,5,5,6,6,7,8,8,9,9,9]

Select one:

- Yes ✓

The given lengths **do** satisfy Kraft inequality

- No

Your answer is correct.

The correct answer is: Yes

```
def prefix_code(lst):
    output = []
    sumatorio = 0
    for i in lst:
        sumatorio += 2**(-i)
    if (sumatorio > 1):
        return output
    else:
        for index, i in enumerate(lst):
            if (index is 0):
                output.append(bin(0)[2:].zfill(i))
            else:
                for j in range(0, 2**i):
                    prefix = False
                    aux = bin(j)[2:].zfill(i)
                    for binword in output:
                        prefix = ( (aux.startswith(binword)) or (binword.startswith(aux)) )
                        if prefix:
                            break
                    if not prefix:
                        output.append(aux)
                        break
        return output

prova = [3,3,3,5,5,6,6,7,8,8,9,9,9]
binarywords = prefix_code(prova)
if not binarywords:
    print("No")
else:
    print("Yes")
    [print(binword) for binword in binarywords]
'''
Yes
000
001
010
01100
01101
011100
011101
0111100
01111010
01111011
011111000
011111001
011111010
'''
```

Question 2

Partially correct

Mark 0.67 out of 1.00

Remove flag

Of the following sets of binary words; which ones are codes?

Select one or more:

- ['00', '11', '001', '111', '00111', '01111', '10000']
- ['000', '001', '010', '0110', '01110', '01111', '10000'] ✓ This is a prefix set
- ['00','01','10','11','00011']
- ['00', '010', '011', '1000', '10010', '10011', '10100'] ✓ This is a prefix set
- ['00', '01', '10', '110', '111','10101']
- ['00','001']

You have correctly selected 2.

The correct answer is: ['00', '010', '011', '1000', '10010', '10011', '10100'], ['000', '001', '010', '0110', '01110', '01111', '10000'], ['00','001']

```
def prefix_code(code):
    for binword in code:
        for aux in code:
            if (binword != aux):
                prefix = ( (aux.startswith(binword)) or (binword.startswith(aux)) )
                if (prefix):
                    return('No')
    return('Yes')

code = ['00', '11', '001', '111', '00111', '01111', '10000']
print(prefix_code(code))
code = ['000', '001', '010', '0110', '01110', '01111', '10000']
print(prefix_code(code))
code = ['00','01','10','11','00011']
print(prefix_code(code))
code = ['00', '010', '011', '1000', '10010', '10011', '10100']
print(prefix_code(code))
code = ['00', '01', '10', '110', '111','10101']
print(prefix_code(code))
code = ['00', '001']
print(prefix_code(code))
'''
No
Yes
No
Yes
No
No
'''
```

Question 3

Correct

Mark 1.00 out of 1.00

Flag question

The **Shannon source code theorem** gives

Select one:

- (•) Lower and upper bounds for the mean length of an optimal code ✓ Yes, this is exactly what it does
- ( ) A criterion to decide whether a set of binary words is a code or not
- ( ) A method to build optimal codes
- ( ) A formula for the entropy of a source

Your answer is correct.

The correct answer is: Lower and upper bounds for the mean length of an optimal code

Question 4

Correct

Mark 1.00 out of 1.00

Flag question

Which of the following sources has the largest entropy?

Select one:

- ( ) A source of 8 letters with different probabilities
- ( ) The 3-extension of a binary source with different probabilities
- ( ) A source of 8 letters with equal probabilities
- (•) The 4-extension of a binary source with equal probabilities ✓ Correct. The entropy is 4, the largest possible among all examples
- ( ) The 4-extension of a binary source with different probabilities

Your answer is correct.

The correct answer is: The 4-extension of a binary source with equal probabilities

```python
from math import log2
from itertools import product

def source(str):
    output = []
    for char in set(str):
        output.append( (char, str.count(char) / len(str)) )
    return output

def source_extension(src,k):
    letters = []
    output = []
    for tupla in src:
        letters.append(tupla[0])
    permutations = [''.join(i) for i in product(letters, repeat = k)]
    for perm in permutations:
        suma = 1
        for let in letters:
            problet = [tupla[1] for tupla in src if tupla[0] == let]
            suma *= problet[0]**(perm.count(let))
        output.append((perm, suma))
    return output

def entropy_source(src):
    hx = 0.0
    for i in range(len(src)):
        pxi = src[i][1]
        hx += pxi*(log2(1.0/pxi))
    return hx

s = [('a',0.03), ('b',0.05), ('c',0.08), ('d',0.09),
    ('e',0.1), ('f',0.15), ('g',0.2), ('h',0.3)]
print('A source of 8 letters with different probabilities',entropy_source(s))

string = '00000100000000000100'
s = source(string)
source_ext = source_extension(s,3)
print('The 3-extension of a binary source with different probabilities',entropy_source(source_ext))
```

```
s = [('a',0.125), ('b',0.125), ('c',0.125), ('d',0.125),
    ('e',0.125), ('f',0.125), ('g',0.125), ('h',0.125)]
print('A source of 8 letters with equal probabilities',entropy_source(s))

string = '0000011111'
s = source(string)
source_ext = source_extension(s,4)
print('The 4-extension of a binary source with equal probabilities',entropy_source(source_ext))

string = '00000100000000000100'
s = source(string)
source_ext = source_extension(s,4)
print('The 4-extension of a binary source with different probabilities',entropy_source(source_ext))

'''
A source of 8 letters with different probabilities 2.700238463388872
The 3-extension of a binary source with different probabilities 1.406986780767844
A source of 8 letters with equal probabilities 3.0
The 4-extension of a binary source with equal probabilities 4.0
The 4-extension of a binary source with different probabilities 1.875982374357125
'''
```

Question 5

Partially correct

Mark 0.33 out of 1.00

Flag question

Looking at a long string of binary digits we see that

- The probability that a digit is a one is $\frac{1}{4}$
- The probability that two consecutive digits are both ones is $\frac{1}{10}$
- The probability that after a zero comes another zero is $\frac{4}{5}$

Which of the following are true?

Select one or more:

- [ ] The probability that a digit is a zero is $\frac{15}{16}$
- [x] The probability that after a one comes a one is $\frac{2}{5}$ ✓
- [ ] The probability of two consecutive zeros is $\frac{13}{15}$
- [ ] The probability of the pair of consecutive symbols zero-one is $\frac{3}{20}$
- [ ] The probability that after a zero comes a one is $\frac{1}{5}$
- [ ] The probability that after a one comes a zero is $\frac{3}{10}$

Your answer is partially correct.

If X,Y are the random variables corresponding to pick randomly two consecutive digits,

- P(X=1) = $\frac{1}{4}$
- P(X=1,Y=1) = p2
- P(Y=0|X=0) = p3

With the formulas and properties of probability theory for pairs of random variables, one can compute:

- P(X=0) = 1 - P(X=1)
- P(X=0,Y=0) = P(Y=0|X=0)*P(X=0)
- P(Y=1|X=0) = 1 - P(Y=0|X=0)
- P(X=0,Y=1) = P(Y=1|X=0)*P(X=0)
- P(X=1,Y=0) = 1 - P(X=0,Y=0) - P(X=1,Y=1) - P(X=0,Y=1)

and answer all the questions

The correct answer is: The probability that after a zero comes a one is $\frac{1}{5}$, The probability that after a one comes a one is $\frac{2}{5}$, The probability of the pair of consecutive symbols zero-one is $\frac{3}{20}$

```
from fractions import Fraction

probX1 = Fraction(1,4)
probX1Y1 = Fraction(1,10)
probY0_X0 = Fraction(4,5)

probX0 = 1 - probX1
probX0Y0 = probY0_X0 * probX0
probY1_X0 = 1 - probY0_X0
probX0Y1 = probY1_X0 * probX0
probX1Y0 = 1 - probX0Y0 - probX1Y1 - probX0Y1
probY0_X1 = probX1Y0 / probX1
probY1_X1 = 1 - probY0_X1

print('The probability that a digit is a zero is',probX0)
print('The probability that after a one comes a one is',probY1_X1)
print('The probability of two consecutive zeros is',probX0Y0)
print('The probability of the pair of consecutive symbols zero-one is',probX0Y1)
print('The probability that after a zero comes a one is',probY1_X0)
print('The probability that after a one comes a zero is',probY0_X1)

'''
The probability that a digit is a zero is 3/4
The probability that after a one comes a one is 2/5
The probability of two consecutive zeros is 3/5
The probability of the pair of consecutive symbols zero-one is 3/20
The probability that after a zero comes a one is 1/5
The probability that after a one comes a zero is 3/5
'''
```

Question 6

Not answered

Marked out of 1.00

Flag question

What knowledge about the outcome of the experiment of throwing two dice contains more information?

Select one:

- The product is (strictly) larger than 10
- The value of the sum
- The sum is (strictly) larger than 10
- The sum is equal to 7
- The outputs have different parity

Your answer is incorrect.

The probability that the sum is larger than 10 is 3/36 (favorable cases: (5,6), (6,5) and (6,6)), and the cooresponding information is log2(12)=3.58496...

The correct answer is: The sum is (strictly) larger than 10

Question 7

Correct

Mark 1.00 out of 1.00

Flag question

Who is the father of **Information Theory** ?

Select one:

- Claude Shannon ✓
- David Huffman
- Alan Turing
- Peter Arithmetic

Your answer is correct.

Information Theory was created in 1947 by engineer and mathematician Claude Shannon

The correct answer is: Claude Shannon

Question 8

Not answered

Marked out of 1.00

Flag question

A sensor in a factory takes measures at regular intervals of time, represented by integers n in the range from 0 to N-1. Due to the properties of the system for every n=0,1,2,...,N-3,N-2 the probability of measuring the value n is twice the probability of measuring the next value n+1.

The chief engineer of the factory proposes the three following ways to binary encode the information provided by the sensor:

- (A) Use a block code with words of length k with k the smallest integer such that $N \leq 2^k$;
- (B) Encode every integer n with the binary word consisting of n ones followed by a zero;
- (C) Encode every integer n with the binary word consisting of n ones followed by a zero, except the largest integer n=N-1, which is encoded just as the word consisting of n ones without a zero at the end.

Answer for true or false:

The code in (A) is a Huffman code for the source described	Choose...
(A) and (B) are prefix sets, but (C) is not prefix, and consequently it is not a code	Choose...
The code (C) is a Huffman code for the source described	Choose...
For N=4 the entropy of the source is log(15)-34/15	Choose...
For N=8 the probability of n=7 is 1/255	Choose...
When N tends to infinity the probability of n=1 tends to 1/4	Choose...
For N=8 the probability of n=7 is 1/7	Choose...
The code (B) is a Huffman code for the source described	Choose...
For N=4 the entropy of the source is log(15)+16/15	Choose...

Your answer is incorrect.

The correct answer is: The code in (A) is a Huffman code for the source described – False, (A) and (B) are prefix sets, but (C) is not prefix, and consequently it is not a code – False, The code (C) is a Huffman code for the source described – True, For N=4 the entropy of the source is log(15)-34/15 – True, For N=8 the probability of n=7 is 1/255 – True, When N tends to infinity the probability of n=1 tends to 1/4 – True, For N=8 the probability of n=7 is 1/7 – False, The code (B) is a Huffman code for the source described – False, For N=4 the entropy of the source is log(15)+16/15 – False

Question 9

Correct

Mark 1.00 out of 1.00

Flag question

An arithmetic code working with integers of 11 bits is used to encode information produced by a source with an alphabet of three letters and probabilities

$$\frac{11}{50436}, \frac{16765}{16812}, \frac{65}{25218}$$

Is the precision 11 enough for working with that source?

Select one:

- No ✓
- Yes

Your answer is correct.

The smallest good precision is 15, the base 2 logarithm of the inverse of the smallest probability (integer part by excess) plus 2

The correct answer is: No

```python
from math import log2, ceil
k = 11
probs = [11/50436, 16765/16812, 65/25218]
print('The smallest good precision is',ceil(log2(1/min(probs))+2))
if k >= ceil(log2(1/min(probs))+2):
    print("Yes")
else:
    print("No")

'''
The smallest good precision is 15
No
'''
```

Question 10

Correct

Mark 1.00 out of 1.00

Flag question

For which probability distribution the following binary code is a Huffman code ? (codewords and probabilities are given in the same order)

['1','0111','0110','01011','01010','01000','01001','0011','0010','0000','0001']

Select one:

- [0.9,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01] ✓
- [0.07,0.08,0.08,0.08,0.08,0.09,0.1,0.1,0.1,0.1,0.11,0.11]
- [1/11,1/11,1/11,1/11,1/11,1/11,1/11,1/11,1/11,1/11,1/11]

Your answer is correct.

The correct answer is: [0.9,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01]

```python
import queue

class HuffmanNode(object):
    def __init__(self, left, right, root):
        self.left = left
        self.right = right
        self.root = root

def fromRootToLeafs(node, binarywords, aux):
    if (node.left == None and node.right == None):
         binarywords.append(aux)
    else:
        if (node.right != None):
            aux += '0'
            fromRootToLeafs(node.right,binarywords,aux)
            aux = aux[:len(aux)-1]
        if (node.left != None):
            aux += '1'
            fromRootToLeafs(node.left,binarywords,aux)

def huffman_code(src):
    id_desempate = 0
    q = queue.PriorityQueue()
    q2 = queue.PriorityQueue()
    for tupla in src:
        leafNode = HuffmanNode(None, None, tupla[1])
        q.put( (tupla[1], id_desempate, leafNode) )
        q2.put( (tupla[1], id_desempate, leafNode) )
        id_desempate += 1
    while q.qsize() > 1:
        leftNode, rightNode = q.get()[2], q.get()[2]
        parentNode = HuffmanNode(leftNode, rightNode, leftNode.root+rightNode.root)
        q.put((parentNode.root, id_desempate, parentNode))
        id_desempate += 1
    huffmanTree = q.get()[2]
    binarywords = []
    fromRootToLeafs(huffmanTree,binarywords,'')
    binarywords.sort(key = len)
    mean_length = 0.0
    for binword in reversed(binarywords):
        mean_length += q2.get()[0] * len(binword)
    return binarywords, mean_length
```

```
bincode = ['1','0111','0110','01011','01010','01000','01001','0011','0010','0000','0001']
for i in bincode:
    print(len(i), end = " ")
print('')

s1 = [('a',0.9), ('b',0.01), ('c',0.01), ('d',0.01),
    ('e',0.01), ('f',0.01), ('g',0.01), ('h',0.01),
    ('i',0.01), ('j',0.01), ('k',0.01)]
huff, mean = huffman_code(s1)
print('Huffman code for the source s1')
print(huff,',',mean)
for i in huff:
    print(len(i), end = " ")
print('')

s2 = [('a',0.07), ('b',0.08), ('c',0.08), ('d',0.08),
    ('e',0.08), ('f',0.09), ('g',0.1), ('h',0.1),
    ('i',0.1), ('j',0.11), ('k',0.11)]
huff, mean = huffman_code(s2)
print('Huffman code for the source s2')
print(huff,',',mean)
for i in huff:
    print(len(i), end = " ")
print('')

s3 = [('a',1/11), ('b',1/11), ('c',1/11), ('d',1/11),
    ('e',1/11), ('f',1/11), ('g',1/11), ('h',1/11),
    ('i',1/11), ('j',1/11), ('k',1/11)]
huff, mean = huffman_code(s3)
print('Huffman code for the source s3')
print(huff,',',mean)
for i in huff:
    print(len(i), end = " ")
print('')

'''
1 4 4 5 5 5 5 4 4 4 4
Huffman code for the source s1
['0', '1010', '1011', '1100', '1101', '1110', '1111', '10000', '10001', '10010', '10011'] ,
1.3399999999999999
1 4 4 4 4 4 4 5 5 5 5
Huffman code for the source s2
['011', '100', '101', '110', '111', '0000', '0001', '0010', '0011', '0100', '0101'] ,
3.4800000000000004
3 3 3 3 3 4 4 4 4 4 4
Huffman code for the source s3
['011', '100', '101', '110', '111', '0000', '0001', '0010', '0011', '0100', '0101'] ,
3.545454545454545
3 3 3 3 3 4 4 4 4 4 4
'''
```

Question 11

Correct

Mark 1.00 out of 1.00

Flag question

What is an **optimal source code** ?

Select one:

- A source code with mean length smaller than the entropy of the source plus one
- An arithmetic code with infinite precision (exact numbers)
- A source code with the smallest mean length among all possible source codes ✓ Yes, this is the definition
- A code with mean length smaller than the entropy of the source

Your answer is correct.

The correct answer is: A source code with the smallest mean length among all possible source codes

Question 12

Correct

Mark 1.00 out of 1.00

Flag question

Consider the source with alphabet {'**a**','**b**','**c**','**d**'} and **uniform probability distribution** (all letters have the same probability).

We encode the text '**cbaa**' using arithmetic coding with infinite precision.

Of the following numbers, which one is an encoding of the given text?

Select one:

- 0.063
- 0.6911
- 0.5636 ✓
- 0.6096

Your answer is correct.

The correct answer is: 0.5636

```
src = [("a",0.25), ("b",0.25), ("c",0.25), ("d",0.25)]
m = 'cbaa'
cumulative_probs = [0]
aux = 0.0
for i in range(len(src)):
    aux += src[i][1]
    cumulative_probs.append(aux)
print('cumulative_probs ->',cumulative_probs)
alpha = 0
beta = 1
for char in m:
    subintervals = []
    for j in range(1, len(cumulative_probs)):
        aux = (alpha + (beta-alpha)*cumulative_probs[j-1],
               alpha + (beta-alpha)*cumulative_probs[j])
        subintervals.append(aux)
    ind = [src.index(tupla) for tupla in src if tupla[0] == char][0]
    print(subintervals)
    print(char, 'ocupa la posicion',ind,'en src')
    alpha = subintervals[ind][0]
    beta = subintervals[ind][1]
    print('Nos quedamos con [',subintervals[ind][0],',',subintervals[ind][1],')')
print('¿Que valor de los siguentes puede representar el mensaje?')
opcionA = 0.063
if (subintervals[ind][0] <= opcionA < subintervals[ind][1]):
    print('opcionA')
opcionB = 0.6911
if (subintervals[ind][0] <= opcionB < subintervals[ind][1]):
    print('opcionB')
opcionC = 0.5636
if (subintervals[ind][0] <= opcionC < subintervals[ind][1]):
    print('opcionC')
opcionD = 0.6096
if (subintervals[ind][0] <= opcionD < subintervals[ind][1]):
    print('opcionD')

'''
cumulative_probs -> [0, 0.25, 0.5, 0.75, 1.0]
[(0, 0.25), (0.25, 0.5), (0.5, 0.75), (0.75, 1.0)]
c ocupa la posicion 2 en src
Nos quedamos con [ 0.5 , 0.75 )
[(0.5, 0.5625), (0.5625, 0.625), (0.625, 0.6875), (0.6875, 0.75)]
b ocupa la posicion 1 en src
Nos quedamos con [ 0.5625 , 0.625 )
[(0.5625, 0.578125), (0.578125, 0.59375), (0.59375, 0.609375), (0.609375, 0.625)]
a ocupa la posicion 0 en src
Nos quedamos con [ 0.5625 , 0.578125 )
[(0.5625, 0.56640625), (0.56640625, 0.5703125), (0.5703125, 0.57421875), (0.57421875, 0.578125)]
a ocupa la posicion 0 en src
Nos quedamos con [ 0.5625 , 0.56640625 )
¿Que valor de los siguentes puede representar el mensaje?
opcionC
'''
```

Pregunta 3

No s'ha respost encara

Puntuat sobre 1,00

Marca la pregunta

Edita la pregunta aritmetica texto -> X

Con el alfabeto ['a','b','c','d'] y probabilidades [0.25,0.25,0.25,0.25] se ha codificado aritméticamente el mensaje bcba. ¿Qué valor de los siguentes puede representar el mensaje?

Trieu-ne una:

- a. 0.1422
- b. 0.9202
- c. 0.3926

```
src = [("a",0.25), ("b",0.25), ("c",0.25), ("d",0.25)]
m = 'bcba'
cumulative_probs = [0]
aux = 0.0
for i in range(len(src)):
    aux += src[i][1]
    cumulative_probs.append(aux)
print('cumulative_probs ->',cumulative_probs)
alpha = 0
beta = 1
for char in m:
    subintervals = []
    for j in range(1, len(cumulative_probs)):
        aux = (alpha + (beta-alpha)*cumulative_probs[j-1],
               alpha + (beta-alpha)*cumulative_probs[j])
        subintervals.append(aux)
    ind = [src.index(tupla) for tupla in src if tupla[0] == char][0]
    print(subintervals)
    print(char, 'ocupa la posicion',ind,'en src')
    alpha = subintervals[ind][0]
    beta = subintervals[ind][1]
    print('Nos quedamos con [',subintervals[ind][0],',',subintervals[ind][1],')')
print('¿Que valor de los siguentes puede representar el mensaje?')
opcionA = 0.1422
if (subintervals[ind][0] <= opcionA < subintervals[ind][1]):
    print('opcionA')
opcionB = 0.9202
if (subintervals[ind][0] <= opcionB < subintervals[ind][1]):
    print('opcionB')
opcionC = 0.3926
if (subintervals[ind][0] <= opcionC < subintervals[ind][1]):
    print('opcionC')

'''
cumulative_probs -> [0, 0.25, 0.5, 0.75, 1.0]
[(0, 0.25), (0.25, 0.5), (0.5, 0.75), (0.75, 1.0)]
b ocupa la posicion 1 en src
Nos quedamos con [ 0.25 , 0.5 )
[(0.25, 0.3125), (0.3125, 0.375), (0.375, 0.4375), (0.4375, 0.5)]
c ocupa la posicion 2 en src
Nos quedamos con [ 0.375 , 0.4375 )
[(0.375, 0.390625), (0.390625, 0.40625), (0.40625, 0.421875), (0.421875, 0.4375)]
b ocupa la posicion 1 en src
Nos quedamos con [ 0.390625 , 0.40625 )
[(0.390625, 0.39453125), (0.39453125, 0.3984375), (0.3984375, 0.40234375), (0.40234375, 0.40625)]
a ocupa la posicion 0 en src
Nos quedamos con [ 0.390625 , 0.39453125 )
¿Que valor de los siguentes puede representar el mensaje?
opcionC
'''
```

Pregunta 4

No s'ha respost encara

Puntuat sobre 1,00

Marca la pregunta

Edita la pregunta huffman y entropía

Una fuente S viene dada per la ddp $\left[\frac{19}{73}, \frac{6}{73}, \frac{17}{73}, \frac{6}{73}, \frac{25}{73}\right]$. Un código de Huffman asociado a S tiene una longitud media $\tilde{l}$. ¿Qué valores puede tomar $\tilde{l}$?

Trieu-ne una:

- a. Entre 3.117 y 3.35
- b. Entre 3.35 y 4.117
- c. Entre 2.117 y 3.117

```python
import queue

class HuffmanNode(object):
    def __init__(self, left, right, root):
        self.left = left
        self.right = right
        self.root = root

def fromRootToLeafs(node, binarywords, aux):
    if (node.left == None and node.right == None):
         binarywords.append(aux)
    else:
        if (node.right != None):
            aux += '0'
            fromRootToLeafs(node.right,binarywords,aux)
            aux = aux[:len(aux)-1]
        if (node.left != None):
            aux += '1'
            fromRootToLeafs(node.left,binarywords,aux)

def huffman_code(src):
    id_desempate = 0
    q = queue.PriorityQueue()
    q2 = queue.PriorityQueue()
    for tupla in src:
        leafNode = HuffmanNode(None, None, tupla[1])
        q.put( (tupla[1], id_desempate, leafNode) )
        q2.put( (tupla[1], id_desempate, leafNode) )
        id_desempate += 1
    while q.qsize() > 1:
        leftNode, rightNode = q.get()[2], q.get()[2]
        parentNode = HuffmanNode(leftNode, rightNode, leftNode.root+rightNode.root)
        q.put((parentNode.root, id_desempate, parentNode))
        id_desempate += 1
    huffmanTree = q.get()[2]
    binarywords = []
    fromRootToLeafs(huffmanTree,binarywords,'')
    binarywords.sort(key = len)
    mean_length = 0.0
    for binword in reversed(binarywords):
        mean_length += q2.get()[0] * len(binword)
    return binarywords, mean_length

src = [('a', 19/73), ('b', 6/73), ('c', 17/73), ('d', 6/73), ('e', 25/73)]
print(huffman_code(src))

'''
(['00', '01', '10', '110', '111'], 2.1643835616438354)
'''
```

Pregunta 5

No s'ha respost encara

Puntuat sobre 1,00

Marca la pregunta

Edita la pregunta aritmetica x->texto

Con el alfabeto ['a','b','c','d'] y probabilidades [0.25,0.25,0.25,0.25] se ha codificado un mensaje de longitud 5 con el valor 0.1094. ¿Cual és el mensaje?

Trieu-ne una:

- a. abcad
- b. abcda
- c. aabcd

```
src = [("a",0.25), ("b",0.25), ("c",0.25), ("d",0.25)]
longitud = 5
valor = 0.1094
cumulative_probs = [0]
aux = 0.0
for i in range(len(src)):
    aux += src[i][1]
    cumulative_probs.append(aux)
print('cumulative_probs ->',cumulative_probs)
alpha = 0
beta = 1
mensaje = ''
for i in range(longitud):
    subintervals = []
    for j in range(1, len(cumulative_probs)):
        aux = (alpha + (beta-alpha)*cumulative_probs[j-1],
               alpha + (beta-alpha)*cumulative_probs[j])
        subintervals.append(aux)
    print(subintervals)
    for ind,subint in enumerate(subintervals):
        if subint[0] <= valor < subint[1]:
            mensaje += src[ind][0]
            alpha = subint[0]
            beta = subint[1]
            print(subint[0],'<=',valor,'<',subint[1])
            print('mensaje =',mensaje)
            print('Nos quedamos con [',alpha,',',beta,')')
            break
print('¿Cual es el mensaje?',mensaje)

'''
cumulative_probs -> [0, 0.25, 0.5, 0.75, 1.0]
[(0, 0.25), (0.25, 0.5), (0.5, 0.75), (0.75, 1.0)]
0 <= 0.1094 < 0.25
mensaje = a
Nos quedamos con [ 0 , 0.25 )
[(0.0, 0.0625), (0.0625, 0.125), (0.125, 0.1875), (0.1875, 0.25)]
0.0625 <= 0.1094 < 0.125
mensaje = ab
Nos quedamos con [ 0.0625 , 0.125 )
[(0.0625, 0.078125), (0.078125, 0.09375), (0.09375, 0.109375), (0.109375, 0.125)]
0.109375 <= 0.1094 < 0.125
mensaje = abd
Nos quedamos con [ 0.109375 , 0.125 )
[(0.109375, 0.11328125), (0.11328125, 0.1171875), (0.1171875, 0.12109375), (0.12109375, 0.125)]
0.109375 <= 0.1094 < 0.11328125
mensaje = abda
Nos quedamos con [ 0.109375 , 0.11328125 )
[(0.109375, 0.1103515625), (0.1103515625, 0.111328125), (0.111328125, 0.1123046875), (0.1123046875, 0.11328125)]
0.109375 <= 0.1094 < 0.1103515625
mensaje = abdaa
Nos quedamos con [ 0.109375 , 0.1103515625 )
¿Cual es el mensaje? abdaa
'''
```

Pregunta 7

No s'ha respost encara

Puntuat sobre 1,00

Marca la pregunta

Edita la pregunta Kraft palabras a añadir

¿Cuántas palabras de longitud máxima se pueden añadir a un código binario con palabras de longitudes [2,3,3,4,5,5,8,10,10,10]?

Trieu-ne una:

- a. 379
- b. 377
- c. 378

```
longitudes = (2,3,3,4,5,5,8,10,10,10)
sumatorio = 0
for li in longitudes:
    sumatorio += 2**(-li)
veces = 0
while sumatorio < 1:
    veces += 1
    sumatorio += 2**(-10)
print(veces)

'''
377
'''
```