# -\*- coding=utf-8 -\*-

# This is the code to deal 19.6.18's homework

# Copyleft: everybody in ECUST grades 3 major in biology could use this code do genetic engineering homework

# Written by yhy, supported by qjz

import re #引入正则库

def reverse(dna):

pairs = {'A':'T','T':'A','C':'G','G':'C'}

return ''.join([pairs[x] for x in dna])[::-1]

def transcribe(dna):

pairs = {'A':'U','T':'A','C':'G','G':'C'}

return ''.join([pairs[x] for x in dna])[::-1]

def find\_start(rna):

index1 = -1

index1\_list=[]

index2 = -1

index2\_list=[]

index3 = -1

index3\_list=[]

coden1 = re.findall(r'.{3}',rna)

coden2 = re.findall(r'.{3}',rna[1::])

coden3 = re.findall(r'.{3}',rna[2::])

for i in coden1:

index1+=1

if i == "GUG" or i == "AUG":

index1\_list.append(index1)

for i in coden2:

index2+=1

if i == "GUG" or i == "AUG":

index2\_list.append(index2)

for i in coden3:

index3+=1

if i == "GUG" or i == "AUG":

index3\_list.append(index3)

coden = []

for i in index1\_list:

coden.append(coden1[i::])

for i in index2\_list:

coden.append(coden2[i::])

for i in index3\_list:

coden.append(coden3[i::])

return(coden)

def find\_end(rna):

index1 = -1

index1\_list=[]

index2 = -1

index2\_list=[]

index3 = -1

index3\_list=[]

coden1 = re.findall(r'.{3}',rna)

coden2 = re.findall(r'.{3}',rna[1::])

coden3 = re.findall(r'.{3}',rna[2::])

for i in coden1:

index1+=1

if i == "UGA" or i == "UAG" or i == "UAA":

index1\_list.append(index1)

for i in coden2:

index2+=1

if i == "UGA" or i == "UAG" or i == "UAA":

index2\_list.append(index2)

for i in coden3:

index3+=1

if i == "UGA" or i == "UAG" or i == "UAA":

index3\_list.append(index3)

coden = []

for i in index1\_list:

coden.append(coden1[:i+1:])

for i in index2\_list:

coden.append(coden2[:i+1:])

for i in index3\_list:

coden.append(coden3[:i+1:])

return(coden)

def CG\_content(conden\_list):

cg\_content = []

for i in conden\_list:

first =(i[1].count('G')+i[1].count('C'))/3

second = (i[2].count('G')+i[2].count('C'))/3

thrid = (i[3].count('G')+i[3].count('C'))/3

cg\_content.append([first\*100, second\*100, thrid\*100])

return cg\_content

def most\_likely(codens, var):

index = 0

min\_var =min(var)

index\_list = []

likely\_coden = []

for i in var:

if i == min\_var:

index\_list.append(index)

index+=1

for i in index\_list:

likely\_coden.append(codens[i])

return likely\_coden

def suit(cg):

variance = []

for i in cg:

variance.append(((int(i[0])-66)\*\*2+(int(i[1])-53)\*\*2+(int(i[2])-93)\*\*2)/3)

return variance

def rare\_coden(coden):

dic = {}

print(coden)

for i in coden:

string = ''

for j in i:

string += j

count\_n = int(i.count("UUA"))+int(i.count("CUA"))

count\_r = int(i.count("UUU"))+int(i.count("AUU"))+int(i.count("GUU"))+int(i.count("UCU"))+int(i.count("CCA"))+int(i.count("ACU"))

dic[string] = str("稀有密码子"+str(count\_r)+",禁止密码子"+str(count\_n))

return(dic)

code\_s = 'CGCTTGCGTGCGTCTCATCTGGAGGCTGCACCTGTCCGACCACTACGGCATCTGGCAGGAATTCGGCCAGAAGTTCCTCGTCGACCCCGACATCATCAAG'

code\_e = 'GGTGCGAGAGCGGATACTCCAGGACCTTGCGCACCTACCTGTCCTACGGCAGCAGCGACAACGCGAACCCAGTCTGCGAGACACGGCTGTGCAGGCCGTC' #输入序列

pair\_code\_s = reverse(code\_e)

pair\_code\_e = reverse(code\_s)

code\_s\_rna = transcribe(code\_s)

code\_e\_rna = transcribe(code\_e)

pair\_code\_s\_rna = transcribe(pair\_code\_s)

pair\_code\_e\_rna = transcribe(pair\_code\_e)

start\_findall = find\_start(code\_s\_rna)

pair\_start\_findall = find\_start(pair\_code\_s\_rna)

end\_findall = find\_end(code\_e\_rna)

pair\_end\_findall = find\_end(pair\_code\_e\_rna)

cg\_start = CG\_content(start\_findall)

pair\_cg\_start = CG\_content(pair\_start\_findall)

start\_suit = suit(cg\_start)

pair\_start\_suit = suit(pair\_cg\_start)

most\_likely\_s = most\_likely(start\_findall, start\_suit)

pair\_likely\_s = most\_likely(pair\_start\_findall, pair\_start\_suit)

rare = rare\_coden(most\_likely\_s)

pair\_rare = rare\_coden(pair\_likely\_s)

分析如下：

给定的起始序列： CGCTTGCGTGCGTCTCATCTGGAGGCTGCACCTGTCCGACCACTACGGCAT

CTGGCAGGAATTCGGCCAGAAGTTCCTCGTCGACCCCGACATCATCAAG

给定的结束序列： GGTGCGAGAGCGGATACTCCAGGACCTTGCGCACCTACCTGTCCTACGGC

AGCAGCGACAACGCGAACCCAGTCTGCGAGACACGGCTGTGCAGGCCGTC

互补起始链为： GACGGCCTGCACAGCCGTGTCTCGCAGACTGGGTTCGCGTTGTCGCTGCTGC

CGTAGGACAGGTAGGTGCGCAAGGTCCTGGAGTATCCGCTCTCGCACC

互补终止链为： CTTGATGATGTCGGGGTCGACGAGGAACTTCTGGCCGAATTCCTGCCAGATG

CCGTAGTGGTCGGACAGGTGCAGCCTCCAGATGAGACGCACGCAAGCG

给的DNA起始转录后为： CUUGAUGAUGUCGGGGUCGACGAGGAACUUCUGGCCGAAUUC

CUGCCAGAUGCCGUAGUGGUCGGACAGGUGCAGCCUCCAGAUGAGACGCACGCAAGCG

给的DNA结束转录后为： GACGGCCUGCACAGCCGUGUCUCGCAGACUGGGUUCGCGUUGU

CGCUGCUGCCGUAGGACAGGUAGGUGCGCAAGGUCCUGGAGUAUCCGCUCUCGCACC

给的DNA互补起始转录后为： GGUGCGAGAGCGGAUACUCCAGGACCUUGCGCACCUACC

UGUCCUACGGCAGCAGCGACAACGCGAACCCAGUCUGCGAGACACGGCUGUGCAGGCCGUC

给的DNA互补结束转录后为： CGCUUGCGUGCGUCUCAUCUGGAGGCUGCACCUGUCCGA

CCACUACGGCAUCUGGCAGGAAUUCGGCCAGAAGUUCCUCGUCGACCCCGACAUCAUCAAG

给的链找起始密码子：

['GUG', 'GUC', 'GGA', 'CAG', 'GUG', 'CAG', 'CCU', 'CCA', 'GAU', 'GAG', 'ACG', 'CAC', 'GCA', 'AGC']

['GUG', 'CAG', 'CCU', 'CCA', 'GAU', 'GAG', 'ACG', 'CAC', 'GCA', 'AGC']

['AUG', 'AUG', 'UCG', 'GGG', 'UCG', 'ACG', 'AGG', 'AAC', 'UUC', 'UGG', 'CCG', 'AAU', 'UCC', 'UGC', 'CAG', 'AUG', 'CCG', 'UAG', 'UGG', 'UCG', 'GAC', 'AGG', 'UGC', 'AGC', 'CUC', 'CAG', 'AUG', 'AGA', 'CGC', 'ACG', 'CAA', 'GCG'] ['AUG', 'UCG', 'GGG', 'UCG', 'ACG', 'AGG', 'AAC', 'UUC', 'UGG', 'CCG', 'AAU', 'UCC', 'UGC', 'CAG', 'AUG', 'CCG', 'UAG', 'UGG', 'UCG', 'GAC', 'AGG', 'UGC', 'AGC', 'CUC', 'CAG', 'AUG', 'AGA', 'CGC', 'ACG', 'CAA', 'GCG']

['AUG', 'CCG', 'UAG', 'UGG', 'UCG', 'GAC', 'AGG', 'UGC', 'AGC', 'CUC', 'CAG', 'AUG', 'AGA', 'CGC', 'ACG', 'CAA', 'GCG']

['AUG', 'AGA', 'CGC', 'ACG', 'CAA', 'GCG']

共6个起始密码子，之后进行密码子分析

互补链找起始密码子： ['GUG', 'CGA', 'GAG', 'CGG', 'AUA', 'CUC', 'CAG', 'GAC', 'CUU', 'GCG', 'CAC', 'CUA', 'CCU', 'GUC', 'CUA', 'CGG', 'CAG', 'CAG', 'CGA', 'CAA', 'CGC', 'GAA', 'CCC', 'AGU', 'CUG', 'CGA', 'GAC', 'ACG', 'GCU', 'GUG', 'CAG', 'GCC', 'GUC']

['GUG', 'CAG', 'GCC', 'GUC']

互补链又两个起始密码子，之后进行密码子分析

给的链找终止密码子：

['GAC', 'GGC', 'CUG', 'CAC', 'AGC', 'CGU', 'GUC', 'UCG', 'CAG', 'ACU', 'GGG', 'UUC', 'GCG', 'UUG', 'UCG', 'CUG', 'CUG', 'CCG', 'UAG']

['GAC', 'GGC', 'CUG', 'CAC', 'AGC', 'CGU', 'GUC', 'UCG', 'CAG', 'ACU', 'GGG', 'UUC', 'GCG', 'UUG', 'UCG', 'CUG', 'CUG', 'CCG', 'UAG', 'GAC', 'AGG', 'UAG']

正链有两个终止密码子

互补链的3‘没有终止密码子

给的链前三个密码子CG比例（单位：%）：

[66.66666666666666, 66.66666666666666, 66.66666666666666]

[66.66666666666666, 66.66666666666666, 66.66666666666666]

[33.33333333333333, 66.66666666666666, 100.0]

[66.66666666666666, 100.0, 66.66666666666666]

[100.0, 33.33333333333333, 66.66666666666666]

[33.33333333333333, 100.0, 66.66666666666666]]

互补链前三个密码子CG比例

[66.66666666666666, 66.66666666666666, 100.0]

[66.66666666666666, 100.0, 66.66666666666666]

给的链前三个密码子CG的方差：

[299.3333333333333, 299.3333333333333, 435.6666666666667, 979.3333333333334, 761.6666666666666, 1342.3333333333333]

互补链前三个密码子CG的方差： [72.66666666666667, 979.3333333333334]

这里的方差，是按照书上给定的CG占比来确定的，可以说，方差越小，越接近真实的密码子结构。

给的链最可能起始子：

['GUG', 'GUC', 'GGA', 'CAG', 'GUG', 'CAG', 'CCU', 'CCA', 'GAU', 'GAG', 'ACG', 'CAC', 'GCA', 'AGC']

['GUG', 'CAG', 'CCU', 'CCA', 'GAU', 'GAG', 'ACG', 'CAC', 'GCA', 'AGC']

由于这两个的方差一模一样，所以都最为最可能的列入，来进行下一步稀有密码子分析。

互补链最可能起始子： ['GUG', 'CGA', 'GAG', 'CGG', 'AUA', 'CUC', 'CAG', 'GAC', 'CUU', 'GCG', 'CAC', 'CUA', 'CCU', 'GUC', 'CUA', 'CGG', 'CAG', 'CAG', 'CGA', 'CAA', 'CGC', 'GAA', 'CCC', 'AGU', 'CUG', 'CGA', 'GAC', 'ACG', 'GCU', 'GUG', 'CAG', 'GCC', 'GUC']

给定链密码子偏好分析：

'GUGGUCGGACAGGUGCAGCCUCCAGAUGAGACGCACGCAAGC': '稀有密码子1,禁止密码子0'

'GUGCAGCCUCCAGAUGAGACGCACGCAAGC': '稀有密码子1,禁止密码子0' 'GUGCGAGAGCGGAUACUCCAGGACCUUGCGCACCUACCUGUCCUACGGCAGCAGCGACAACGCGAACCCAGUCUGCGAGACACGGCUGUGCAGGCCGUC': '稀有密码子0,禁止密码子2'

由这个可见，互补链上的那条虽然很符合启动子模型，但是有两个禁止密码子（简直够提前终止了）所幸这个里面有１０个以上的ａａ符合题目要求，我们就假设这个提前终止