# Finding a Motif in DNA

# http://rosalind.info/problems/subs/

with open('rosalind\_subs.txt','r') as openfile:

s,t = openfile.read().split()

poslist = []

ls = len(s)

lt = len(t)

for i in range(ls):

#print('s[i:i+lt]=',s[i:i+lt])

if s[i:i+lt]==t:

poslist.append(i+1)

resultstr = ''

for i in poslist:

resultstr+=str(i)+' '

print(resultstr)

# RNA Splicing

# http://rosalind.info/problems/splc/

# 1. Import task

temp\_key = ''

inputdatadic = {}

inputdata = []

with open('rosalind\_splc.txt','r') as f:

line = f.readline().strip()

while line:

if line[:1]=='>':

temp\_key = line[1:]

else:

if temp\_key not in inputdatadic.keys():

inputdatadic[temp\_key] = ''

inputdatadic[temp\_key]+=line

line = f.readline().strip()

for k,v in inputdatadic.items():

inputdata.append(inputdatadic[k])

#print(inputdata)

# 2. Import DNA codon table

dna\_list = []

dna\_dict = {}

with open('dna.txt','r') as openfile:

line = openfile.readline().strip()

while line:

dna\_list = line.split()

for i in range(0,7,2):

dna\_dict[dna\_list[i]]=dna\_list[i+1]

line = openfile.readline().strip()

# 3. Now remove introns

spliced = inputdata[0]

for i in range(1,len(inputdata)):

spliced = spliced.replace(inputdata[i],'')

# 4. function for DNA >> protein translation

def translatedna(inputdna):

codons = []

codons = [inputdna[i:i+3] for i in range(0, len(inputdna), 3)]

protein = ''

for i in codons:

if i=='TAA' or i=='TAG' or i=='TGA' or len(i)<3:

break

else:

#print(dna\_dict[i])

protein+=dna\_dict[i]

return(protein)

# 5. Result

print(translatedna(spliced))

# Transcribing DNA into RNA

# http://rosalind.info/problems/rna/

with open('rosalind\_rna.txt','r') as openfile:

line = openfile.readline().strip()

print(line.replace('T','U'))

# Complementing a Strand of DNA

# http://rosalind.info/problems/revc/

### 3

with open('rosalind\_revc.txt','r') as openfile:

line = openfile.readline().strip()

result = ''

linereversed = line[::-1]

for i in linereversed:

if i=='A': result+='T'

elif i=='T': result+='A'

elif i=='C': result+='G'

elif i=='G': result+='C'

print('input: ',line)

print('\*\*\*')

print('reversed: ',linereversed)

print('\*\*\*')

print('result: ',result)

### Alternatively

'''with open('rosalind\_revc.txt','r') as openfile:

line = openfile.readline().strip()

result = line.replace('A','t').replace('T','a').replace('G','c').replace('C','g').upper()[::-1]

print(line)

print('\*\*\*')

print(result)'''

# Locating Restriction Sites

# http://rosalind.info/problems/revp/

# 1. Import task

sequence = ''

with open('rosalind\_revp.txt','r') as f:

line = f.readline().strip()

while line:

if line[:1]!='>':

sequence+=line

line = f.readline().strip()

print(sequence,len(sequence))

# 2. Let's search for reverse palindromes length 4-12

results = {}

# v1

for i in range(4,13,2):

for n in range(len(sequence)-i+1):

left = sequence[n:n+i//2]

right = sequence[(n+i//2):(n+i)]

right\_inversed = right[::-1]

right\_complement = right\_inversed.replace('A','t').replace('T','a').replace('G','c').replace('C','g').upper()

#print(i,n,left,right,'(',right\_complement,')')

if left==right\_complement:

results[n+1]=i

# v2 - can't understand why it doesn't work

'''

for i in range(4,13,2):

for n in range(len(sequence)-i+1):

direct = sequence[n:n+i]

rev = direct[::-1]

rev\_compl = rev.replace('A','t').replace('T','a').replace('G','c').replace('C','g').upper()

print(n+1,direct,rev,rev\_compl)

if direct==rev\_compl:

print('!')

results[n+1]=i

'''

for k,v in results.items():

print(k,v)

# Translating RNA into Protein

# http://rosalind.info/problems/prot/

rna\_list = []

rna\_dict = {}

input\_rna = ''

# import RNA codon table

with open('rna.txt','r') as openfile:

line = openfile.readline().strip()

while line:

rna\_list = line.split()

for i in range(0,7,2):

rna\_dict[rna\_list[i]]=rna\_list[i+1]

line = openfile.readline().strip()

# import RNA

with open('rosalind\_prot.txt','r') as openfile:

line = openfile.readline().strip()

while line:

input\_rna+=line

line = openfile.readline().strip()

# split RNA into codons

codons = []

codons = [input\_rna[i:i+3] for i in range(0, len(input\_rna), 3)]

# translate RNA into protein

aaresult = ''

for i in codons:

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

break

else:

aaresult+=rna\_dict.get(i)

# print result

print(aaresult)

# Enumerating Gene Orders

# http://rosalind.info/problems/perm/

import random

# 1. Import task

with open('rosalind\_perm.txt','r') as f:

n = int(f.readline().strip())

print('n=',n)

# 2. Let's calculate the number of possible permutations

perm\_qnt = 1

for i in range(n):

perm\_qnt\*=(n-i)

# 3. Le'ts generate possible variants of permutations

def randcomb(n):

variants = [0]\*n

for i in range(n):

while variants[i]==0:

y = random.randint(1,n)

if y not in variants:

variants[i]=y

#print(variants)

return(variants)

all\_perm = []

while len(all\_perm)!=perm\_qnt:

temp = randcomb(n)

if temp not in all\_perm:

all\_perm.append(temp)

print(len(all\_perm))

for i in all\_perm:

line = ''

for x in i:

line+=str(x)+' '

print(line)

# Open Reading Frames

# http://rosalind.info/problems/orf/

dna\_list = []

dna\_dict = {}

temp\_key = ''

inputdata = ''

# 1. Import DNA codon table (http://rosalind.info/glossary/dna-codon-table/)

# ATG - DNA start-codon (AUG for RNA)

with open('dna.txt','r') as openfile:

line = openfile.readline().strip()

while line:

dna\_list = line.split()

for i in range(0,7,2):

dna\_dict[dna\_list[i]]=dna\_list[i+1]

line = openfile.readline().strip()

# 2. Import DNA sequence

with open('rosalind\_orf.txt','r') as f:

line = f.readline().strip()

while line:

if line[:1]!='>':

inputdata+=line

line = f.readline().strip()

print(inputdata,'\n')

# 3. Complementary DNA string

inversed = inputdata.replace('A','t').replace('T','a').replace('G','c').replace('C','g').upper()[::-1]

#print(inversed)

# 4. Search for open reading frames (ORF)

def look4ORF(dnasequence):

proteinlist = []

flagup = False

protein = ''

codon = ''

additionalATG = []

for i in range(0,len(dnasequence),3):

codon = dnasequence[i:i+3]

if codon=='ATG':

protein+='M'

if flagup==False:

flagup=True

else:

# for the case when 'inside' of the bigger protein sequence

# there's another, smaller protein - let's store positions

# of such 'M' residues in 'protein' and then if ORF is

# confirmed (stop-codon found) then we flush the original

# protein to 'proteinlist' and also those 'inside' ones

additionalATG.append(len(protein))

elif codon=='TAA' or codon=='TAG' or codon=='TGA' and flagup==True:

flagup=False

if protein!='' and protein not in proteinlist:

#print(protein)

proteinlist.append(protein)

if additionalATG!=[]:

for x in additionalATG:

if protein[x-1:] not in proteinlist:

#print(protein[x-1:])

proteinlist.append(protein[x-1:])

protein=''

additionalATG = []

else:

if flagup==True and len(codon)==3:

protein+=dna\_dict[codon]

#print(proteinlist)

return proteinlist

# 5. Final search for proteins

finalresults = []

for n in range(3):

finalresults.append(look4ORF(inputdata[n:]))

finalresults.append(look4ORF(inversed[n:]))

#print(finalresults)

finalfinal = []

for sublist in finalresults:

if sublist!='':

if type(sublist)==list:

for i in sublist:

finalfinal.append(i)

finalfinal=set(finalfinal)

for i in finalfinal:

print(i)

# Inferring mRNA from Protein

# http://rosalind.info/problems/mrna/

rna\_list = []

rna\_dict = {}

protseq = ''

codonvariants = {}

rnavariants = 1

# 1. Import RNA codon table (http://rosalind.info/glossary/rna-codon-table/)

with open('rna.txt','r') as openfile:

line = openfile.readline().strip()

while line:

rna\_list = line.split()

for i in range(0,7,2):

rna\_dict[rna\_list[i]]=rna\_list[i+1]

line = openfile.readline().strip()

# 2. Import protein sequence (task)

with open('rosalind\_mrna.txt','r') as openfile:

line = openfile.readline().strip()

while line:

protseq+=line

line = openfile.readline().strip()

# 3. Create a dictionary - how many codons for each aminoacid (aa) (+ stop-codons)

for key,value in rna\_dict.items():

if value in codonvariants:

codonvariants[value]+=1

else:

codonvariants[value]=1

# 4. Count for number of rna variants

for i in range(len(protseq)):

rnavariants\*=codonvariants[protseq[i]]

rnavariants\*=codonvariants['Stop']

result = rnavariants % 1000000

print(result)

# Finding a Protein Motif

# http://rosalind.info/problems/mprt/

import requests

from bs4 import BeautifulSoup

# 1. Import task data

with open('rosalind\_mprt.txt','r') as f:

fasta\_ids = [x.strip() for x in f.readlines()]

# 2. Make urls for further data gathering

uniprot\_urls = []

for i in range(len(fasta\_ids)):

url = 'http://www.uniprot.org/uniprot/'+fasta\_ids[i]+'.fasta'

uniprot\_urls.append(url)

# 3. Function that takes an url, gets fasta-format protein sequence

# from uniprot.org, parses it for protein sequence, searches

# N-glycosylation motif in protein sequens (N{P}[ST]{P}, where {P} -

# any aminoacid except P, [ST] S or T. If no such motif, returns False,

# else returns protein ID + positions of motif

def search4Nglycomotif(url):

r = requests.get(url)

r\_html = r.text

soup = BeautifulSoup(r\_html,'html.parser')

# get all info from Uniprot page

uniprot\_text = soup.get\_text()

# take only protein sequence

temp\_list = uniprot\_text.split('\n')

ps = ''

for i in range(1,len(temp\_list)):

ps+=temp\_list[i]

# look for 4-character pieces N-not\_P-S\_or\_T-not\_P

positions = []

for i in range(1,len(ps)-3):

if ps[i]=='N' and ps[i+1]!='P' and (ps[i+2]=='S' or ps[i+2]=='T') and ps[i+3]!='P':

positions.append(i+1)

if len(positions)==0:

return False

else:

return(positions)

# 4. Iterate througn our proteins ID list (uniprot\_urls) and

# get the final results using function search4Nglycomotif

for i in range(len(uniprot\_urls)):

positions = []

positions = search4Nglycomotif(uniprot\_urls[i])

if positions:

print(fasta\_ids[i])

output = ''

for x in range(len(positions)):

output+=str(positions[x])+' '

print(output.strip())

# Enumerating k-mers Lexicographically

# http://rosalind.info/problems/lexf/

import itertools

# 1. Import task

with open('rosalind\_lexf.txt','r') as f:

voc = f.readline().strip().split()

n = int(f.readline().strip())

print(voc, n)

# 2. Generate all possible n-length combinations of characters from voc

var = list(itertools.product(voc,repeat=n))

variants = []

for x in range(len(var)):

temp = ''

for y in range(len(var[x])):

temp+=var[x][y]

variants.append(temp)

variants=sorted(variants)

for i in variants:

print(i)

# Finding a Shared Motif

# http://rosalind.info/problems/lcsm/

# 1. Import data

temp\_key = ''

inputdata = {}

with open('rosalind\_lcsm.txt','r') as f:

line = f.readline().strip()

while line:

if line[:1]=='>':

temp\_key = line[1:]

else:

if temp\_key not in inputdata.keys():

inputdata[temp\_key] = ''

inputdata[temp\_key]+=line

line = f.readline().strip()

#2. Let's find key-value pair with the shortest value

shotest\_key = ''

shortest\_value = '0'\*1001

for key, value in inputdata.items():

if len(value)<len(shortest\_value):

shortest\_value = value

shotest\_key = key

#print('shortest\_value=',shortest\_value)

#3. Now let's look for similar sequences starting from 2 signs

# and up to shortest\_value

longest\_common = ''

for i in range(2,len(shortest\_value)): # for indenting strings length (min - 2)

for x in range(len(shortest\_value)+1-i): # for shifting the search "window"

#print('i=',i,'x=',x,'search for',shortest\_value[x:(i+x)])

#if all(v for v in inputdata.values() if shortest\_value[x:(i+x)] in v):

if all(shortest\_value[x:(i+x)] in v for v in inputdata.values()):

longest\_common = shortest\_value[x:(i+x)]

#for v in inputdata.values():

# print(v)

print('\n',longest\_common)

# take the smallest string and check if it's in other

# if all strings are of same lenght = continue

# take 2 first signs [0:2] of string0 and search for them in other strings

# if they are in all other strings, store and go to 3 signs

# if no take the next 2 signs (2:4) etc

# if no, take 2 signs with 1 shift (1:3) etc

# if we have 2-sign common, go for 3 first signs

# search for +1-signs line until there's no - then take the previous found

# Mendel's First Law

# http://rosalind.info/problems/iprb/

# k - AA

# m - Aa

# n - aa

# AA or Aa

with open('rosalind\_iprb.txt','r') as openfile:

line = openfile.readline().strip()

mylist = line.split()

print(line)

print(mylist)

k = int(mylist[0])

m = int(mylist[1])

n = int(mylist[2])

print(k, m, n)

sum = k+m+n

#m\*m\*0.75+m\*n\*0.5

#aa = 0

mm0\_25 = m/sum \* (m-1)/(sum-1) \* 1/4

mn1\_2 = m/sum \* n/(sum-1) \* 1#/2

nn1 = n/sum \* (n-1)/(sum-1) \* 1

print('mm0\_25=',mm0\_25)

print('mn1\_2=',mn1\_2)

print('nn1=',nn1)

aa = mm0\_25 + mn1\_2 + nn1

print('aa=',aa)

Ax = 1-aa

print('Ax=',Ax)

# Calculating Expected Offspring

# http://rosalind.info/problems/iev/

'''

AA-AA - r1 - 100% A\_

AA-Aa - r2 - 100% A\_

AA-aa - r3 - 100% A\_

Aa-Aa - r4 - 75% A\_, 25% aa

Aa-aa - r5 - 50% A\_, 50% aa

aa-aa - r6 - 0% A\_, 100% aa

Let's calculate the probability of getting aa offsprings and

totalN-double.recessive

'''

# 1. Import input

with open('rosalind\_iev.txt','r') as f:

r1, r2, r3, r4, r5, r6 = [int(i) for i in f.read().split()]

# 2. Calculate how much aa offsprings we'll get (r4 gives 25%, r5 - 50%, r6 - 100%,

# each pair gives 2 offsprings)

aa = r4\*0.25\*2 + r5\*0.5\*2 + r6\*2

# 3. Total number of offsprings

total = (r1 + r2 + r3 + r4 + r5 + r6)\*2

# 4. A\_ offsprings

result = total - aa

print(result)

# Counting Point Mutations

# http://rosalind.info/problems/hamm/

with open('rosalind\_hamm.txt','r') as openfile:

s = openfile.readline()

t = openfile.readline()

hammdist = 0

print(s)

print(t)

for i in range(len(s)):

if s[i]!=t[i]:

hammdist+=1

print(hammdist)

# Overlap Graphs

# http://rosalind.info/problems/grph/

# variables

tempstr = '' # import and parcing of input data

templist = [] # import and parcing of input data

fastakeys = [] # a list of names for strings in Fasta format (like 'Rosalind\_0498')

fastastrings = [] # a list for strings (like 'GGGTGGG')

pair = '' # auxiliary variable

resultlist = [] # final results (pairs of string keys in Fasta format)

# import to string

with open('rosalind\_grph.txt','r') as openfile:

line = openfile.readline().strip()

while line:

tempstr+=line

line = openfile.readline().strip()

# string to list

templist = tempstr.split('>')

# list to dictionary

for i in range(1,len(templist)):

fastakeys.append(templist[i][:13])

fastastrings.append(templist[i][13:])

cycles = len(fastakeys)

for i in range(cycles):

for x in range(cycles):

if fastastrings[i]!=fastastrings[x] and fastastrings[i][-3:]==fastastrings[x][:3]:

pair = fastakeys[i]+' '+fastakeys[x]

if pair not in resultlist:

resultlist.append(pair)

for value in resultlist:

print(value)

# Computing GC Content

# http://rosalind.info/problems/gc/

def cdpercent(mystring):

strlen = len(mystring)

cg = mystring.count('C')+mystring.count('G')

percent = round(cg\*100/strlen,6)

return percent

totalstr = ''

with open('rosalind\_gc.txt','r') as openfile:

line = openfile.readline()

while line:

totalstr+=line.strip()

line = openfile.readline()

inputlist = []

inputlist = totalstr.split('>')

inputdic = {}

for i in range(1,len(inputlist)):

inputdic[inputlist[i][0:13]]=inputlist[i][13:]

resultdic = {}

for key,value in inputdic.items():

resultdic[key]=cdpercent(inputdic[key])

v=list(resultdic.values())

k=list(resultdic.keys())

print(k[v.index(max(v))])

print(max(v))

#Mortal Fibonacci Rabbits

# http://rosalind.info/problems/fibd/

n = 0 # duration of experiment, months

m = 0 # rabbit's live duration, months

inputlist = []

with open('file.txt','r') as open\_file:

inputlist = open\_file.readline().strip().split(' ')

n = int(inputlist[0])

m = int(inputlist[1])

rabbits = [0]\*m

temp\_rabbits = [0]\*m

for i in range(n):

if i==0:

rabbits[0]=1

else:

temp\_rabbits = [0]\*m

temp\_rabbits[0]=sum(rabbits[1:])

for x in range(1,m):

temp\_rabbits[x]=rabbits[x-1]

rabbits=temp\_rabbits

print(sum(rabbits))

# Rabbits and Recurrence Relations

# http://rosalind.info/problems/fib/

n = 0

k = 0

inputlist = []

with open('rosalind\_fib.txt','r') as open\_file:

line = open\_file.readline().strip()

inputlist = line.split(' ')

n = int(inputlist[0])

k = int(inputlist[1])

print(inputlist)

print('n=',n)

print('k=',k)

series = [[0,1,0],[1,0,0]] # mature/young/total, 1st and 2nd months

mature = 0

young = 0

total = 0

for x in range(2,n):

print('\nMonth',x+1)

mature = series[x-1][0] + series[x-2][0]\*k

print('mature=',mature)

young = series[x-1][0]\*k

print('young=',young)

total = mature + young

print('total=',total)

series.append([mature,young,total])

print(series)

print(series[-1][-1])

# Counting DNA Nucleotides

# http://rosalind.info/problems/dna/

with open('rosalind\_dna.txt','r') as open\_file:

line = open\_file.readline()

print(line)

A, C, G, T = 0, 0, 0, 0

for i in line:

if i=='A': A+=1

elif i=='C': C+=1

elif i=='G': G+=1

elif i=='T': T+=1

print(A,C,G,T)

# Consensus and Profile

# http://rosalind.info/problems/cons/

strdict = {}

bigstring = ''

biglist = []

with open('rosalind\_cons.txt','r') as f:

line = f.readline()

while line:

bigstring+=line.strip()

line = f.readline()

biglist = bigstring.split('>')

#print(biglist)

n = 13

strdict = {biglist[i][0:n]: biglist[i][n:] for i in range(1, len(biglist))}

#print(strdict)

strlen = len(strdict[next(iter(strdict))])

countA = [0]\*strlen

countC = [0]\*strlen

countG = [0]\*strlen

countT = [0]\*strlen

for i in range(strlen):

for value in strdict.values():

#print('i=',i,'; value=',value,'; value[i]=',value[i])

if value[i]=='A':

countA[i]+=1

elif value[i]=='C':

countC[i]+=1

elif value[i]=='G':

countG[i]+=1

elif value[i]=='T':

countT[i]+=1

consensus = ''

maxval = 0

for i in range(strlen):

maxval = max(countA[i],countC[i],countG[i],countT[i])

if countA[i]==maxval:

consensus+='A'

elif countC[i]==maxval:

consensus+='C'

elif countG[i]==maxval:

consensus+='G'

elif countT[i]==maxval:

consensus+='T'

print(consensus)

strA = ''

strC = ''

strG = ''

strT = ''

for i in range(strlen):

strA+=str(countA[i])+' '

strC+=str(countC[i])+' '

strG+=str(countG[i])+' '

strT+=str(countT[i])+' '

print('A:', strA)

print('C:', strC)

print('G:', strG)

print('T:', strT)