In this problem, you will learn a simple Markov model (a hidden Markov model structure) for predicting protein secondary structure. Download the data set associated with the homework, and unzip it (it will create a large number of text files); each of these is a labelled secondary structure and a protein sequence. (These data come from Astral, `http://astral.berkeley.edu/`; see that page for more information.)

I suggest that you use Python+NumPy to manipulate the data and resulting matrices, although you may use any software you like. Please include your code with your submission.

First, load the data and convert it from text strings into numeric indices (to make things easier):

```
import numpy as np
from os import walk
mypath = 'proteins/'  # use path to data files
_, _, filenames = next(walk(mypath), (None, None, []))

mSeq = len(filenames)        # read in each sequence
```

```
o,x = [],[]
for i in range(mSeq):
    f = open( filenames[i] , 'r')
    o.append( f.readline()[:-1] )  # strip trailing '\n'
    x.append( f.readline()[:-1] )
    f.close()

xvals, ovals = set(),set()  # extract the symbols used in x and o
for i in range(mSeq):
    xvals |= set(x[i])
    ovals |= set(o[i])
xvals = list( np.sort( list(xvals) ) )
ovals = list( np.sort( list(ovals) ) )
dx,do = len(xvals),len(ovals)

for i in range(mSeq):       # and convert to numeric indices
    x[i] = np.array([xvals.index(s) for s in x[i]])
    o[i] = np.array([ovals.index(s) for s in o[i]])
```

Each $x^{(i)} =$ `x[i]` is one observed sequence, each with different lengths.

(a) From your data, estimate $p(x_0)$, the initial state distribution for a given sequence.

(b) Now, estimate the transition probability matrix, $T_{ij} = \Pr[x_t = j | x_{t-1} = i]$. What is the stationary distribution of this model? (Print T_{ij} for i, j in the first five state values.)

Note that in this representation, if we represent $p(X_0)$ as a row-vector v, then

$$p(X_1 = j) = \sum_i \Pr[X_t = j | X_{t-1} = i] p(X_0 = i) = v \cdot T.$$

(c) Estimate the emission probability matrix, $O_{ik} = \Pr[o_t = k | x_t = i]$. (Again, print O_{ik} for i, k in the first five values.)

(d) Notice that our model is not a fully generative model: it does not explain the sequence length (it takes it as fixed and known). How might we modify the model (including training, etc.) to also explain sequence length? (You do not need to actually do this; just explain.)

(e) Complete the code to compute the marginal probability of the state x_t given the observation sequence $O = [o_1, \ldots, o_L]$, for each t:

```
function markovMarginals(x,o,p0,Tr,Ob):
  '''Compute p(o) and the marginal probabilities p(x_t|o) for a Markov model
     defined by P[xt=j|xt-1=i] = Tr(i,j) and P[ot=k|xt=i] = Ob(i,k) as numpy matrices'''
  dx,do = Ob.shape()   # if a numpy matrix
  L = len(o)
  f = np.zeros((L,dx))
  r = np.zeros((L,dx))
  p = np.zeros((L,dx))

  f[0,:] = ...   # compute initial forward message
  log_p0 = ...   # update probability of sequence so far
  f[0,:] /= f[0,:].sum()  # normalize (to match definition of f)
```

```
  for t in range(1,L):    # compute forward messages
    f[t,:] = ...
    log_p0 += ...
    f[t,:] /= f[t,:].sum()

  r[L,:] = 1.0  # initialize reverse messages
  p[L,:] = ...  # and marginals

  for t in range(L-1,-1,-1):
    r[t,:] = ...
    r[t,:] /= r[t,:].sum()
    p[t,:] = ...
    p[t,:] /= p[t,:].sum()

  return log_p0, p
```