I519 Homework 5, Fall, 2015

Due Dec 4th (Friday), 2015 11:59pm (total 100 pt)

November 15, 2015

1 Working with protein structures

This assignment is to help you get some basic idea about how to work with a protein structure (as in PDB format). And parsing PDB files will be an excellent exercise for those who still need to sharpen their programming skills. In a PDB file, each line that begins with "ATOM" records the cartesian coordinate of an atom.

Contact order (CO), which reflects the relative importance of local and non-local contacts to a protein's native structure, was proposed by Plaxco et al (JMB, 277, 985-994, 1998) as a measure of the topology complexity of a protein structure. Contact order can be used for studying protein folding with higher contact orders indicating longer folding times; it is also a significant parameter that shows correlation with thermostability of protein structures (Structure, 13(6):857-860, 2005).

Given a protein of length L (with L residues), its contact order is the average sequence distance between all pairs of contacting residues normalized by the protein length as follows.

$$CO = \frac{1}{LN} \sum^{N} \Delta S_{i,j} \tag{1}$$

where $S_{i,j}$ is the sequence separation, in residues, between contacting residues i and j, and N is the total number of contacts with $S_{ij} > 0$. For example, $S_{i,j}$ equals 3 for contacting residues 5 and 9. Residues are considered contacting if they contain non-hydrogen atoms that are within a certain threshold, e.g, 6.0 $\mathring{A}$ (inclusive) .

1.1 Write a PDB parser (ContactOrder.py) for contact order calculation (50 points total)

- Your program has two options:
  -p pdbfile (to read in a pdf file)
  -d cutoff (to define the cutoff for defining contact residues, e.g., 6 for 6 $\mathring{A}$)

- Outputs:
  - simple statistics: the total number of atoms (non-hydrogen atoms only) and residues

that have coordinates for the input protein (*note you get 30 points for going this far*)
- contact order

You may use /u/yye/I519/HW5/1FDL-clean.pdb as a test file. Note this is a modified "clean" file with coordinates for only one chain (so you do not need to consider complex structures). If you use -d 6 as the parameter, you should get 0.124 as the contact order for this protein.

2 Motif finding problem

2.1 A definition of the problem

Motif Finding Problem: Given a collection of strings, find a set of k-mers, one from each string, that minimizes the score of the resulting motif.
Input: A collection of t strings Dna and an integer k.
Output: A collection Motifs of k-mers, one from each string in Dna, minimizing SCORE(Motifs) among all possible choices of k-mers.

2.2 A randomized approach to motif finding

```
RandomizedMotifSearch(Dna, k, t);
randomly select k-mers Motifs=(Motif_1, ..., Motif_t) in each string from Dna;
BestMotifs ← Motifs;
while forever do
    Profile ← PROFILE(Motifs);
    Motifs ← MOTIFS(Profile, Dna);
    if SCORE(Motifs) ≤ SCORE(BestMotifs) then
        BestMotifs ← Motifs;
    end
    else
        return BestMotifs;
    end
end
```

Algorithm 1: Randomized Motif Search Algorithm

2.3 How can a randomized approach work?

At first glance, the randomized motif search algorithm appears to be doomed. Let's use an example to see if/how it works. Assume we work on a small set of 5 short strings with the implanted (4,1)-motif ACGT (the sequences are: TTACCTTAAC, GATGTCTGTC, CCGCGTTAG, CACTAACGAG, and CGTCAGAGGT, with the implanted motifs in each

string underlined). Imagine that the algorithm chooses the following 4-mer Motifs at the first iteration: TAAC, GTCT, CCGG, ACTA, and AGGT(where the first motif is from the first string, and so on). A profile matrix PROFILE(Motifs) of the chosen 4-mers can be constructured,

PROFILE(Motifs)

A:	0.4	0.2	0.2	0.2
C:	0.2	0.4	0.2	0.2
G:	0.2	0.4	0.2	0.2
T:	0.2	0.2	0.2	0.4

Now we can compute the probabilities of every 4-mer in the short strings based on the profile matrix (e.g., $P(TTAC|profile) = 0.2 * 0.2 * 0.2 * 0.2 = 0.0016$), and choose the best one for each string. For example, for the first string TTACCTTAAC, we have TTAC (0.0016), TACC (0.0016), ACCT (0.128), CCTT (0.0016), CTTA (0.0016), TTAA (0.0016), TAAC (0.0016). So ACCT will be picked as the Motif for the first string. Once Motifs for all the strings are picked using the same strategy, we have a new set of $Motifs = Motif_1, Motif_2, ..., Motif_t$.

How to score a set of Motifs? We will use a simple scoring function for this assignment:

$$\begin{aligned} SCORE(Motifs) &= d(CONSENSUS(Motifs), Motifs) \\ &= \sum_{i=1}^{t} d(CONSENSUS(Motifs), Motif_t) \end{aligned} \tag{2}$$

where d is the Hamming distance between two strings, here a Motif and the consensus of the set of Motifs.

2.4 Implement the randomized motif finding approach (50 pts + 5 bonus points for correct implementation).

You will write a program called RandomMotif.py, which has two options:
-i for input-file (in FASTA format), and
-k for specifying the length of motifs.
You may use /u/yye/I519/HW5/Dna.fasta as a test file