CS 466 MINI PROJECT: Due sometime around finals week, exact due date to be announced later.

To be performed in teams. For students taking the course for 4 credits, a team should have two members. For students taking the course for 3 credits, a team should have three members. All students in a team will get the same grade.

Feel free to use news forum or piazza to form teams.

Submission:

- **(i)** **A PDF document briefly describing your algorithm,**
- **(ii)** **Excel spreadsheets/pdfs showing performance evaluation, and**
- **(iii)** **PDF document briefly describing any observations from the evaluation phase.**

Demos: ~15 min demos will be scheduled (also around finals week), where each team will explain their report to the instructor and/or TA.

Please consult with the instructor if you have any doubts.

The project involves developing a “motif finding” program *and testing it*. The three major components that you have to think about and implement are:

- Building a benchmark
- Implementing the “motif finder”
- Evaluating the motif finder on the benchmark and making intelligent inferences.

Word of advice: Do not wait until the last week to do everything. In the past, some teams have tried this, confident in their programming abilities, but in many cases they failed to do the last step (of performance evaluation) satisfactorily, because this step took more running time than they had budgeted for!

Step 1: <u>Building a benchmark for motif finding</u>

A benchmark is a collection of synthetic data sets. A synthetic data set is a set of DNA sequences, into which a “motif” (position weight matrix) has been “planted”. To construct a synthetic data set, you will do the following:

1) Take as input:
(a) A positive number called ICPC (“information content per column”)
(b) A positive integer called ML (“motif length”)
(c) A positive integer called SL (“sequence length”)
(d) A positive integer called SC (“sequence count”)

2) Generate SC random sequences (with uniform nucleotide frequencies). Each random sequence has length SL.

3) Generate a random motif (position weight matrix) of length ML, with total information content (as discussed in class) being ICPC * ML.

4) Generate SC binding sites by sampling from this random motif.

5) “Plant” one sampled site at a random location in each random sequence generated in step 2. “Planting” a site means overwriting the substring at that location with the site.

6) Write out the SC sequences into a FASTA format file called “sequences.fa” (Search the web for information on this file format.)

7) In a separate text file (called “sites.txt”) write down the location of the planted site in each sequence. (Use any format, your code will be reading this file later.)

8) In a separate text file (called “motif.txt”) write down the motif that was generated in step 3. The motif should be stored in the format shown in the following example:

```
>MOTIF1    5
10    1     1     8
2     4     4     10
3     3     3     11
1     15    1     3
1     1     15    3
<
```

(This motif has name “MOTIF1”, length 5, and each row after the header is one column of the PWM, with the four numbers representing (unnormalized) frequencies of the nucleotides A,C,G,T respectively in positions 1, 2, … 5.)

9) In a separate test file (called “motiflength.txt”) write down the motif length.

The four files generated in steps 6-9 are stored in a subdirectory, this subdirectory is a “data set”. The numbers ICPC, ML, SL, SC are called the “experiment parameters”. Create data sets for different combinations of these parameters as follows:

Let the default parameter combination be “ICPC = 2, ML = 8, SL = 500, SC = 10”

a) set ICPC = 1, 1.5, 2, while other parameters are at default values.
b) Set ML = 6,7,8, while other parameters are at default values.
c) Set SC = 5, 10, 20, while other parameters are at default values.

You may keep SL fixed at 500.

For each parameter combination prescribed in a-c above, generate 10 data sets. Note that each data set will be different from other data sets, even those with the same parameter combination, because the data set generation is stochastic.

So, there are (1+2+2+2) x 10 = 70 data sets in the benchmark.

Write a program that will generate a benchmark.

Step 2: Implementing the motif-finder

Write a program that will read the “sequences.fa” file and “motiflength.txt” file in a data set and find a motif (position weight matrix) of length given by the “motiflength.txt” file. How you find the motif (i.e., the algorithm) is entirely up to you. Needless to say, your program should not “look at” motif.txt or sites.txt.
The program should produce two output files for the data set:

1) “predictedmotif.txt”, in the same format as “motif.txt” from the data set.
2) “predictedsites.txt”, in the same format as “sites.txt” from the data set. This file will have your program’s prediction of one site per sequence.

Step 3: Evaluating the motif finder on the benchmark

You will write a program/script that will run and evaluate the motif finder on each data set. You are welcome to think of ways to evaluate motif finding performance on a data set, but some obvious criteria are:

1) Relative Entropy between “motif.txt” and “predictedmotif.txt” (search the web for what “relative entropy” means).

2) Number of overlapping positions or overlapping sites between “sites.txt” and “predictedsites.txt”.

3) Running time.

Show the results of your evaluations graphically (e.g., Excel/Matlab/R charts). For instance, you could show the impact of ICPC on each performance metric. For this, you would compute the average performance over the 10 data sets for each value of ICPC, and plot the three averages and standard errors in a line chart.

Rubric:

Method 60%
Evaluation 40%

Methods [60%]
Full points for a correct implementation of either algorithm presented in class, that is deemed reasonably accurate and pratical.

- Extra credit for cool ideas adding to these algorithms, or a cool algorithm. (Presuming it works reasonably well.)
- New heuristics with obvious flaws may be penalized.

Evaluation [40%]

- Evaluation points are awarded for how deeply the method was evaluated, not how good the method itself is. (Within reason, a method that is too bad may make a meaningful evaluation impossible.)
- Should be looking for trends in the results, how well the system performs in relation to the difficulty of the input.

[10%] Analysis of run-time.
[15%] site-overlap.
[15%] Some evaluation of how good the discovered PWM is, and how well it agrees with the motif that was planted. Did students choose good/meaningful metrics here?

Appendix:

Simple algorithm for generating motifs with (approximately) a desired ICPC (credit: Bryan Lunt, former TA of course).

First, we assume a uniform background distribution, and that logarithms are base 2. (which is necessary in order to be able to get an ICPC of 2 for perfectly conserved columns.)

For each column, randomly choose a nucleotide to be the "preferred nucleotide". (This choice should be uniform.) The preferred nucleotide gets probability p, the three others each get probabilities (1-p)/3.

The information content (ICPC) for this one column (and thus the average over the whole motif) will be a function of p. Forming an inverse function that takes ICPC and finds p is a difficult problem.
Fortunately, we do not expect you to solve that problem.
I have approximated (with one exact) the values of p that will render the ICPC's requested in the project.

p ~ 0.8105 (ICPC ~ 1.0)
p ~ 0.9245 (ICPC ~ 1.5)
p = 1.0 (ICPC = 2.0)

Remember that we still want graphs and reports to be in terms of the ICPC, not p, so you may use a lookup table or some other means to get these values into your generator for your experiments.