ASSIGNMENT 1: STRING MAPPING

Goal: The goal of this assignment is to take a complex new problem and formulate and solve it as search. Formulation as search is an integral skill of AI that will come in handy whenever you are faced with a new problem. Heuristic search will allow you to find optimal solutions. Local search may not find the optimal solution, but is usually able to find good solutions for really large problems.

Scenario: You are a Genetics researcher working on phylogeny data – you have gene sequences of various organisms. You want to prove that some organisms are more related than others. You decide to map the strings onto each other and calculate mapping scores between each pair of organisms. The organisms with lower map scores probably are more related. Similarly, later you may have multiple organisms and you want to prove that they are all cumulatively related. The computational task is to compute an overall mapping score for a group of strings.

Problem Statement: There are K strings X_i from the vocabulary V. Each string X_i has length N_i. Your goal is to map the strings to each other. An easy way to do this is to think of this in two steps – conversion and matching. Conversion is a function F that takes in a string and returns another string. All $F(X_i)$s have the same length N. N is greater than equal to all N_is. The function F is only allowed to make one change to the original string – it can introduce dashes. It can introduce any number of dashes at any position. The conversion cost of X to F(X) is CC*number of dashes, CC being a constant. Once all strings have been converted the matching step just matches characters at each position. The matching cost between two characters is given by a symmetric function MC(c1, c2) where c1 and c2 are two characters ϵ V U {-}. Matching cost of two strings is the sum of matching costs of their conversions at each position. Finally, the matching cost of K strings is the sum of pairwise matching costs between each pair.

Example: Let K =3. Let vocabulary V= {A,C,T,G}. Suppose the three strings X_is are:

X_1: ACTGTGA

X_2: TACTGC

X_3: ACTGA

So, for this problem N_1, N_2, N_3 are 7, 6 and 5 respectively. Let all costs be as follows: CC = 3, MC(x,y) = 2 if x, y ϵ V and x != y; MC(x, -) = 1; MC(x, x) = 0. We may define our conversions as follows:

$F(X_1)$: -ACTGTGA

$F(X_2)$: TACT--GC

$F(X_3)$: -ACTG--A

With these conversions N = 8. The conversion costs are respectively 3, 6, and 9. The matching cost between $F(X_1)$ and $F(X_2)$ is 1+0+0+0+1+1+0+2 = 5. Similarly between $F(X_2)$ and $F(X_3)$ is 1+0+0+0+1+0+1+2=5 and between $F(X_1)$ and $F(X_3)$ is 0+0+0+0+0+1+1+0=2. Hence the total matching cost of this conversion is 5+5+2=12.

The final output cost of this mapping problem is sum of conversion and matching costs = 3+6+9+12=30.

Your goal is to find a conversion with the lowest final cost. (The current solution is not the optimal solution for this problem).

Input format:

Time (in mins)

|V|

V

K

X1

X2

…

CC

MC

#

Here is the input format for the example

5.5

4

A, C, T, G

3

ACTGTGA

TACTGC

ACTGA

3

0 2 2 2 1

2 0 2 2 1

2 2 0 2 1

2 2 2 0 1

1 1 1 1 0

#

Here is the format of MC is that it is representing a matrix of |V|+1 rows and columns. The last row and column is for dash. All diagonal entries will be zero. Any other entry represents the MC between the appropriate characters in V (based on the order they appear in V in the input line number 3). # represents end of the file.

For this problem you are given 5.5 mins of wallclock time to solve the problem.

Output format:

F(X1)

F(X2)

…

In our example

-ACTGTGA

TACT--GC

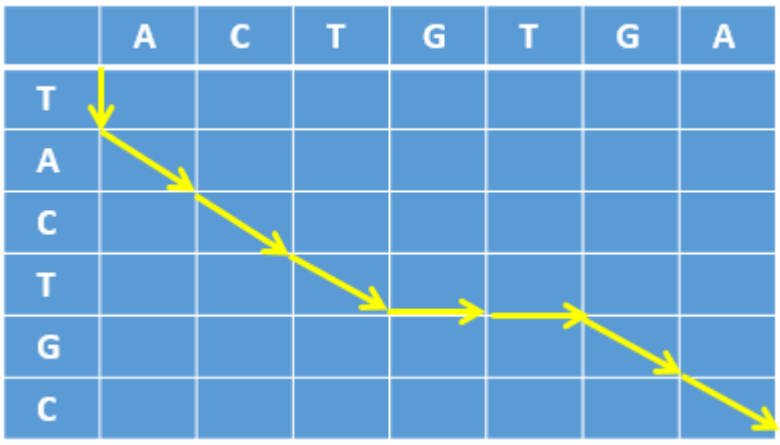
-ACTG—A

Basic Algorithms you can try:

1. Heuristic Search: Design a state space and transition function for this problem. Define a heuristic function for evaluating a state. Implement A* (or variants) and measure quality of solutions found (and scalability). If heuristic is admissible – quality is optimal but algorithm may be slower. Test a couple of heuristics if you can design them.
2. Branch and Bound: Design a state space and transition function for this problem. Optionally define a heuristic function for evaluating a partial schedule. Also, optionally, define a ranking to pick the next successor. Implement Depth First Branch and Bound (or variants) and measure scalability.
3. Local search: Implement a neighbor function for this problem. Implement local search (or variants). Measure of quality of best solution found as a function of time or other model parameters.

Recommended: start with local search as your base algorithm.

Hint (State Space Formulation): As a hint (you may of course choose to ignore it), here is a way to formulate the pairwise mapping problem as a shortest path problem in a matrix where the row is one string and the column is the other. For example, ACTGTGA and TACTGC are mapped in our running example using the following path below:



One can similarly generalize the path to K dimensions for K string mappings.

Hint (Dynamic Programming): An astute reader may observe that there is a dynamic programming formulation to this problem. We won't stop you from implementing that, however, dynamic programming does not scale very well for bigger K. Heuristic search or branch & bound might do much better for K=5 or more. And local search might be best for much larger K values. We will definitely test on problems with large enough K to dissuade you from submitting a dynamic programming solution (the goal of the course is AI techniques). That said, dynamic programming might be a subcomponent of the final good solution.

Code: You may program the software in any of C++, Java or Python. The versions of the compilers that will be used to test your code are

JAVA: java version "1.7.0_79" (OpenJDK)
Python 3.6
g++ 4.8.1

What is being provided? We are providing you with a format checker. The format checker outputs "False" if the output format is not according to the specifications: the output strings are not of the same size, or have out-of-vocabulary characters, or do not match with corresponding input strings. If the format is correct, it also gives the cost of the output.

Usage: format_checker.py input.txt output.txt

Code verification before submission: Your submission will be auto-graded. This means that it is absolutely essential to make sure that your code follows the input/output specifications of the assignment. Failure to follow any instruction will incur significant penalty.

We shall be generating a log report for every submission within 12 hours of submission. This log will let you know if your submission followed the assignment instructions (format checker, scripts for compilation & execution, file naming conventions etc.). Hence, you will get an opportunity to resubmit the assignment within half a day of making an inappropriate submission. However, please note that the late penalty as specified on the course web page will still apply for resubmissions beyond the due date. Exact details of log report generation will be notified on Piazza soon.

Also, note that the log report is an additional utility in an experimental stage. In case the log report is not generated, or the sample cases fail to check for some other specification of the assignment, appropriate penalty for not adhering to the input/output specifications of the assignment will still apply at the time of evaluation on real test cases.

What to submit?

1. Submit your code in a .zip file named in the format **<EntryNo>.zip.** If there are two members in your team it should be called <EntryNo1>_<EntryNo2>.zip. Make sure that when we run "unzip yourfile.zip" it contains a directory with the same name as the zip file and the following files are present in that directory:
   compile.sh
   run.sh
   writeup.txt

   You will be penalized for any submissions that do not conform to this requirement.

   Your code must compile and run on our VMs. They run amd64 Linux version ubuntu 12.04. You are already provided information on the compilers on those VMs. These configurations are similar to GCL machines like 'todi' (have a RAM of 16 GB). We will use ./compile.sh to compile your code.

Your run.sh should take 2 inputs, someinputfile.txt and someoutputfile.txt. It should read the first input as input and output the answer in the outputfile.
./run.sh input.txt output.txt (please note any name could be given to the two files).

2. The writeup.txt should have two lines as follows
   First line should be just a number between 1 and 3. Number 1 means C++. Number 2 means Java and Number 3 means Python.
   Second line should mention names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None.
   After these first two lines you are welcome to write something about your code, though this is not necessary.

Evaluation Criteria

Final competitions on a set of similar benchmark problems. The points awarded will be your normalized performance relative to other groups in the class. Extra credit may be awarded to standout performers.

What is allowed? What is not?

1. You may work in teams of two or by yourself. We do not expect a different quality of assignment for 2 people teams. At the same time, please spare us the details in case your team cannot function smoothly. Recall, that you can't repeat a team in the COL333 course (and repeat a team only once for COL671). If you are short of partners, our recommendation is that this assignment might not be that hard and a partner should not be required.
2. You cannot use built-in libraries/implementations for search or computational biology algorithms.
3. While you are allowed one of three languages we recommend that you use C++ since it produces the most efficient code. This assignment requires you to produce the best solution within a given time constraint.
4. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.
5. Please do not search the Web for solutions to the problem.
6. Your code will be automatically evaluated. You get a zero if your output is not automatically parsable.
7. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.