Assignment Three: Median String

* **Due** Friday by 11:59pm
* **Points** 5
* **Available** Jul 7 at 12am - Jul 22 at 11:59pm 16 days

BOOK section: [motif and median string algorithms.pdfreview the documentiew in a new window](https://usu.instructure.com/courses/414327/files/61771784/download?wrap=1)

Power point slides: [Ch04\_Motifs\_Flann.pptxreview the documentiew in a new window](https://usu.instructure.com/courses/414327/files/61771791/download?wrap=1)

Implement a branch and bound algorithm for finding the median string given a set of **t** DNA sequences of length **n** and a target median string length of**L**.

Here is a sketch of the algorithm: (NOTE, I use indentation to denote nesting of statements as in Python)

globalBestScore = +infinity

Def MedianString(length, ms): // length is the number of characters in the remaining median string initialized to L, ms is the median string, initially ""

if length==0:   // Have the median string of the correct length  
   score = totalDistance(ms, DNA) //DNA is a global tXn array of DNA sequences  
   if score < globalBestScore :  // update the best global solution and score  
      globalBestScore = score   
      bestMedianString = ms

   return

// apply the bound once the ms string is at least 4 characters long (you can experiment with this)  
if   len(ms) >= 4 and (totalDistance(ms,DNA) > globalBestScore ):   
      return

// otherwise, keep trying new combinations  
for base in {A,G,C,T}  
    MedianString(length-1,ms+base)  //note the plus adds a character to ms and returns a**new** string

Total distance is defined in the attached powerpoint and pages from the book. NOTE: the book makes a big fuss about "search trees." I would simply ignore that bit and focus on the simple recursive algorithm above. There is no need to deal with these trees if we write the code recursively. Recursion handles all the "tree search" automatically in its use of the stack to keep track of the calls as it goes down the recursion.

Test the code on some synthetic data that is created to include known median strings. A simple test would be to generate random sequences of AGCT's of length say 200 characters, then create a median string of length **L** (say 12) and insert it into a random position in each sequence. Next, try the algorithm when the median string is mutated by one character when inserted into each DNA sequence. How does the method perform? What about two mutations? Try more until the system fails to find the median string you inserted.

How does the system perform w.r.t run time? Start with small examples and build up to larger examples. We expect the system to be linear in **n**, but near exponential in L. The effectiveness of the bound will dictate how the runtime grows as a function of L.

Here is a file of 24 the promotor region of 24 human genes.

[promotorRegionsSample.txtreview the documentiew in a new window](https://usu.instructure.com/courses/414327/files/61775725/download?wrap=1)

In the file lines that look like: >hg38\_knownGene\_uc057aup.1 give the name of the gene. The following lines contain 1000 DNA bases (a g c t) that make up the sequence immediately before the start of the gene. It is in this region that biologists are interested in transcription factor binding sites (which computer scientists find as median strings). Run your code on this real data to see what it finds. set L to around 12.

For this assignment submit:

Your code, examples of your code running and producing results.

* A brief report that includes:
  + A brief description of the biological problem and its motivation
  + A formal CS/Mathematical definition of the problem (given, find, such that) with variables (such as **L, n** etc.) defined.
  + Your results obtained for the synthetic data
  + Your results obtained for the real data.