**Median String Problem**

**Problem Description**:

Median String Problem aka Motif Finding problem is the problem of discovery of related patterns of a specified length in a collection of DNA fragment.

**Motivation**:

The goal of motif finding is the detection of novel, unknown signals in a set of sequences. For example you might want to detect transcription factor binding sites in a genome, but do not know what they look like.

Unfortunately, such motifs are never conserved exactly but only approximately. This fact makes the problem difficult. We want to solve this problem algorithmically. [1]

**Mathematical Definition**:

Given a set of **t** DNA sequences of length **n**, and a target *Median String* of length **L,** we want to find a Median String that maximizes the Consensus Score.

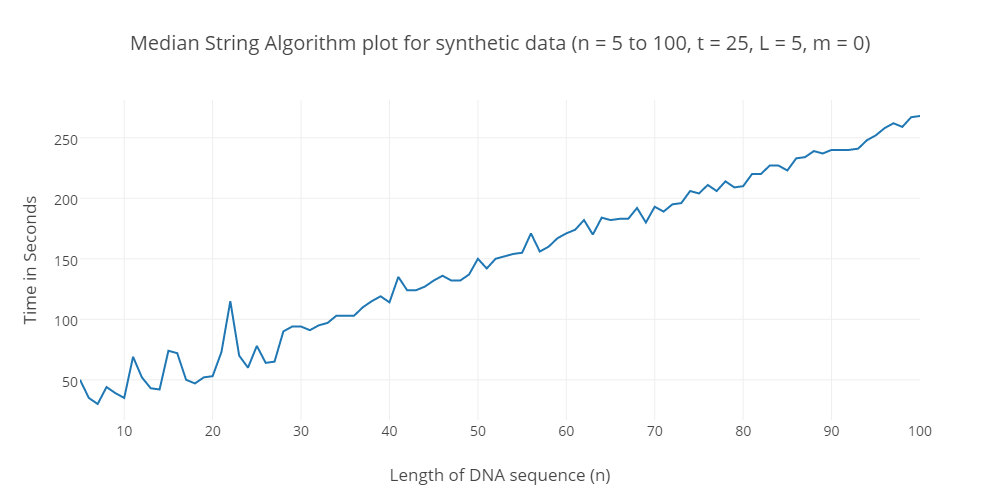
**Results obtained from synthetic data:**

To generate synthetic DNA data I have credited a method “*findMedianStringInSyntheticData”,* which takes in4 parameters:

1. Length of DNA sequence (**n**)
2. Number of DNA sequence (**t**)
3. Length of Median String (**L**)
4. Number of Mutations (**m**)

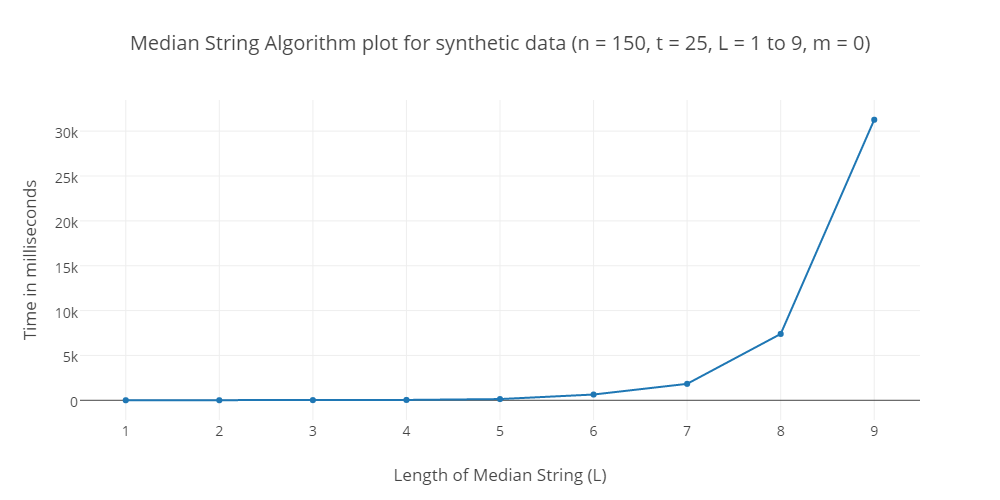
Next, I have used *findMedianStringInSyntheticData* to generate DNA sequences containing different number of Median String and Mutations. Then I have applied the *medianString* algorithm to this synthetic data to see if the algorithm finds the original median-string/motif or not, and how does the algorithm perform.

* For variable length of DNA sequence (**n** = 5 to 100, **t** = 25, **L** = 5, **m** = 0)



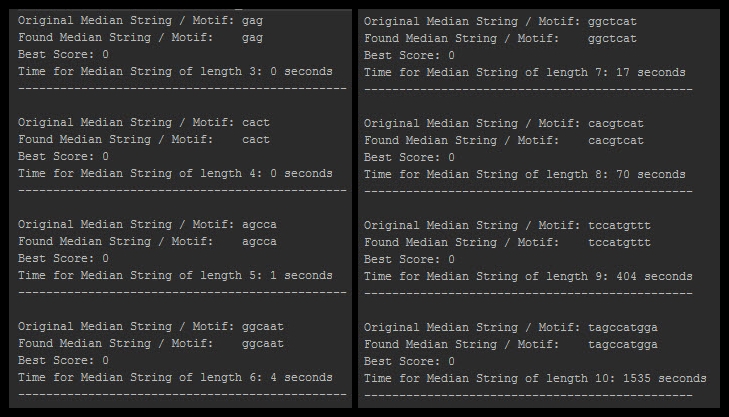
**Observation**: As per expectations, the algorithm runs in linear time, as we increase the length of DNA sequence, as shown in the plot above.

* For variable length of Median String (**n** = 5 to 100, **t** = 25, **L** = 5, **m** = 0)



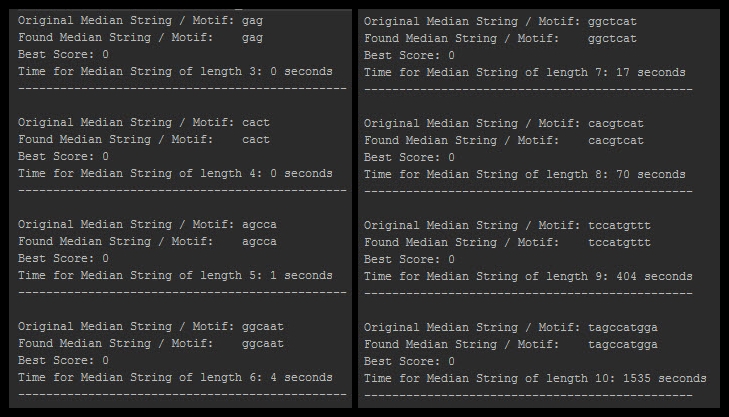
**Observation**: The algorithm runs in exponential time, as we increase the length of Median String, as shown in the plot above, i.e. the algorithm becomes inefficient in finding the median string as its size increases.

* For 0 Mutation (**n** = 250, **t** = 250, **L** = 3 to 11, **m** = 0)



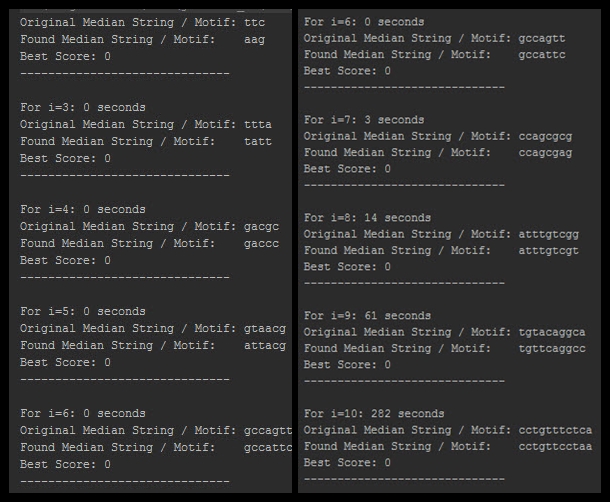
**Observation**: In the figure above we can see that the algorithm successfully find all the original median string from the synthetic data with 0 mutations. Its run time is also same as the pervious plot.

* For 1 Mutation (**n** = 250, **t** = 250, **L** = 3 to 11, **m** = 1)



**Observation**: In the figure above we can see that the algorithm does not successfully find all the original median string from the synthetic data with 1 mutations. For L = 3 to 6, it fails to find the original motif.

* For 2 Mutation (**n** = 250, **t** = 250, **L** = 3 to 11, **m** = 2)

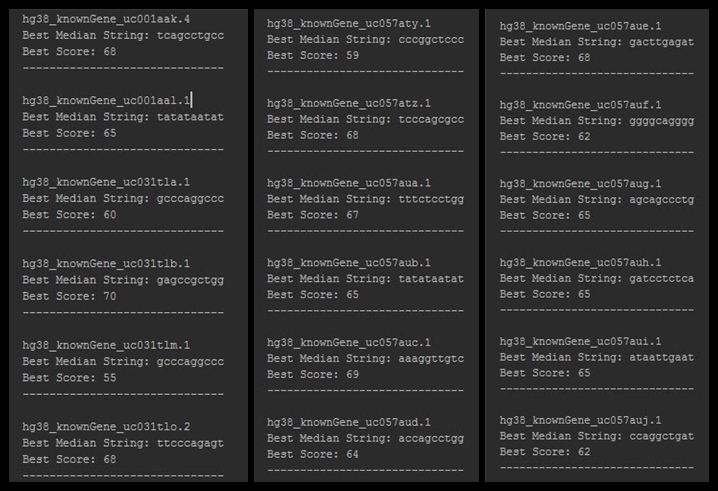


**Observation**: In the figure above we can see that the algorithm does not find any correct median string from the synthetic data with 2 mutations. So its success rate is 0%.

* *For L > 11, the algorithm takes too much time to find the median string, I kept my computer running for more than 2 hours, and even then the algorithm didn’t complete execution.*

**Results obtained for the real data:**

* I ran the algorithm for the DNA samples in the [promotorRegionsSample.txt](https://usu.instructure.com/courses/414327/files/61775725/download?wrap=1) file with value of L as 10.
* It took ~10 minutes to run the algorithm for all 24 DNA data.
* The image below shows output for some of the DNA samples:
  1. The name of the file.
  2. Best median string found.
  3. Best score.



Citations:

[1]: ab.inf.uni-tuebingen.de/teaching/ws08/seqan/Motif-Finding.pdf