MARMARA
UNIVERSITY

CSE4065 - Introduction to Computational Genomics

Assignment 1 Report
November 18, 2021

Yusuf Talha Erdem (150117012)
Sena Dilara Yangöz (150119706)
Ali Reza Ibrahimzada (150119870)

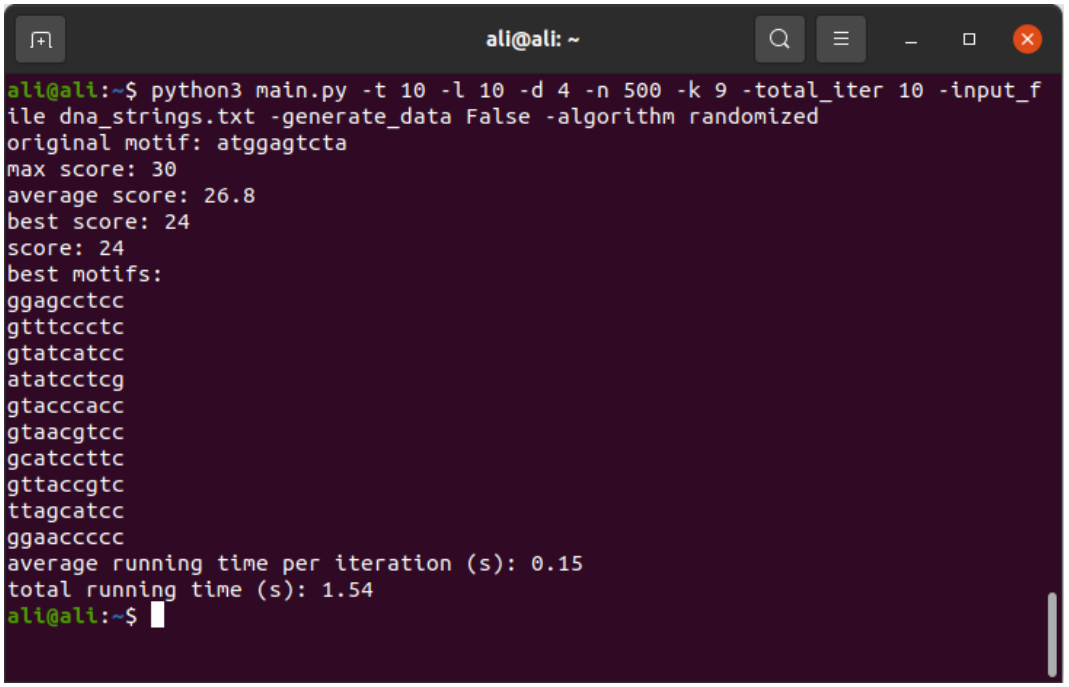
Prior to executing the algorithms on the randomly generated DNA strings, we randomly generated the motifs, DNA strings, positions of the mutations, base pair mutations, positions of the mutated motifs in the DNA string, etc. Then for the rest of the project, we simply use the same DNA strings we have generated in the beginning of the project.

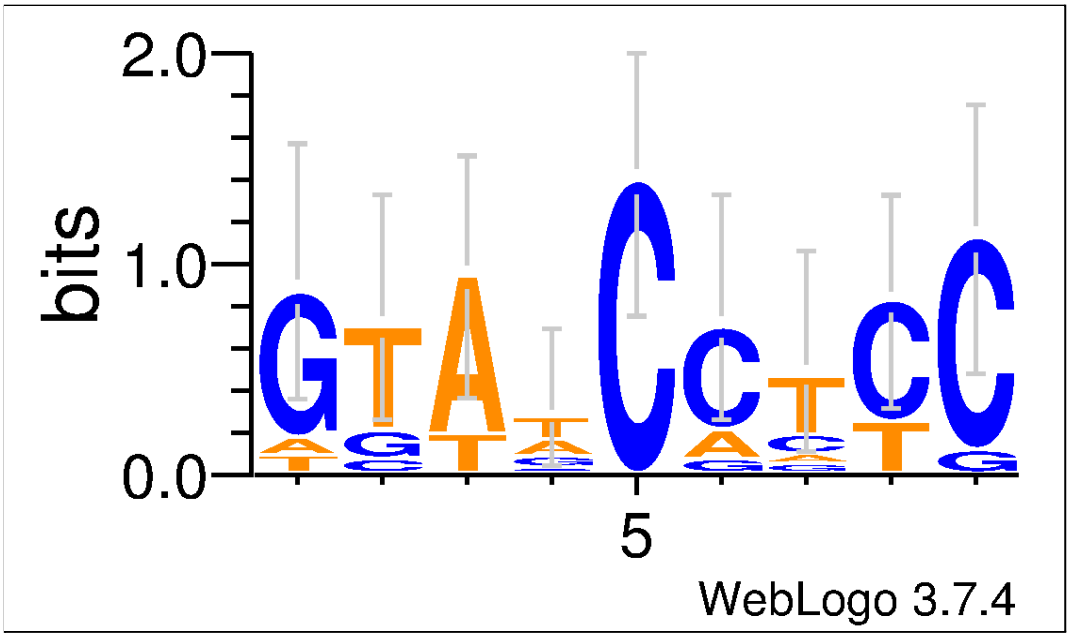
Randomized Motif Search

For the Randomized Motif Search algorithm, we started by choosing 10 random motifs from the DNA strings. Then, we randomly select initial motifs from the DNA strings, set it to best motifs, and calculate a profile based on the initial motifs using the *get_profile* function. Furthermore, we call the *get_likely_motifs* function in order to extract the set of kmers with the highest probability to be our motifs. The function runs a sliding window over the DNA string and calculates the probability at each iteration. Later at the end, it returns the set of most probable motifs from the DNA strings. At the end, we check if there is any improvement in the overall score of the algorithm compared to the previous iteration. That is, if there is any improvement in the score, we re-iterate and use the new motifs as our best one, otherwise we end the iteration and return the current best scores. What we have explained so far describes a single run of the Randomized Motif Search algorithm, we continue doing this for a total of 10 runs and report different statistics about this iterative algorithm at the end.

K = 9:

Original Motif (Randomly Generated): ATGGAGTCTA
Algorithm Output (Consensus String): GTATCCTCC-

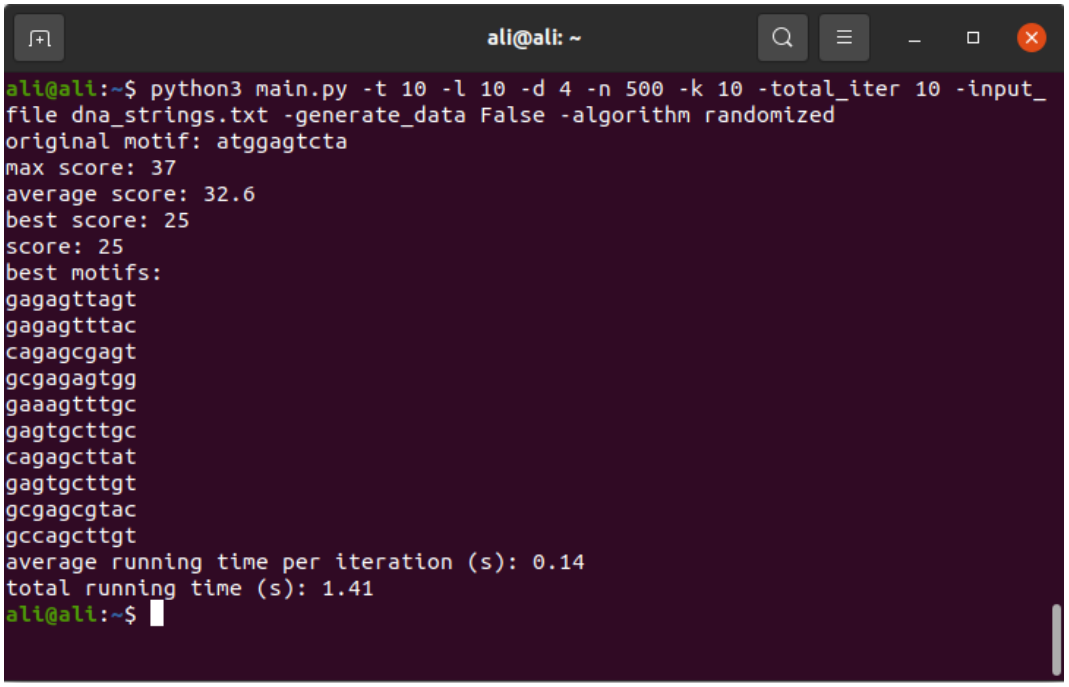


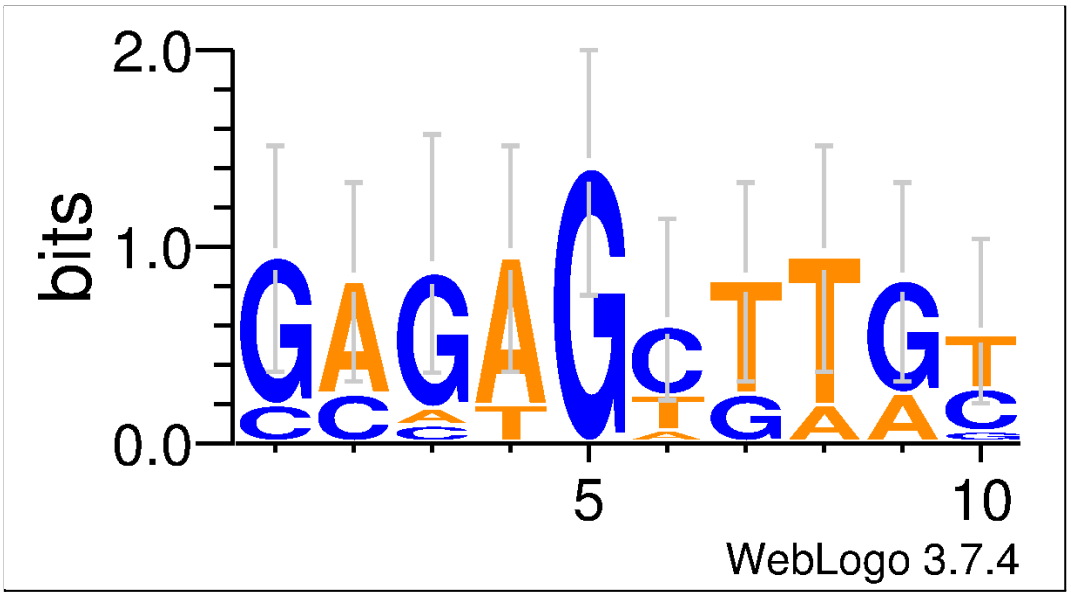


K = 10:

Original Motif (Randomly Generated): ATGGAGTCTA

Algorithm Output (Consensus String): GAGAGCTTGT

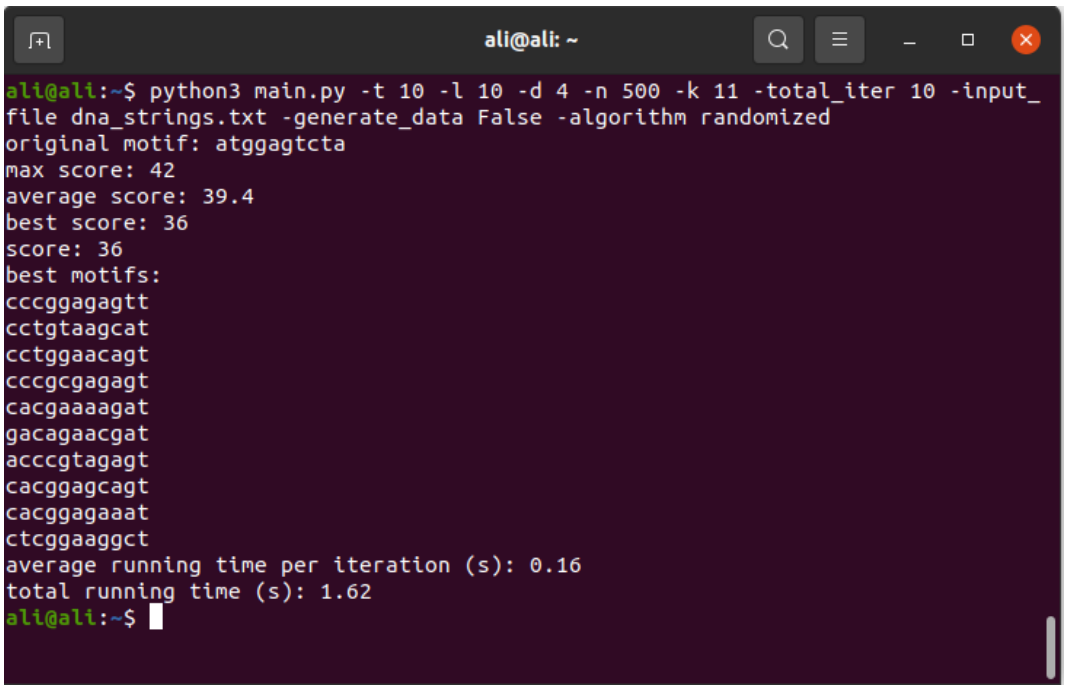


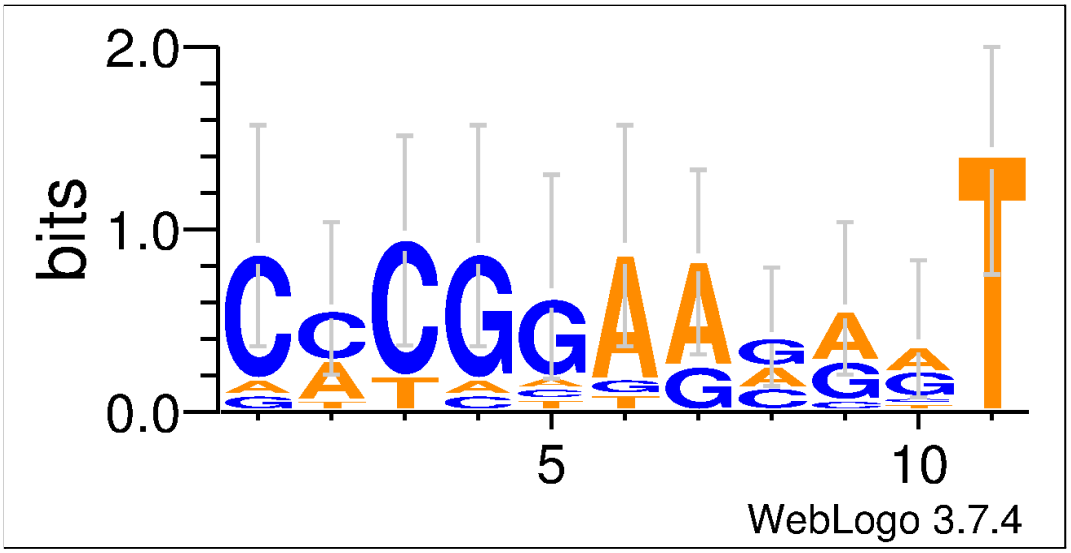


K = 11:

Original Motif (Randomly Generated): ATGGAGTCTA-

Algorithm Output (Consensus String): CCCGGAAGAAT

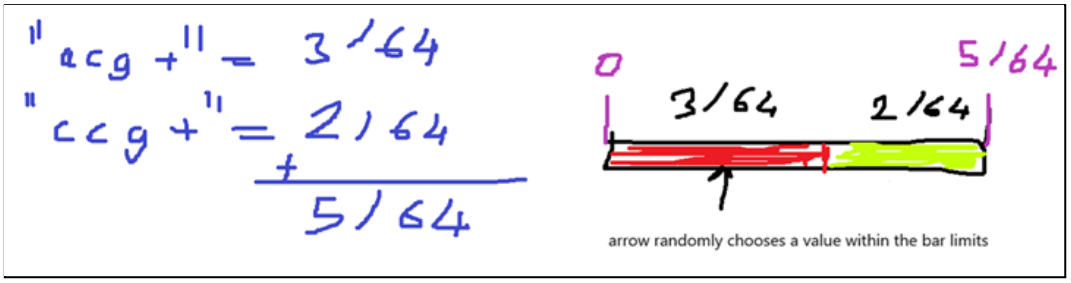




Gibbs Sampling

For the Gibbs Sampling algorithm, we started by choosing 10 random motifs from the DNA strings like we did in Randomized Motif Search. Then, since one of those motifs should be eliminated randomly, we randomly selected the motif to be eliminated and removed it from the list which contains the initial motifs. After creating a profile with the *get_profile* function for the rest of the 9 motifs, with the *calculate_prob_of_deleted_string* function the probabilities of the k-mers which are in the eliminated DNA strings is calculated and kept in a dictionary (key = k-mer, value= probability of the k-mer). After that, with the *roll_dice* function a k-mer is selected randomly but the one with the highest probability is

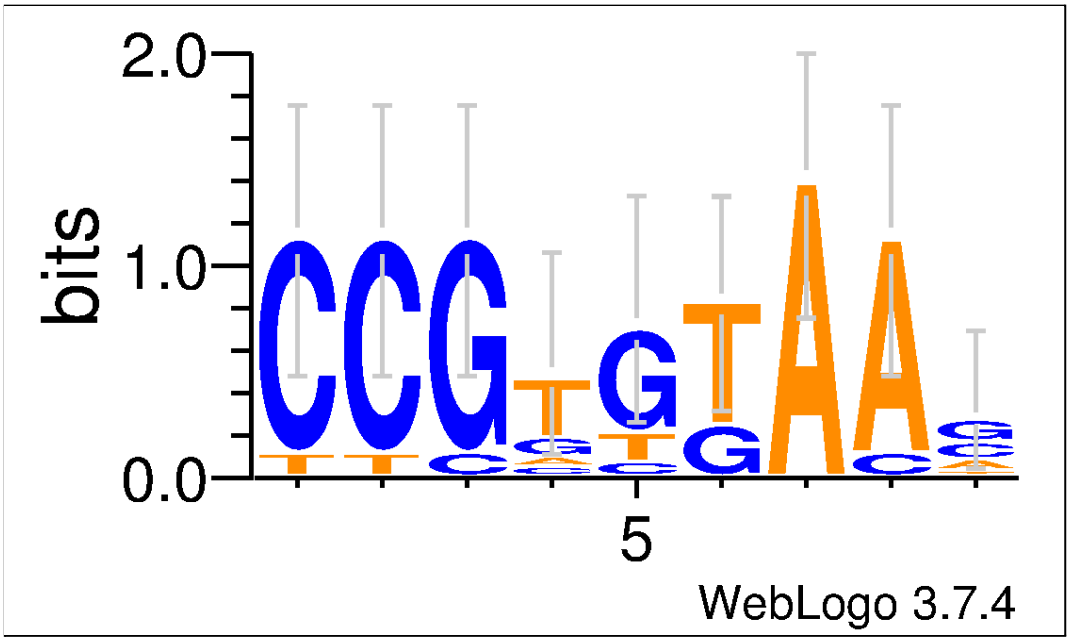
more likely to be chosen. *roll_dice* takes probabilities as an argument. What is stored in the probabilities is possible motives and their matches with profile. Such as “atgt” : 3/64, “ccgt” : 2/64. To randomly choose a motif, we calculate the sum of probability of all motifs. Then we choose a random value between 0 and the sum of motives and next we need to check which motif this random value is corresponding to then we return to the motif. Then the new k-mer is added to the motifs list and the score is compared with the best score. If the new score is better than the best score, the best score is synchronized to the new score and the loop counter is reset, otherwise, the loop counter is incremented by one. When the loop counter is equal to 50, this means that the score has not been improved for the last 50 iterations, so the loop ends. An illustration of how the dice rolling works is given below:



K = 9:

Original Motif (Randomly Generated): ATGGAGTCTA

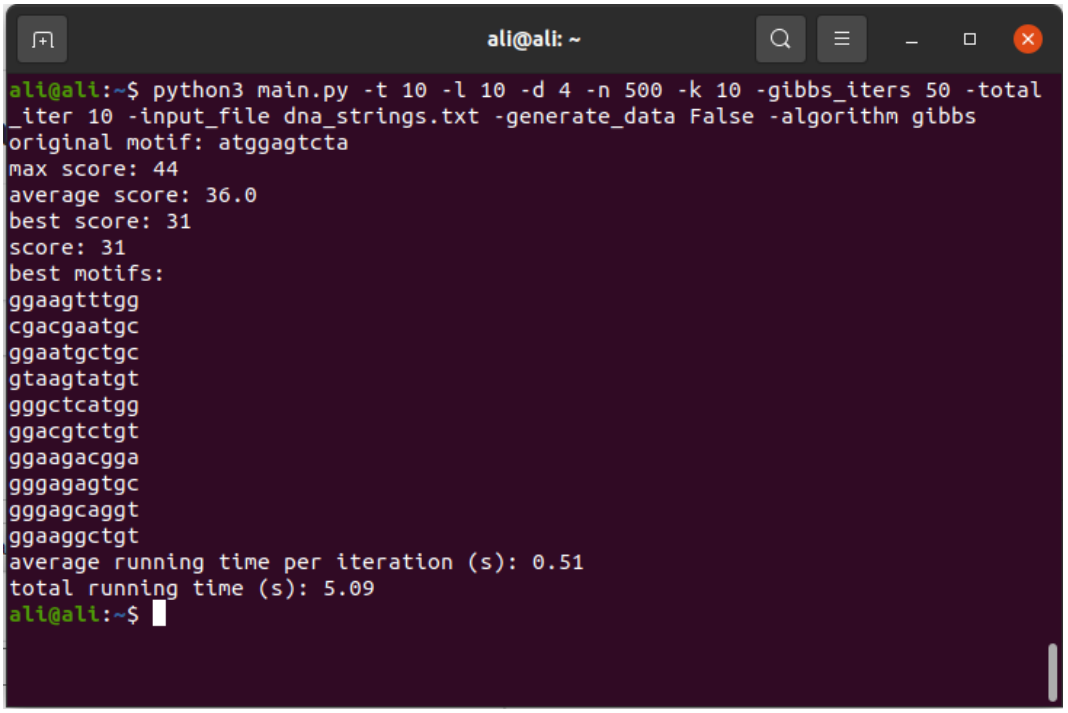
Algorithm Output (Consensus String): CCGTGTAAG-

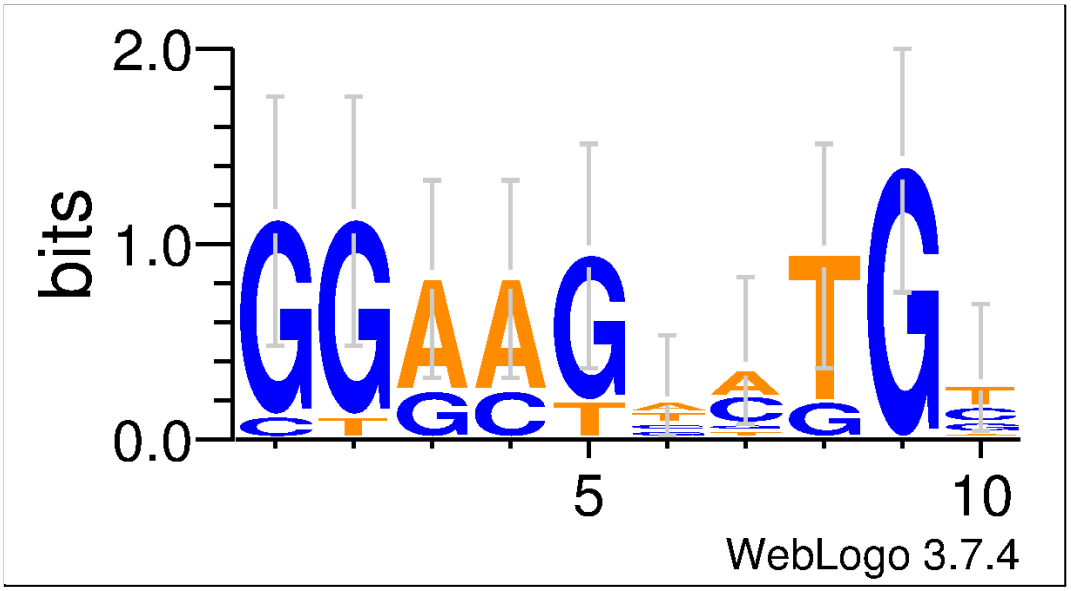


K = 10:

Original Motif (Randomly Generated): ATGGAGTCTA

Algorithm Output (Consensus String): GGAAGAATGT

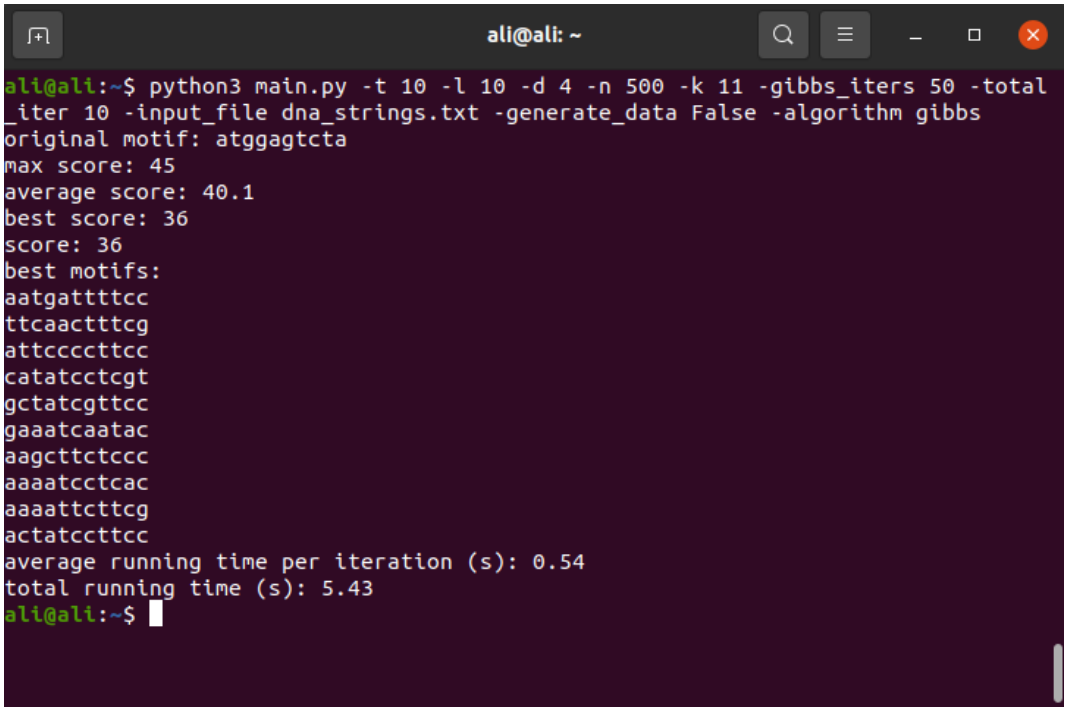


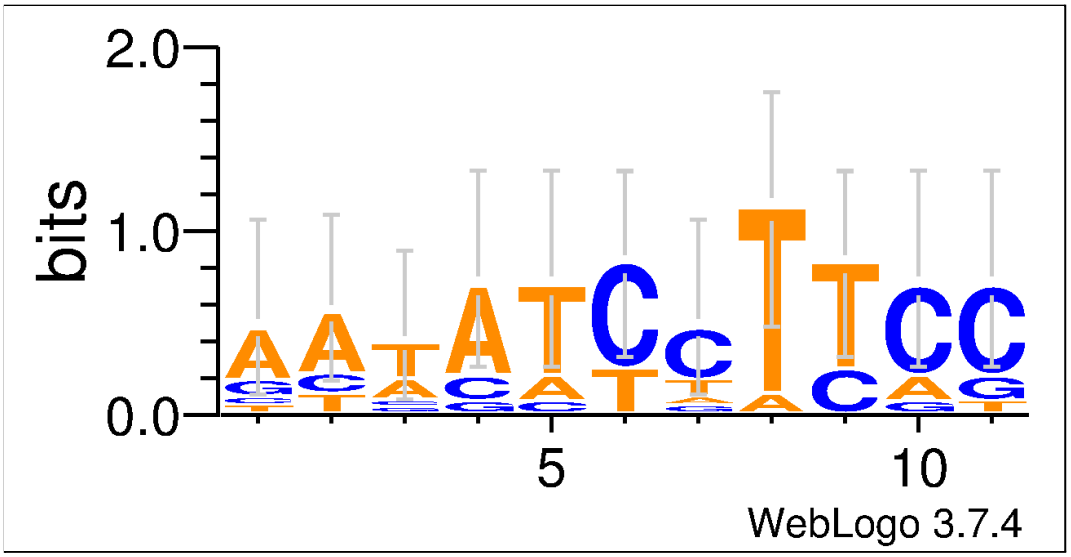


K = 11:

Original Motif (Randomly Generated): ATGGAGTCTA-

Algorithm Output (Consensus String): AATATCCTTCC





Comparison and Conclusion

In this section, we will compare the performance of the Randomized Motif Search (RMS) with the Gibbs Sampling algorithm. In terms of the running time, Gibbs Sampling takes a significantly longer time compared to RMS because of the improvement waiting in Gibbs. In contrast, RMS does not wait for another iteration if there is no improvement in the score value. Moreover, it is visible from the score graph that as the value of k increases, the overall score of both algorithms increase as well. It can be concluded that although Gibbs takes a lot of time to produce its best motifs, RMS produces almost the same scores in a much shorter amount of time. On the other hand, we must also mention that

Gibbs algorithm has the potential to produce better scores if we let it run for longer iterations.