Student: Miriam Briskman
Faculty Adviser: Professor Sokol

CISC 5001 Project Proposal
Spring 2020 – Brooklyn College

Efficient Implementation of the Tandem Repeats Software (TRed)

Background

A **biological sequence**, or just a **sequence**, is a chain of ordered sub-molecular building blocks in living organisms. Biological sequences such as DNA, RNA and proteins dictate the physiological characteristics of the organism and are responsible for maintaining healthy cell activity. A **substring** is an ordered collection of zero or more neighboring building blocks. **Tandem repeats** are multiple adjacent copies of nonempty substrings in a given sequence. For instance, the DNA sequence `CAGTGTGTCT`, constructed from the bases `A, C, G` and `T`, introduces the tandem repeat `GTGTGT`, for the substring `GT` is being contiguously repeated 3 times. The phenomenon of tandem repeats has been profoundly studied and related to the individual organism's identification on the one hand and the discovery of diseases on the other [7].

Evolutive tandem repeats are repeats in which each copy is based on the neighboring copy that existed prior the former's emergence [6]. Since the process of copying nucleotide bases involves errors, among which are insertions, deletions, and mismatches, scientists strive to collect statistical data regarding such approximate tandem repeats. For example, the tandem repeat `GTGTGT` in the DNA sequence depicted above could be extended to `GTGTGTCT`, assuming that the `C` in the rightmost copy was erroneously generated by copying `G`. Since the presence of an error can be interpreted differently, many definitions of what constitutes an approximate tandem repeat have been proposed, following differing approaches well summarized in [6] on how to discover all such repeats in a given sequence. A comprehensive definition called **edit distance** considers those tandem repeats in which the cumulative errors from one copy to the other do not exceed a specified number. In particular, a tandem repeat is counted as ***k*-edit** if it contains at most k cumulative errors of the kinds listed above.

Materializing this definition, one may find all the tandem repeats over the edit distance in a sequence S of size n by using an intuitive algorithm described in [6]. It employs the notion that a k-edit repeat exists if and only if a prefix of S and a suffix of S can be aligned such that the amount of errors does not exceed k. The information about the errors in every alignment is stored in a $n \times n$ matrix that is filled using a dynamic programming approach. The procedure has the running time of $\mathcal{O}(n^4)$: per each of the $\mathcal{O}(n^2)$ iterations, a $\mathcal{O}(n^2)$-size computation takes place. However, several enhancements can considerably reduce this asymptotic upper bound:

- Main and Lorentz [4] suggest a recursive method that requires only $\mathcal{O}(\log(\frac{n}{k}))$ iterations. Their idea is to scan a given substring for tandem repeats that cross its center, halve the substring, and apply the same search procedure on the new two substrings. Thereby, the overall work decreases to $\mathcal{O}(n^2 \log(\frac{n}{k}))$.
- Landau and Vishkin [3] show that, given that one wishes to obtain repeats with at most k errors, it is not necessary to fill the entire $n \times n$ matrix; only those entries on the diagonals corresponding to at most k errors should be determined. The recommendation is to use suffix trees requiring only near linear work. Hence, the amount of computations done per each iteration reduces to $\mathcal{O}(nk)$ on the average case, leading to $\mathcal{O}(nk \log(\frac{n}{k}))$ running time.
- Noticing the worst case running time of $\mathcal{O}(n^2 k)$ of algorithm in [3], Landau, Myers, and Schmidt [2] claim that each alignment conveys a great deal of information about the subsequent alignment, thus eliminating the need of computing a matrix from scratch. The proposed technique uses computations of size $\mathcal{O}(nk \log k)$ per each iteration, implying that the entire procedure can be done in $\mathcal{O}(nk \log k \log(\frac{n}{k}))$ time in the worst case.

Over the last decade, the online Tandem Repeats Database (TRedD) was established and has constantly been updated by Sokol et al [5, 6]. This website exhibits the results of searching for all the tandem repeats over the edit distance in chromosomes $1 - 22$, X and Y of the human being, inspired by the grandiose completion of the Human Genome Project in 2003 and using the information derived thereby. The software outputting these findings, TRed [5], which was written in C++, implements the first enhancement and the principle of the second that suggests looking only at the elements with at most k errors, but does not use suffix trees. Despite those upgrades, TRed runs for 20 hours on average to scan a single chromosome for tandem repeats.

Schemes for other effective methods have been raised but have yet been incorporated into TRed. In Fall 2014, Cohen [1] programmed the algorithm of Landau and Vishkin [3] in Java, and it was later translated into C++ by Yocheved Levitan. The program utilizes suffix arrays mimicking the suffix trees structure discussed in the original paper, demanding only linear storage and close-to-linear ($\mathcal{O}(nk)$) running time. The incorporation of suffix arrays into TRed could noticeably improve its efficiency.

Objectives

This study has four cardinal purposes:

1. Integrating the suffix array version of the Landau and Vishkin algorithm into the TRed software by extending its current version,
2. Analyzing how both the iterative and computational processes in TRed could be additionally simplified to increase the efficiency of the program, and applying these findings to the program as far as is practicable,
3. Running the improved version of TRed on the human chromosomes to compare its running time with this of the current version, and
4. Publishing this improved version on the Tandem Repeats Database website, providing global access to the program.

References

[1] Cohen, B. (2015). Implementation of the Landau Vishkin Algorithm [GitHub Repository; Computer Software]. Retrieved from https://github.com/Brani/Implementation-of-the-Landau-Vishkin-Algorithm

[2] Landau, G. M., Myers, E. W., & Schmidt, J. P. (1998). Incremental string comparison. *SIAM Journal on Computing, 27(2), 557-582.* doi: 10.1137/S0097539794264810

[3] Landau, G. M., & Vishkin, U. (1989). Fast parallel and serial approximate string matching. *Journal of algorithms, 10(2), 157-169.* doi:10.1016/0196-6774(89)90010-2

[4] Main, M. G., & Lorentz, R. J. (1984). An O (n log n) algorithm for finding all repetitions in a string. *Journal of Algorithms, 5(3), 422-432.* doi: 10.1016/0196-6774(84)90021-X

[5] Sokol, D., & Atagun, F. (2010). TRedD—A database for tandem repeats over the edit distance. *Database, 2010.* doi: 10.1093/database/baq003

[6] Sokol, D., Benson, G., & Tojeira, J. (2007). Tandem repeats over the edit distance. *Bioinformatics, 23(2), e30-e35.* doi: 10.1093/bioinformatics/btl309

[7] Sokol, D., Adkins, F., Che, Z., & Pfabe, K. (2009). Finding Repeats Within Strings. *DIMACS Educational Module Series, Module 09-2.* Retrieved from http://archive.dimacs.rutgers.edu/Publications/Modules/Module09-2/dimacs09-2.pdf