UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

PROJECT

Fast sequence alignment

Kristijan Biščanić

Luka Hrabar

Ela Marušić

Zagreb, January 2016.

CONTENTS

1. Introduction **1**

2. Algorithm **2**

3. Test results **3**

4. Conclusion **5**

5. Bibliography **6**

1. Introduction

This project will implement, test and analyse a faster algorithm for computing string edit distances and sequence alignment. This algorithm was published by Masek and Paterson [3] and is inspired by the Four Russians Algorithm. Implemented algorithm will be compared to and tested against Needleman-Wunsch algorithm [4], which is based on dynamic programming.

The *string edit distance* is defined as the minimal cost of transforming one character string into the other. Operations allowed in those transformations are only insertion, deletion and replacing of one character, each of these having defined some cost. *Edit Script* is defined as the actual sequence of operations used to transform one string into the other. There are many algorithms that are using string edit distances and edit scripts for further calculations, and they are used extensively in bioinformatics for sequence alignment.

Sequence alignment is a process of arranging the symbolic representations of DNA, RNA or protein sequences so that their most similar elements are juxtaposed. Such alignment is useful to identify regions of similarity and many bioinformatics tasks depend upon successful alignments.

2. Algorithm

3. Test results

Our algorithm was tested on synthetic sequences. Pairs of sequence reads were simulated using wgsim tool from reference chromosome of Escherichia coli bacteria, and then converted into FASTA file format which is used as an input file format. Output file format of our program was MAF (*Multiple Alignment Format*) containing score, which is equal to string edit distance, and sequence alignment for algorithms that support it.

Pairs of sequence reads were generated with lengths of 100, 200, 500, 1 000, 5 000, 10 000, 50 000, 100 000, 500 000 and 1 000 000 characters. Every pair was then tested on our algorithm, as well as Needleman-Wunsch algorithm for reference. Every test was timed and maximal memory usage was observed for each of the algorithms. Needleman-Wunsch algorithm was tested on sequences up to 100 000 characters because of its large time complexity. Both algorithms were tested in same conditions on the same computer running inside Bio-Linux-8 virtual machine with access to 2 CPU cores and 2GB RAM.

N	Needleman-Wunsch	Masek-Paterson	Masek-Paterson (alignment)
100	0.483 ms	$2s$	
200	1.633 ms	$2s$	
500	7.954 ms	$2s$	
1000	44.34 ms	$2s$	
5000	847.7 ms	$2s$	
10000	3.253 s	$2s$	
50000	79.94 s	$2s$	
100000	331.6 s	$2s$	
500000	$2 - 3$ h [1]	$2s$	
1000000	$8 - 10$ h [1]	$2s$	

Table 3.1: Time comparison

[1] estimated

N	Needleman-Wunsch	Masek-Paterson	Masek-Paterson (alignment)
100	1756 KB	$2s$	
200	1756 KB	$2s$	
500	1768 KB	$2s$	
1000	1772 KB	$2s$	
5000	1836 KB	$2s$	
10000	1924 KB	$2s$	
50000	2464 KB	$2s$	
100000	3188 KB	$2s$	
500000	$2-3h^1$	$2s$	
1000000	$9-10h^1$	$2s$	

Table 3.2: Maximal memory consumption comparison

4. Conclusion

We've presented an implementation of fast string alignment as described in [3]. Using the Four Russians algorithm combined with the proposed edit matrix reduction system we've achieved a significant improvement over the quadratic string alignment algorithm. In terms of time complexity, our implementation is faster by a factor of $log_{3\sigma}(N)$ where σ denotes the alphabet size. When discussing space complexity, we use less memory (by the same factor) but the required memory is still a bottleneck when dealing with long strings. Nevertheless, an improvement is visible - we can calculate the edit script for strings up to a million length if provided with a supercomputer, and that was not the case for the quadratic algorithm.

The method we explored leaves barely any room for space complexity optimization. With that in mind, the next potential improvement would be to research and implement a more memory-efficient idea, for example [2].

5. Bibliography

[1] Dan Gusfield. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge university press, 1997.

[2] Vamsi Kundeti i Sanguthevar Rajasekaran. Extending the four russian algorithm to compute the edit script in linear space. U *Computational Science–ICCS 2008*, stranice 893–902. Springer, 2008.

[3] William J Masek i Michael S Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System sciences*, 20(1):18–31, 1980.

[4] Saul B Needleman i Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.