

Aldabi

# Projekt 1 Dokumentation

Lucas Rieckert, Maximilian Otto, Johanna Eitel

Projekt Beginn: 1.11.17 Projekt Abgabe: 24.11.2017

## Abstract

**Motivation:** Unsere Motivation hinter dem Projekt, liegt darin den Origin of Replikation einer Dna Sequenz zu finden. Dies wollen wir durch Algorithmen lösen und daraufhin deren Laufzeit bestimmen.

**Results:** es geht um das finden der am häufigsten auftretenden k-mere in einem bestimmten Bereich (500bp) um den Replikationsursprung.

**Contact:** lackyluck@hotmail.de maxotto45@gmail.com eitel.johanna@yahoo.de

## 1 Introduction

Unsere Aufgabe war es drei Algorithmen zu implementieren, die sich zum finden eines OriC eignen. Die Wahl der Programmiersprache stand uns frei. Unsere Wahl fiel auf Java, einer der bekanntesten Programmiersprachen.

- FrequentWords
- FasterFrequentWords
- FindingFrequentWordsBySorting

Dies sind die Algorithmen, die in einem String nach den am häufigsten auftauchenden K-mere (Teilstrings der Länge k) durchsuchen und diese ausgeben. Dabei gehen sie unterschiedlich vor. Damit wollen wir herausfinden, welche extrem häufig auftreten und somit für den Organismus sehr wichtig sind. Wenn wir zusätzlich wissen, wo ungefähr der Replikationsursprung liegt, können wir diese in einem Feld um den OriC herum suchen. Mit unseren selbst geschriebenen Algorithmen sollten wir dann den OriC zweier Organismen

- *Thermotoga petrophila*
- *Vibrio Cholerae*

untersuchen.

## 2 Vorgehen

Die JAVA-Datei mit dem Code für alle 3 Algorithmen muss nur ausgeführt werden, (bspw. mit Eclipse) und dann wird der Input-String, sprich, die zu durchsuchende Sequenz gefordert. Im Anschluss soll man angeben, wie lang die k-mere sein sollen, die das Programm filtern soll. Das Ergebnis wird für jeden Algorithmus einzeln ausgegeben.

## 3 Conclusion

Zusammenfassung: Um den OriC von *Thermotoga Petrophila* ist als häufigstes 3-mer "aaa" und 9-mer "acctaccac" anzutreffen. Um den

Replikationsursprung von *Vibrio Cholerae* ist das häufigste 3-mer "aat" und die beiden am häufigsten vorkommenden 9-mere "tatgtgggt" und "gaattcagt".

Im Folgenden nun die Laufzeiten der einzelnen Algorithmen abhängig der Eingabesequenz, dargestellt in Nanosekunden. Der FREQUENTWORDS-Algorithmus hat mit seiner enorm hohen Laufzeit (ca. 0,1s) einen eigenen Plot benötigt, um die anderen beiden Algorithmen nicht auf der X-Achse liegen zu lassen.



