

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 loesen und daraufhin deren Laufzeit bestimmen.

Results: es geht um das finden der am haeufigten auftretenden k-mere in einem bestimmten Bereich

(500bp) um den Replikationsursprung.

Contact: lackyluck@hotmail.de maxotto45@gmail.comeitel.johanna@yahoo.de

1 Introduction

Unsere Aufgabe war es drei Algorythmen 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
- $\bullet \quad Finding Frequent Words By Sorting \\$

Dies sind die Algorithmen, die in einem String nach den am haeufigsten auftauchenden K-mere (Teilstrings der Laenge k) durchsuchen und diese Ausgeben. Dabei gehen sie unterschiedlich vor. Damit wollen wir Genabschnitte finden die extrem haeufig auftreten und somit fuer den Organismus sehr wichtig sind. Wenn wir zusĤtzlich wissen wo ungefĤhr der Replikationsursprung liegt, kĶnnen 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

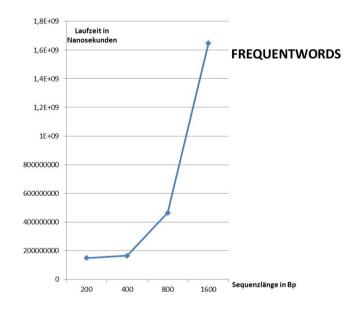
Die JAVA-Datei mit dem Code f \tilde{A} 1/4r alle 3 algorithmen muss nur ausgef \tilde{A} 1/4hrt werden, (bspw. mit Eclipse) und dann wird der Input-String, sprich, die zu durchsuchende Sequenz gefordert. Im anschluss soll man angeben, wie lang die k-mere sein sollen, die das Programm filtern soll. Das ergebnis wird f \tilde{A} 1/4r jeden ALgorithmus einzeln ausgegeben.

3 Conclusion

Zusammenfassung: Um den OriC von Hermotoga Petrophila ist als haeufigstes 3-mer "aaa" und 9-mer "acctaccac" anzutreffen. Um den

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

Im Folgenden nun die Laufzeiten der einzelnen Algorithmen abh \tilde{A} \bowtie ngig der Eingabesequenz, dargestellt in Nanosekunden. Der FREQUENTWORDS-Algorithmus hat mit seiner enorm hohen Laufzeit (ca. 0,1s) einen eigenen Plot ben \tilde{A} \parallel tigt, um die anderen beiden Algorithmen nicht auf der X-Achse liegen zu lassen.



© The Author 2015. Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com

2 Sample et al.

