Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: 12 Januar 2018
Manuscript Category

OXFORD

Algorythmische Bioinformatik

Projekt 2 Dokumentation

Lucas Rieckert, Maximilian Otto, Johanna Eitel

Projekt Beginn: 4.12.17 Projekt Abgabe: 12.1.2018

Abstract

Motivation: Unsere Motivation hinter dem Projekt, liegt darin einen Genom-Assembler basierend auf einem DeBrujin Graphen zu implementieren.
Results: Eine Fasta Datei, mit allen generierten Contigs.
Contact: lackyluck@hotmail.de maxotto45@gmail.comeitel.johanna@yahoo.de

1 Introduction

Unsere Aufgabe war es mit diesem Graphen alle Contigs der reads des Genoms von Nasuia deltocephalinicola zu berechnen. Die Wahl der Programmiersprache stand uns frei. Unsere Wahl fiel auf Java, einer der fuer uns bekanntesten Programmiersprachen.

Dies funktioniert in dem wir aus den gegebenen Reads einen DeBrujin Graphen aufbauen, und uns mittels des MaximalNonBranchingPath Algorythmus die laengsten unverzweigten Wege ausgeben lassen. Diese bilden dann die Contigs.

2 Vorgehen

Zunaechst haben wir jede Zeile des Datensatzes, die je einen Read enthÃ¤llt, mittel eines FileReaders in unser Programm eingelesen Da der uns zur Verfuegung gestellter Datensatz von reads der Laenge 100 allerdings keine vollstaendige 100er komposition des Genoms von Nasuia deltocephalinicola dartesllt, haben wir jeden gegebenen Read in all seine Moeglichen k-mere zwerlegt um eine bessere Abdeckung des Genoms zu erziehlen. K ist dabei Variable. Je kleiner k desto besser die Abdeckung aber auch umso komplexer wird auch der Graph und umso hoeher wird die Laufzeit. Wir haben uns nach mehreren Versuchen fÃ¼r einen k-Wert von 95 entschieden. Anhand dieser neuen k-mere haben wir dann den DeBrujin-Graphen aufgebaut.Indem wir aus allen reads Kanten (= kmere der reads) gebildet haben.Dann haben wir diese alphabetisch sortiert und die zugehoerigen Startknoten aus den k-1meren (Die Praefixe der gebrochenen reads) gebildet. Nach dem naechsten Schritt haben wir zwei Listen: Eine mit allen Startknoten und eine, in der an gleicher Stelle alle zugehoerigen Kanten stehen. Durch diesen Graphen sind wir dann mit dem MaximalNonBranchingPath Algorythmus durchgelaufen um die Contigs zu generieren. Jeder gefundene Pfad stellt ein Contig dar. Diese haben wir dann mittels eines FileWriters in eine Fasta Datein geschrieben, getrennt durch die Id-Lines, die die Nummer des Contigs angeben.

3 Conclusion

Zusammenfassung: Bei einem K von 95 laeuft es ca 119 Sekunden, allerdings steigt die Laufzeit bei einem K wert von 85 schon auf 7 Minuten. Da wir Probleme mit dem ausfuehren von Spades hatten koennen wir leider keine Annahme ueber den Nga50 Wert unseres Assemblers machen, da aber eine exorbitante Menge an Contigs erstellt wird kann dieser nicht alzu gut sein.