Computer Science for Mathematicians -Spring 19

Project: a genome assembly graph

The goal of this project is to expose you to a realistic problem on a real dataset that challenges you to make use of skills you have picked up in the course. How you use these skills will determine your grade.
The particular problem comes from genome assembly (see Problem background), where graphs are used to represent knowledge about segments of DNA, so-called *contigs* and their overlaps. Your will characterize and partition a graph you are given.

Be sure to read the section Requirements!

Overview

You are given a graph *G* where a node represent a segment of DNA and an edge means that two segments look like they overlap and should probably be merged into one longer segment. The real chromosomes from the genome that gave rise to the graph should, in principle, be embedded in *G* and correspond to a set of paths.
Unfortunately, the overlap detection is difficult because the original DNA data is (in some places) very repetitive. The effect is that there are *many* edges that are wrong, so many that the graph has become too large for some standard genome assembly tools.

It is to some extent possible to guess where the problems are. If the data had been "easy" (here: the DNA contains no repetitions), each node in our graph should have on average about 3-5 neighbors. The number neighbors is the outcome of a random experiment, so it is not surprising with up to 10 or 20 neighbors once in a while. In the data you get, there are however some nodes with far more neighbors than that.

It should be possible to simplify *G* by throwing away suspicious nodes and/or edges! But at the same time, we don't want to throw away too many because that might mean we are throwing away data.

Assignment

Your assignment is to advice and help an imaginary collaborator, whom can handle subgraphs of size up to 1000 nodes. To help your collaborator, partition the assembly graph into components by removing nodes. The specific tasks are:

- **Characterize the dataset**. Please report:
  - The node degree distribution of *G*.
  - The number of components of *G*.
  - Component size distribution of *G*.
- **Suggest a partitioning of the graph.**
  - Develop Java code to partition the data into graph components.
  - Each component is a connected subgraph with no more than 1000 nodes.
  - How many nodes and edges do you remove? Fewer is better!

Groups

You are expected to work in groups of 3 to 5 persons. The assignment can certainly be solved by one person alone, but a part of the project is to collaborate and practice writing software together. Every team member is expected to participate in developing code and scripts for the project.
Send a link to the group's github repository to Lasse as soon as possible.

Requirements

You are expected to hand in:

1. A report describing your work and your results. Write the report as if describing and documenting for your imaginary collaborator what you have done.
2. A partitioned graph.
3. A github repository with code (Java code and shell scripts) used in the project. *This repository is used to determine project participation!*
4. A *lab notebook* for each team member, noting what you have done each project day. The real purpose of the lab notebook is not about showing a teacher what you have done, and it is not a diary of how fun the project was, but a reminder to your future self what you actually did. It can also be used as a communication tool within a team. Here, it becomes a means of involving the teacher.

There is a great paper by William Stafford Noble with good advice for projects like this (that apply to many other disciplines than Computational Biology!): A Quick Guide to Organizing Computational Biology Projects (PLoS Comp Biol, 2009).

Grading

You have to solve the problem using basic software engineering principles and write a report on it to pass and get grade E. For a higher grade, demonstrate more skills. Each demonstrated skill increases the grade one step:

- Demonstrated use of Unix tools in the project.
  - Your report should give examples of how you have used Unix tools (like in Lab 1).
- The team has used a shared repository at `git` and `github.com` in a skilled way.
  - Share your repository with me (user: arvestad at github.com). I want to be able to see your commit history to determine proper use of git and github.com. No git commmits means an F for the project.
  - Describe in your report how you have used git and github.com
- Demonstrated application of algorithm techniques from the course.
  - Describe in your report the algorithms you have used to solve the problem.
- Demonstrated ability to discuss and reflect on algorithm characteristics, for example time/space complexity and applicability of an algorithm, on a real-life dataset.
  - Describe in your report the characteristics of the algorithms you have considered, why you choose them, their potential limitations, and experiences using them.

Data

The data is overlap information for a set of contigs (n=11 393 435) built using several tools on several datasets. Contig overlaps have been computed already (using `dfp-overlap`) and is available as a 7 GB textfile (1.6 GB compressed).
The data file contains the following columns:

1. Identifier for one contig in overlap.
2. Identifier for second contig in overlap.
3. A number representing similarity, but not relevant for this project.
4. Fraction of identical positions in the overlap.
5. A zero. Unused.
6. Start of overlap in first contig.
7. End of overlap in first contig.
8. Length of first contig.

9. A zero (0) or one (1) to say whether the overlap is on the reverse sequence (1) or not (0). See below for how this is used.
10. Start of overlap in second contig.
11. End of overlap in second contig.
12. Length of second contig.

This data is implicitly defining a graph. *Remember:* each contig is represented by a node and an overlap yields an edge.
Sometimes, the overlap is actually *containment*: if contig A is a subsequence of contig B, then A can be discarded.

The redundant columns 3 and 5 are there to follow a *de facto* standard for reporting sequence similarity.

Column 9 indicates a direction, but this can also be ignored.

Hints

- Using the contig identifiers as vertex identifiers is a waste of RAM memory!
- Be sure to create small test files to test your code on. Doing the first experiments on the full graph is a waste of time.

Problem Background

You can work on the project without reading the following description, but it might help you get a context for the assignment.

Your computational problem stems from the Spruce genome assembly project which aims at inferring the genome of *Norway spruce*, which is the tree you find all over Sweden. This genome is large (about 20 Gbp long, compared to 3 Gbp for human and most vertebrates), contains 12 chromosomes of roughly equal size, and is highly repetitive. The high degree of repetition and low complexity regions makes assembly very difficult and the current ambition is therefore to primarily assemble the gene-containing regions. The genes seems to be fairly distributed over the genome.

Using several sequencing techniques and several assembly methods, we now have a large set of contigs which we would like to combine. The contigs range in size from 1 kbp to 20 kbp and have a large number of overlaps. There is today no software to combine the overlapping contigs in a reasonable time (we have tried!) so we want to break up the problem in smaller pieces. Due to the low-complexity regions, there is a large number of *false* overlaps, i.e., overlaps that are due to the same subsequences appearing in many places in the genome, not because two reads were sampled from approximately the same position.

Spruce is a quite heterozygous diploid species. About 1% of positions differ in any two individuals (comparable to the difference between human and chimp), and affects the sequencing data we have. In this genome project, there is no information on the sequence level from which chromosome a contig comes from and there are therefore many contigs that have significant overlap, but with some differences because they come from two different versions of the same chromosome.

Senast modifierad: måndag, 20 maj 2019, 13:50

◄ Nyhetsforum

Hoppa till...

Extentor ►

NAVIGATION

Hem

Mitt hem

Mina kurser
Vetenskaplighet VT19
MT5002 VT19
MT5009 VT19
MT5012 VT19
MT6001 VT19
DA3018 VT19
Deltagare
Märken
Kompetenser
Betyg
General
Project: a genome assembly graph
Kandidatprogram i matematik
MM2001 - ht16
Anmälning etenta Matematik I - ht16
Handledning - ht16
Mer...

INSTÄLLNINGAR

Administration av kurs