Inlämningsuppgift

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.

You will characterize a graph you are given. 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. Note that you actually do not need understand the underlying biology at all to complete this project.

Be sure to read the section Requirements!

Overview

You are given a graph *G=(V, E)* where a vertex represents a segment of DNA. If two DNA segments have long and significant similarity at their ends, then we create an edge between their corresponding vertices. So, if $v, w \in V$ and $(v, w) \in E$, then the DNA contigs represented by *v* and *w* are *believed* to come from overlapping regions in the genome. In *genome assembly*, this means that it might be a good idea to merge the contigs that *v* and *w* represent into a larger contig.

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. The graph would be used to *assemble* the genome from parts.

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 standard genome assembly tools. So what is wrong with the data and what can we do to fix it? That question is too difficult, but a first step is to understand the graph better. If it was a small data set we could try visually inspect the graph, but the scale of this data makes that impossible. What we can do, however, is to look at features of the graph.

Assignment

Your assignment is to advice and help an imaginary collaborator who wants to better understand the dataset.

Please report:

- The node degree distribution of *G*.
- The number of components of *G*.
- Component size distribution of *G*.

The distributions can be summarised using tables or histograms. Do not bother trying to convey the precise observed distribution.

Apply the skills you have learned in the course. In particular, use shell commands and/or scripts, use git and GitHub (for storing code, scripts, notes, etc), implement suitable algorithms and reflect on their suitability for solving problems.

Requirements

1. Use a *private* GitHub repository for your code (Java code and shell scripts) used in the project *and* your report plus notebook (see below). Use this repository as if you are expecting to get collaborators in your project!
2. During your project work, keep a *lab notebook* in your repository noting what you have done each project day. The purpose of the lab notebook is to document the *process* of your work. See *Hints* (below) for comments on the lab notebook.
3. Write a brief report describing your work *and* your results as a file in your repository. Write the report as if describing and documenting for your imaginary collaborator what you have done.
4. Report and lab notebook can either be a Markdown file (.md) or a regular text file (.txt).
    - **Not accepted:** Files in Word, Open/Libre Office, Pages, PDF, or similar format.
    - Markdown format is preferred. Give it a try! It is a convenient and useful format that is very easy to learn.

Submission

Share your repository with Lars (user: arvestad at github.com), and be ready to share with a TA as well, because they will be involved in the grading.

Grading

The project is graded *Pass/Fail*. To pass, you have to earn 5 points, and points are given as follows:

- Solving the computational problems:
    - On the full graph: 2 points.
    - On a significantly reduced graph: 1 point
- You have used a git repository at github.com in a skilled way.
    - This means that we can see several git commits and find code, lab notebook, and report easily accessible in the repository.
    - Score: 1 point.
- Demonstrated application of algorithm techniques from the course.
    - Describe in your report the algorithms you have used to solve the problem.
    - Score: 1 point
- Demonstrated ability to discuss and reflect on algorithm characteristics.

- Describe in your report the characteristics of the algorithms you have considered, why you choose them, their potential limitations, and experiences using them.
- For example time/space complexity and applicability of an algorithm, on a real-life dataset.
- Score: 1 point

- A useful lab notebook.

    - A teacher can find your notebook, read it, and be informed about your work.
    - Score: 1 point

Partial points may be rewarded.

Please note that the length of your report does *not* have impact on your grade.

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 first vertex for an edge.
2. Identifier for second vertex of the edge.
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. Columns 1 and 2 are vertex identifiers, so each line is a *potential* edge. Columns 6-8 and 10-12 describes the overlap. The overlap is actually *containment* if contig A is a subsequence of contig B — then A should be discarded because it is redundant.

For example, if columns 6, 7, and 8 are indicating the whole contig is involved in the overlap, then this line can be ignored: it should not define an edge.

You can ignore columns 3, 4, 5, and 9. Column 4 is not important in this project, and columns 3, 5, and 9 are only present to follow a *de facto* standard for reporting overlap graphs.

Hints

- Use Unix tools to your advantage. This data file is too large to open in an editor, but just fine to apply Unix tools on.
- Using the contig identifiers (columns 1 and 2) as vertex identifiers is a waste of RAM memory, because they are all very long. You need to create a translation into integer identifiers for vertices. Store those integer identifiers as integers, not as strings. An integer uses 4 bytes on a modern computer. A string containing four digits uses more than that (I have seen claims that an empty string uses 24 bytes).
- Be sure to create small test files to test your code on. Doing your first experiments on the full graph is a waste of time.
- Lab notebooks are very useful when working with data in general. The real purpose are not about showing a teacher what you have done or keeping 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. In this project, the lab notebook is a means of involving the teachers.
- 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).
- Consider having a `doc` folder in the repository where you keep `report.md` and `notebook.md`. One can also store the notebook as a subfolder with one file per day. For example, the file `doc/notebook/2021-05-27.md` would describe your progress on May 27.