Suffix Structures for DNA Sequencing

Final Project Draft

Guillaume Laliberté

CSCI E-51: Visualization

Prof. Greg Morrisett

CS51 Advisor Thomas Jiang

April 2015

Draft Specification

1. Overview and Goals

While DNA sequence used to be costly to achieve and required significant resources, it is now becoming more accessible everyday. One reason for this is that the quantity of data available is growing exponentially, and DNA databases make available that data to researchers. One challenge remains: processing DNA and comparing the (very long) strings of A,T,C and G can be computationally demanding.

One way to improve the way DNA sequences are compared is to use data structures adapted to this task. The suffix arrays of suffix trees are a very good candidate when is comes down to comparing strings.

With this project, **my goal is to create an Ocaml program that implements said data structure and allows the user to compare strings of DNA**.

2. Feature List

In order to be successful, the following features should be implemented in priority:

1. **Implementation of the suffix tree/array (the absolute minimal feature)**
2. A terminal command to load DNA files to compare
3. A comparator that uses suffix tree/array to compare the files
4. A simple output of the results (statistics like % similarity, frequent sequences, etc.)

The following features, in order of priority, would be really interesting extensions:

5. Additional metrics for DNA comparison (ATCG composition, particular genes i.e. cancer genes, precise know and interesting patterns)
6. Graphical way to represent the DNA similarities
7. A queue to automate comparisons (multiple files)
8. A web interface to control the program and visualize the output
9. Compare suffix tree and suffix array

3. Technical Specification

3.1 Core

The core of the program will be a `DNASuffix` module implemented with a suffix tree or array.

The `DNASuffix` is a sequence `sequence` of elements of type `base`. The following items must be implemented in order for the program to work. Additional items will be included in the final *Technical Specification*. More research on the suffix structures and DNA metrics is required at this point.

```
is_empty : sequence -> bool

insert : base -> sequence -> sequence

member : sequence -> sequence -> sequence

string_of_sequence : sequence -> string

blast : sequence -> sequence ->
      (percentage*base_frequencies*other_metrics)
```

The `SuffixArray` or `SuffixTree` will have similar functions we can use to implement the DNA sequence as and array or tree.

*Note that `array` may be replace by `tree`. The decision for the first implementation of `DNASuffix` still has to be made.

```
is_empty : array -> bool

insert : elt -> array -> array

member : array -> array -> array

string_of_array : array -> string
```

3.2 Comparison

The comparison can be completed exclusively in the `blast` function of `DNASuffixTree`, since going through the sequences multiple times is undesirable.

3.3 Input/Output

- A way to take as input two files, save them as suffix structures.
- Process the files and use `DNASuffix` to index the sequence
- A module to output relevant DNA data

4. Next Steps

Here are the steps that must be completed before writing the final *Technical Specification*:

- Choose between suffix trees and suffix arrays (which one to implement first)
- Select which metrics to compute first
- Find an efficient way to compute the metrics
- Setup the repository and initialize the project
- Finalize the project structure (files and modules)
- Familiarize myself with the chosen suffix structures
- Finalize the *Technical Specification*

5. Project Schedule

Proposed schedule for the first week:

Date	Goal	Turn in
Apr 13, 2015	Select primary metrics	
Apr 14, 2015	Selection of suffix trees or arrays	
Apr 15, 2015	Proposed modules and functors	
Apr 16, 2015	Proposed project structure	
Apr 17, 2015	Final Technical Specification	**Technical Specification**
Apr 24, 2015	Work in progress, Suffix structure working	**Functionality Checkpoint**
May 1, 2015	Cleaned up code, fully-working project	**Finish Project**