Retrieval and mapping

1. Introduction

In the previous topic, you were introduced to methods for aligning two sequences based on dynamic programming. When you want to search for similar sequences in a large database, it is computationally impossible to calculate the exact alignment between the query and all sequences in the database. Smarter and faster methods are necessary. The same “need for speed” holds for studies to detect genomic variation, to identify differentially expressed genes, or to map DNA-protein interactions, which require the mapping of huge amounts of (short) genomic reads to a reference genome. So, both retrieving sequences from large databases and mapping sequences to large genomes require fast solutions.

The general solution to this problem is “indexing”: a technique to quickly find possible high-scoring short local alignments, which can then be extended into final alignments (and scores). The downside of these methods is that they do not guarantee to find the optimal solution. In this topic of the course you will study different indexing techniques, relevant for both database searching and read mapping. You will read about some widely used methods, BLAST and FASTA, and implement an algorithm called SSAHA.

A common data structure, used by many read-mapping tools, is a hash table, containing short oligomers from either the query/reads or the reference. Alternative (and sometimes more sophisticated) data structures that we will discuss are: suffix tries/trees/arrays, the Burrows-Wheeler index, and the FM-index.

Reading material

Section 4.6 and 4.7 of “Understanding bioinformatics” address database searching with nucleic acid and protein sequences. Section 5.3 focuses on database-indexing techniques and section 5.4 is about the significance of optimal alignments between a query and a database. In the assignment you will implement the SSAHA algorithm, described in: Ning *et al.*, “SSAHA: a fast search method for large DNA databases”. *Genome Research* (2001) 11(10):1725-9.

2. Assignment

In this assignment you will implement (parts of) the SSAHA algorithm, a fast search method for large DNA databases. The algorithm is described in Ning *et al.*, “SSAHA: a fast search method for large DNA databases”. *Genome Research* (2001) 11(10):1725-9. You will use the data from the paper and the example from Table 1 for development of your algorithms, and later apply it to search the Arabidopsis genome.

Use the code skeleton in `assignment2_skeleton.py` to:

- implement a function to construct a hash table of non-overlapping *k*-mers (Table 1 in the paper). The “word size” *k* should be a parameter of the function. You may leave out the *k*-mers that do not occur in the dataset. You also don’t have to calculate E(w).
- implement a function to iterate over all *k*-mers in a query sequence and to return a list of hits, sorted by *index* and *shift* (column *M* in Table 2 in the paper).
- implement a function to return the longest match between the query and the database. Focus on exact matches only, so same index and shift, different offset, as is highlighted in bold in Table 2.
- implement a function to print the alignment of the query and the subject of the match (so, an alignment of two sequences, ungapped).

Questions:

1. Build a hash table for the provided list of sequences “seqs” (same as in the paper). Print the table.

2. Calculate the sorted list of hits for the provided query sequence. Each hit should be a tuple of *(index, shift, offset)*. Print the number of hits and the first and last hit in your list.

3. Calculate the maximum match for the provided query sequence, and print the alignment (the two sequences with the right offset, such that the characters match). For example:

   ```
      GACGG
      .||||
   ATTCACGGCTA
   ```

4. DATA: http://www.bioinformatics.nl/courses/BIF-31306/TAIR10.fasta
   Write a fasta parser to read in the *Arabidopsis thaliana* genome. How many sequences does the genome file contain and what is the total sequence length? Does this match your expectation on the Arabidopis genome?
   NOTE: parsing the fasta file should only take a few seconds, otherwise you are doing something very inefficient.

5. Build a hash table for the Arabidopis chromosomes. Use *k* = 15. How many keys (different 15-mers) does your hash contain?

6. DATA: http://www.bioinformatics.nl/courses/BIF-31306/athal_query.fasta
   For the query sequences in the FASTA file **athal_query.fasta**, find the maximum hit(s) in the genome. Report your findings (if any).

7. What is the algorithm trying to optimize?

8. What did you learn about the time complexity of the search part of the algorithm?

9. What is the main difference in time complexity of the SSAHA algorithm and BLAST?

10. How many *k*-mers with an overlap of *p* bases does a reference sequence of length *m* contain?

Optional:
Can you implement matches in the reverse direction? Now, can you reproduce the output from SSAHA2 for the query sequences? Why or why not?

SSAHA2 reports the following matches for these query sequences:

```
===================================================
Matches For Query 0 (444 bases):
===================================================
Score         Q_Name S_Name   Q_Start Q_End  S_Start    S_End    Direction #Bases identity
ALIGNMENT::50 439    Chr3     1       444    160638     161081   F         444    100.00 444

===================================================
Matches For Query 1 (190 bases):
===================================================
Score         Q_Name S_Name   Q_Start Q_End  S_Start    S_End    Direction #Bases identity
ALIGNMENT::50 171    Chr4     190     1      43142      43331    C         190 100.00 190

===================================================
Matches For Query 2 (621 bases):
===================================================
Score         Q_Name S_Name   Q_Start Q_End  S_Start    S_End    Direction #Bases identity
ALIGNMENT::50 620    Chr4     621     1      2726910    2727530  C         621 100.00 621
```

Hints

- You can sort a list in python, simply by calling the method sort on it, e.g. `my_list.sort()`
- The enumerate function in python can be used to iterate over a tuple of (index, item) for items in a list. For example:

```
for seq_idx, seq in enumerate(seqs):
    print seq_idx, seq
```

Please, email your Python script and a PDF file containing the answers to the questions to algorithms@bioinformatics.nl no later than 14.00 on the day of the lecture.

3. Further reading

As searching for sequence similarity is essential to answer many (evolutionary) biological questions, (database) indexing is an important part of many computational methods in bioinformatics. SSAHA uses a hash table as index data structure, but several other data structures are frequently used: suffix trees, suffix arrays, or the Burrows-Wheeler index. The material for further reading for this topic focuses on some of these alternative indexing techniques:

- Whole genome alignment (Delcher, *NAR* 1999)
- Short read mapping (Langmead, *Genome Biology* 2009)
- ...

You are asked to present one of these papers, focusing on the algorithmic challenges and implementation rather than the (biological) results.