CPSC 445: Project Proposal

Group: Bobby Dhillon, Arthur Lai
Title: Local Alignment of DNA Using BWT-SW
Website: Yes

Chosen Paper: T. W. Lam, W. K. Sung, S. L. Tam, C. K. Wong, S. M. Yiu. *Compressed indexing and local alignment of DNA*. Bioinformatics 24:6, 791-797 (2008) - **Local Alignment**

Project Type: Type 1 (take a paper, implement it and run it on some interesting data).

Paper Description: The paper by Lam *et al.* (2008) introduces an extension to the Smith-Waterman (SW) dynamic programming algorithm using a compression technique known as the Burrows-Wheeler transform (BWT). This technique is used to create a BWT index of a text T in order to optimize the dynamic programming for finding all local alignments given a pattern P. This speeds up the SW algorithm enough to allow feasibility of using extremely long DNA sequences (*e.g.*, the human genome).

The authors refer to their algorithm as BWT-SW. The expected running time of the algorithm is $O(n^{0.628}m)$, where $n = |T|$ and $m = |P|$. This is much better than the $O(nm)$ running time of SW, but still slower than BLAST. However, BLAST is a heuristic algorithm which does not always find every local alignment. Therefore, the algorithm produced by Lam *et al.* (2008) is important to ensure that alignments critical for biological research are not missed.

Implementation Plan: We will be implementing BWT-SW using Java. The progress of our implementation will be tracked using the tools offered by GitHub. Given the nature of implementing a local alignment algorithm, it is not possible to easily modularize the work. Hence both group members will work continuously on the code and push small implementational changes to the GitHub repository.

Repository URL: `https://github.com/watchstopper/cpsc-445-project`

The major steps to coding our implementation are:

1. Given a text T, construct a BWT index in the main memory.

    - Lam *et al.* (2008) implemented the construction algorithm of BWT described in Hon *et al.* (2007) so we will likely need to refer to that paper to complete this step.

2. **Phase I** as described in Lam *et al.* (2008): For all $1 \leq i \leq n$, $1 \leq j \leq m$, and $1 \leq h \leq i$, compute $A[h, i, j]$ which equals the largest alignment score of $T[h..i]$ and any substring of P ending at position j.

    - Use a pre-order traversal of the suffix trie of T to enumerate the substrings.

    - Compute, for each node u on a path of the suffix trie and for all $1 \leq j \leq m$, the best alignment score of the substring represented by u and any substring of P ending at j.

    - For each node in the suffix trie, when filling the dynamic programming tables, ignore all entries with values less than or equal to zero. For a node u, if there is a row with all entries in all dynamic programming tables with values less than or equal to zero, prune the whole subtree rooted at u.

3. **Phase II** as described in Lam *et al.* (2008): For all $1 \leq i \leq n$, $1 \leq j \leq m$, return the largest among all alignment scores $A[h, i, j]$ for different h.

Data Plan: Using the human genome, Lam *et al.* (2008) found that the miss percentage of BLAST increases in proportion to the evolutionary distance of the species from which the query sequence is obtained. In the analysis, four species were used: chimpanzee, mouse, chicken, and zebrafish. After we complete our implementation of BWT-SW, we will use query sequences from plants and bacteria to determine whether the pattern found by Lam *et al.* (2008) still holds.

The plant sequences will be obtained from PlantGDB:
`http://www.plantgdb.org/`

The bacterial sequences will be obtained from Ensembl Bacteria:
`http://bacteria.ensembl.org/`