# Homework 4 BWT, Suffix Trees and Suffix Array

陈斯杰 2016310721 自博16

Deadline: March 27.

1. Design a BWT data structure for the human genome (assuming 24 chromosomes totaling 3 billion basepairs). State exactly how many bytes are needed to store the data structure, and describe the expected # of operations to find an exact match of a pattern of 50 bases (considering both the forward strand and the reverse strand).

By shifting the text in a cyclic manner, we can get |Text| strings of different orders. Combine |Text| strings in rows to form a B-W matrix. The BWT-based algorithm stores the first column and the last column of B-W matrix; thus at least 2\*|Text| bytes are required.

For genome mapping problem, we shall do BWT to both the original genome sequence and its reverse complementary, i.e. 4\*|Text| bytes used in total. As for human genome, 4\*3\*109 bytes=1.2 GB of memory will be consumed in the algorithm.

For pattern matching problem, the exact string matching with FM-index takes O(|Pattern|) times. We could expect 50\*2=100 operations to find an exact match of 50 bases on both the forward strand and the reverse strand.

1. Design a suffix tree for the human genome, estimate the number of bytes that are needed to store the suffix tree.

Using starting position and ending position to represent tree edge, a suffix tree could be store with a total usage of Θ(|Text|); therefore we could expect a memory usage of 2\*3\*109 bytes=600MB.

1. Given *r* strings S1, S2, …, Sr, use the suffix tree to find the longest common substring among them.

Construct a generalized suffix tree and the longest common substrings are the maximally deep node.

1. Design an algorithm to construct a suffix tree from a suffix array in linear time.
2. Construct suffix array and LCP array. This can be done in linear time.
3. Start from the initialized root.
4. Add the first edge for the first suffix
5. For each next suffix, go up from the leaf until LCP with previous is below
6. Add a new edge for the new suffix.

**Reference**

1. Wikipedia *Generalized Suffix Tree* <https://en.wikipedia.org/wiki/Generalized_suffix_tree>
2. P3G Wiki *Longest Common Substring* <http://wcipeg.com/wiki/Longest_common_substring#Suffix_tree>
3. Coursera *Course Material* <https://www.coursera.org/learn/algorithms-on-strings/lecture/8QnH6/construct-suffix-tree-from-suffix-array-and-lcp-array>