CSCI 102 assignment 6 – BSTs for counting k-mers

Mar 17

In this assignment you will be implementing a binary search tree (BST) to count the occurrences of strings of length k in DNA. As an example, we may want to take the sequence `AGGAGAGG`, break it into sequences of length $k = 3$ – in biology we call these 3-mers – `AGG`, `GGA`, `GAG`, `AGA`, `GAG`, `AGG`, and count the occurrences of these strings in a map {AGG:2, GGA:1, GAG:2, AGA:1}. A classical option for implementing such a map is a hash table. Hopwever, sometimes we want k to be large and the DNA sequences we analyze can get very large; in these cases we can expect a large number of collisions, making hash tables inefficient. Instead, we can use BST to store entries with keys that are k-mers and values that are integers. Here you will implement a BST to store counts of k-mers in long DNA sequences.

To build a BST with keys that are `Strings` we will need a way to decide if one string is "greater or less than" another. To compare strings as such, we decide strings that are alphabetically later are "greater than". If you have two `Strings`, `string1` and `string2`, Java allows you to compare them alphabetically by writing `string1.compareTo(string2)`.

- Define a class `KmerCounter` which implements `Map<String, Integer>`. It will have private class `Node`. `Node` will have attributes `Integer count`, `Node parent`, `Node left`, `Node right`, `String kmer` with get and set methods for each. You can think of the node as storing an entry with key `kmer` and value `count`. `KmerCounter` will have two private attributes, the size `int size` and the root of the BST `Node root`. Feel free to reuse our BST code from class.

- Create a method `void printPositions` that prints the k-mers and their counts at each node by performing an in-order traversal.

- Create a method `int get(String kmer)` that returns the count corresponding to a k-mer stored in the tree (return 0 if the k-mer is not in the tree).

- Create a method `int addCount(String kmer)` that adds one to the count of the count of `kmer`. Use the regular BST rules (no need for AVL). If `kmer` is not in the BST create a new node for storing its count.

- Create a method `int remove(String kmer)` that removes and returns the count of `kmer`. If `kmer` is not in the BST return 0.

- Create a main method where you take the DNA sequence of human insulin below and build a `KmerCounter` counting all of its 4-mers; do this by looping through each position of the sequence and adding 1 to the count of the 4-mer at that position with the method `addCount`. Remove the 4-mer `TGAG` from your tree and run `printPositions`.

Insulin DNA sequence:

```
TGTTTCCATGCTCTGTGTACGTGCCGGACGAGTGGGAGGTGTCTCGAGAGAAGATCACCCTCCTTCGAGAGC
TGGGGCAGGGCTCCTTCGGCATGGTGTATGAGGGCAATGCCAGGGACATCATCAAGGGTGAGGCAGAGACCC
GCGTGGCGGTGAAGACGGTCAACGAGTCAGCCAGTCTCCGAGAGCGGATTGAGTTCCTCAATGAGGCCTCGG
TCATGAAGGGCTTCACCTGCCATCACGTGGTGCGCCTCCTGGGAGTGGTGTCCAAGGGCCAGCCCACGCTGG
TGGTGATGGAGCTGATGGCTCACGGAGACCTGAAGAGCTACCTCCGTTCTCTGCGGCCAGAGGCTGAG
```

Please submit your code and answers to the questions in a zipped folder on Brightspace by April 2.