Overview

In our progress to make a key-value store, our team was impacted by the pandemic. We've compiled the work that we have been able to complete here. Namely, that is a basic KV store using a proprietary dataframe, and elements of distribution including basic networking and serialization.

Data Structures

In order to implement this application in C with classes, certain basic data structures had to be recreated. The first of these is a basic resizeable array. Our resizing factor is 2 so when the allotted capacity is reached, it doubles in size.

Because there was a requirement for our arrays not to copy the actual values that they stored, we created an ArrayList-like class called ArrayArray. This class starts with one list of Arrays (we used our own implementation for ease-of-use functions but the underlying arrays never resize). Whenever a second-level array is filled, the first-level array adds another second-level array of elements to itself. This means its only pointers to first-level arrays that ever get copied.

A column is a special type of Array that is used in dataframes. The only significant difference about it is that keeps track of a type, and it also has a convenience constructor that uses variable arguments.

Next, we have our own implementation of a hashmap that uses the linear-probe method to get and put elements. It uses a tombstone flag to keep track of elements that have been removed to keep the runtime complexity O(1). It has a load factor of .5.

Dataframe

Our dataframe uses a couple of special classes including Row, Column, Schema, Rower, and Fielder. A Dataframe has a schema to keep track of what type each of its columns are. When a column is added to a Dataframe, the Schema is appropriately updated. There is a little bit of complexity with our schema because we have an enum of each data type but we also use characters in our Schema class. This is something that may be factored in future revisions.

The dataframe also has classes for iterating the Dataframe including rower and fielder. These classes are meant to be used in conjunction. The Rower interface is implemented and given a Row. Inside the Rower's accept method, a fielder is instantiated and accepts each element in the row. Together, these classes can be used to manipulate the data in the dataframe in a variety of ways.

File Reading

The 'cli' folder is a misnomer. It does not have a command line interface, but it can be used to read SOR files into our Dataframe. We adapted 4500ne's implementation of the sorer program in C to come up with this implementation. Because both of our implementations were sufficiently abstract, the connection required only a few small changes.

Serialization

The serialization in our project uses two separate classes: Serial and Deserial. Serial is used to serialize objects into a string buffer. The only objects it really serializes is primitives. All other datastructures have methods inside Serial, but they end up breaking down those objects into their constituent parts. For example, a simple IntArray is serialized into a code indicating the object type, the size, and then each of the individual ints. The Deserial works symmetrically. It first reads a code declaring what kind of object follows and then puts the object back together.