ECSE 4961 Proj 4

Ryan Walton

March 25, 2022

1 Objective

The purpose of this project was to implement a dictionary encoder for strings of varying length and create a data structure to aid in serving queries regarding strings located in the data set.

2 Building and Running

Run `g++ main.cpp -O3` and then `./a.out` to run.

3 Code Structure

The strings found in the provided text files were mapped unique integer IDs of incremental value. For example, the first string would be assigned to value 0, the next unique string to 1, etc. The strings are mapped to these unique IDs by a hash-map. The mapping from ID to string is done by indexing, where there is an array of strings with index corresponding to the ID. Since the average storage size of the ID is smaller than that of the strings and there are many repeats of each string, we can save space by including a mapping table and replacing the strings with their IDs.

With this basic functionality determined, 5 main procedures were implemented to manipulate data sets of this type.

3.1 `load_from_raw`

This loads data from a file including raw column string data into C++ objects.

3.2 `write_encoded`

This stores data in C++ objects into a file containing a mapping table and column data, substituting raw 32-bit integers for the original strings.

3.3 `load_from_encoded`

This loads data from a file including a mapping table and substituted column data into C++ objects.

3.4 `write_decoded`

This stores data in C++ objects into a file in raw column data form.

3.5 `query`

Performs a query on the loaded data, returning how many instances of the string are in the data set. This operation uses a hash map so the time complexity is independent of sample size.

4 Results

Below show typical results from the trials that were run, for time to encode (and write back encoded data) and also for time to perform a query. It would appear that the first query performed takes longer than subsequent ones, so typical values for both are recorded.

	Small	Medium	Large
encode	9.99ms	108ms	4.55s
query (first)	15.24us	12.23us	13.54us
query (subsequent)	2.09us	2.15us	2.35us

5 Conclusion

The data structure and encoding algorithm were implemented successfully and the performance of the required operations was acceptable. Multi-threading could be implemented in the future to speed up the performance.