Digital Wave Finance – Round 1

Answers and Reasoning by Alessandro Biella

Part 1

As commonly known searching, inserting, and removing elements from a hash set with linear probing takes constant time, if the hash function is of high-enough quality (std::hash<>()) and the load of the hash table is a constant strictly less than one. The calculation of the Chernoff bound is omitted, as this is a well-known task and easily looked up on the web.

This leaves us with the non-trivial task of implementing get_first() and get_last() in constant runtime. As there have not been any memory bounds given, an additional doubly-linked list was used. It seems worth to mention, that memory usage is still in O(n), as this list is maximally half the size of the hash table. This is caused by the implementation of the tables-load balancing.
Each key-value pair subscribed to this list by order of insertion. This custom linked list now simply needs to keep track of it's head and tail elements to execute get_first() and get_last() in O(1). Maintenance of this list is also in O(1), as nodes may remove themselves from it without searching through the list, whenever a key-value pair storing it is freed.

Part 2

Sadly, I haven't been able to provide a working solution to this exercise. I've been caught up with installing a library (first winsocked and openssl, then curl) to communicate with binance's API and underestimated the timely effort this caused. Thus, I did not actually receive the necessary trade stream to write the parser. Nonetheless I'd like to walk through the steps in theory: From the documentation I've taken, that the parser would be a reverse-engineered version of the actual implementation of binance when calling GET /fapi/v1/aggTrades. Optimally, a dynamic programming (dp) algorithm should be able to convert each trade into an aggregation of the specified format in linear time, reading each entry of each trade exactly once. Printing these aggregations is then trivial, by adding line-breaks and indents to the dp-array.

Conclusion

The exercises provided were very interesting on a theoretical level and I'd gladly learn more about their real-world implementations. I sincerely hope that I was able to show that I am eager and interested in the areas were I'd need to be properly instructed.