University of
Zurich^UZH

Department of Informatics

University of Zürich
Department of Informatics
Binzmühlestr. 14
CH-8050 Zürich
Phone. +41 44 635 43 11
Fax +41 44 635 68 09
www.ifi.uzh.ch/dbtg

UZH, Dept. of Informatics, Binzmühlestr. 14, CH-8050 Zürich

Prof. Dr. Michael Böhlen
Professor
Phone +41 44 635 43 33
Fax +41 44 635 68 09
boehlen@ifi.uzh.ch

Zürich, 7. März 2016

BSc Thesis: Implementing an Index Structure for Streaming Time Series Data

A streaming time series s is a sequence of data points that is extended continuously (e.g. every 10 minutes a new value arrives). Such data appears in many applications, e.g., the financial stock market, meteorology, sensor networks and network monitoring to name only a few. Since streaming time series are by definition unbounded, a system can only keep a portion of a time series in main memory. Let $W = [\underline{t}, \bar{t}\,]$ be a time window of length $|W|$, where time $\bar{t}$ represents the most recent time point for which the stream produced a new value and $\underline{t}$ represents the oldest time point that still fits into the sliding window.

The system needs to be able to efficiently perform the following operations on s in window W:

- shift($\bar{t}, v$): add value v for the new current time point $\bar{t}$ and remove value v' for the time point $\underline{t} - 1$ that just dropped out of time window W.
- lookup(t): return the value of time series s at time t, denoted by $s(t)$.
- neighbor($v, \mathbf{T}$): given a value v and a set of time points $\mathbf{T}$, return the time point $t \notin \mathbf{T}$ such that $|v - s(t)|$ is minimal.

To efficiently implement the above mentioned operations the system combines two data structures: a circular array and a B^+ tree, as explained in [2] and shown in Figure 1. The lookup operation can be efficiently handled by the circular array, while the neighbor operation exploits the fact that the leaves of a B^+ tree are interconnected and sorted. Notice that unlike normal B^+ trees, this variant of the B^+ tree maintains the leaves interconnected in both directions. Moreover, the B^+ tree needs to be able to deal with duplicate values.

Tasks

- Read [2] to understand the requirements the data structure must satisfy.

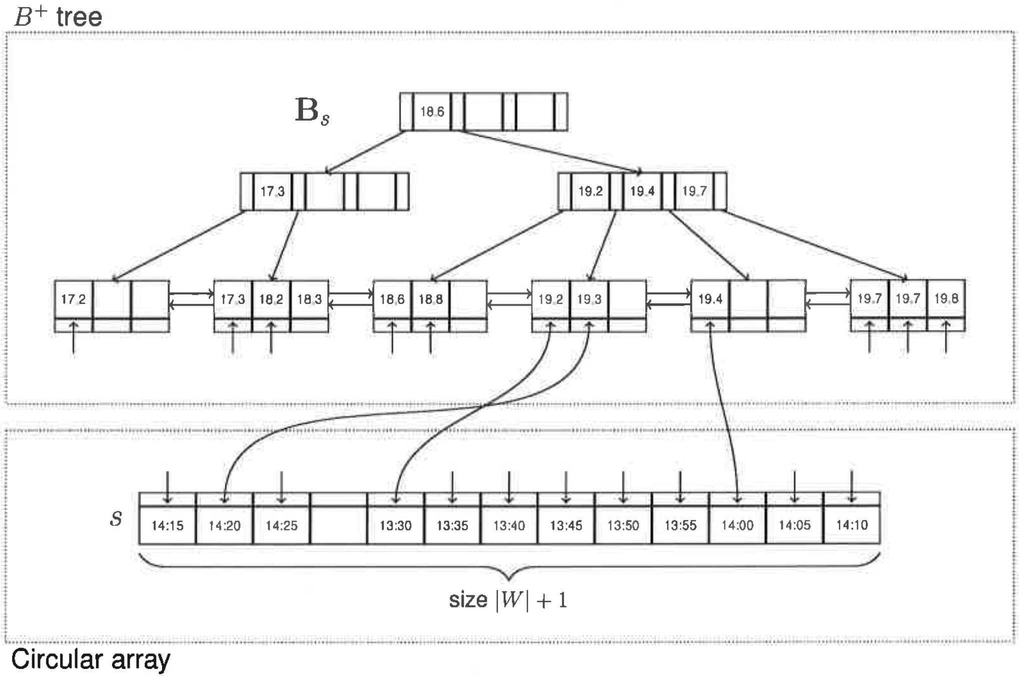

figure 1: time series s as circular array with B^+ tree index $\mathbf{B}_s$

- Implement the combined data structure (preferably in C).
- Analyze the runtime complexity of the operations on the data structure.
- Analyze the space complexity of the data structure.
- Perform experiments to underpin your theoretical results.
- Summarize your work in a detailed report.

Optional Task

- Integrate the data structure into the TKCM algorithm [2].
- Make the B^+ tree cache-concious [1].

References

[1] J. Rao and K. A. Ross. Making b+- trees cache conscious in main memory. In *SIGMOD*, 2000.

[2] K. Wellenzohn, M. Böhlen, A. Dignös, J. Gamper, and H. Mitterer. Continuous imputation of missing values in highly correlated streams of time series data. Unpublished, 2016.

Supervisor: Kevin Wellenzohn (wellenzohn@ifi.uzh.ch)

University of Zurich
Department of Informatics

Prof. Dr. Michael Böhlen
Professor