CAS CS 561: Data Systems Architectures
Data-intensive Systems and Computing Lab
Department of Computer Science
College of Arts and Sciences, Boston University
http://bu-disc.github.io/CS561/

CS561 Spring 2026 - Research Project

Title: *Joins with Near-Sorted Data*

Background: Modern day applications often generate ***near-sorted*** data with respect to some attribute. For example, near-sorted data is frequently collected by stock market applications, event-based applications like sensor readings, and data that have correlated columns. In fact, near-sorted data can also occur as a result of a previous join operation, or data that was recently sorted but received a few updates that occurred out-of-order.

Join algorithms are fundamental to database systems to answer complex queries with real-world data [1]. Sort-merge join (SMJ) is a prominent algorithm known for its high efficiency when the input relations are sorted on the join attribute. In such a scenario, the sorting phase of the algorithm is eliminated, and the algorithm only merges the two relations.

Objective: This project aims to explore the design space of join algorithms with near-sorted data. Particularly, the project targets to answer the following questions: *What if the input relations are instead, **near-sorted**? Does sort-merge join offer a viable option in exploiting near-sortedness? Can indexes that exploit data sortedness help SMJ become sortedness-aware?*

Technical: This project requires C++ programming skills (particularly working with pointers). The following steps outline the high-level milestones:

1. Develop a simple API that will use SMJ to sort two input data streams of integers.
2. Generate differently sorted data using the workload generator from the Benchmark on Data Sortedness (**BoDS**) [3].
3. Measure the runtime of SMJ with differently sorted data.
4. Use **QuIT** [2] and **SWARE** [4] to replace the ***sorting*** phase of SMJ and measure performance.
5. As an optional extension, you can also explore the design of near-sorted hash merge join.

References:

[1] Helmer, Sven, Till Westmann, and Guido Moerkotte. "Diag-Join: An opportunistic join algorithm for 1: N relationships." *None* (1997).

[2] Raman, Aneesh, et al. "QuIT your B+-tree for the Quick Insertion Tree." *In Proceedings of the International Conference on Extending Database Technology (EDBT)*. 2025.

[3] Raman, Aneesh, et al. "BoDS: A benchmark on data sortedness." *Technology Conference on Performance Evaluation and Benchmarking*. Cham: Springer Nature Switzerland, 2022.

[4] Raman, Aneesh, et al. "Indexing for near-sorted data." *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023.

CAS CS 561: Data Systems Architectures
Data-intensive Systems and Computing Lab
Department of Computer Science
College of Arts and Sciences, Boston University
http://bu-disc.github.io/CS561/