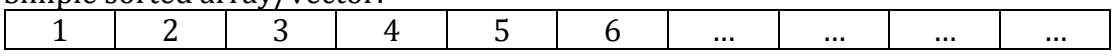
CAS CS 561: Data Systems Architectures
Data-intensive Systems and Computing Lab
Department of Computer Science
College of Arts and Sciences, Boston University
http://bu-disc.github.io/CS561/

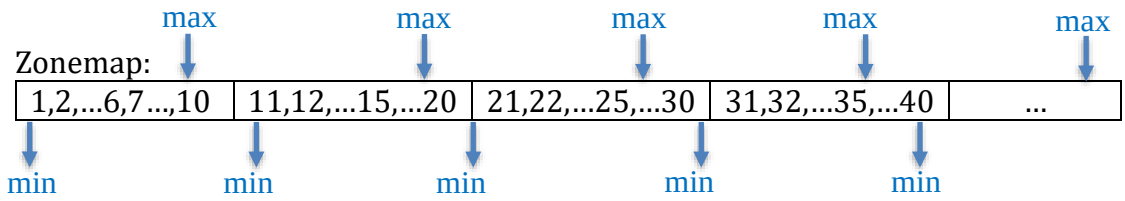
CS561 Spring 2024 – Project 0

Title: *Implementation of a Zone Map*

Background: A zone map is a coarse index that maintains minimum/maximum value ranges of one or more specified columns over contiguous sets of data blocks or rows, called zones of a table [1]. A zone map helps in data pruning of both single keys and a range of keys. The queried key/range of keys is first checked with the min/max values of every block/zone before searching within the block. By avoiding unnecessary I/Os to the storage, zone maps improve query performance.

Simple sorted array/vector:





Objective: The objective of the project is to implement a simple zone map and evaluate its performance on both point and range queries. For range queries, you will need to implement the query generator that can issue range queries with specific selectivities. The workflow for this is as the following.

(a) Implement a zone map by cloning the API available at: https://github.com/BU-DiSC/cs561_templatezonemaps2. This API contains a header file with basic functionality definitions for a zone map. You are free to modify certain components to improve performance.

(b) Implement the range query generator with selectivity $\boldsymbol{s}$ ($0 \leq s \leq 1$) as input in the existing workload generator (workload_generator.cpp). You can fix the number of elements to be selected as $\lfloor s * N \rfloor$ in your implementation. The existing workload generator generates three files: a raw data (to be input) file, a point query file, and an empty file. The empty file is supposed to store the generated range queries after you implement the range query generator.

(c) In main.cpp, write the code for a parser that parses the range query file. Report the average execution time for the following workloads.
W1: 5M inserts with sorted data*, 10K point queries
W2: 5M inserts with sorted data, 1K range queries, selectivity: 0.001
W3: 5M inserts with sorted data, 1K range queries, selectivity: 0.1
W4: W1 with unsorted data
W5: W2 with unsorted data
W6: W3 with unsorted data

*To generate sorted data just use the --sort flag while generating the data.

(d) Research questions.

i) What is the expected memory footprint (in bytes) to build the zone map? (assume: number of elements is ***N*** and every zone has ***d*** entries)

ii) In practice, we may have raw data stored on disk, and the zone map is maintained in memory to reduce the number of required I/Os to answer queries. What should we do if we only have a limited memory budget to build the zone map (i.e., when the memory budget of ***M*** bytes is smaller than the expected memory footprint)?

Deliverables: Zone map implementation code that runs the test cases. It is *required* to have comments within the implementation, that explain various design decisions. **Do NOT upload** your code to public repositories, such as GitHub and Bitbucket.

[1] M. Ziauddin, A. Witkowski, Y. J. Kim, D. Potapov, J. Lahorani, and M. Krishna. 2017. Dimensions based data clustering and zone maps. PVLDB Endow. 10(12), pp. 1622–1633. DOI: https://doi.org/10.14778/3137765.3137769.