Kaiman

Technical Specification

Version 1.0.0

Lucas Gabriel Brito Silveira

May 28, 2023

Contents

1 Introduction

Kaiman is a simple in-memory key-value database. Written in C++ 17 and inspired in the std::multiset, Kaiman was initially designed for homework in the Advanced Data Structures class. The most significant difference between Kaiman and multiset behavior is that a Kaiman can store multiple entries if the same key saves in a stack.

2 Interface

Kaiman provides the following methods and attributes to access data.

2.1 Methods

- insert(X key, T value): Inserts a key-value entry in the kaiman.
- clear(): erase all values on the kaiman.
- erase(X key): erase if exist the entry with this key, in case of more than one entry with the key all entries are removed.
- find(X key): return a std::stack<T> * who contains all values with the key.
- empty(): return true if the kaiman is empty, false otherwise.
- size(): return the number of entries in the kaiman.

2.2 Example

```
Kaiman<int,std::string> kaiman;
kaiman.empty();
//true
kaiman.size();
//0
kaiman.insert(1,"Banana");
kaiman.empty();
//false
kaiman.size();
//1
kaiman.find(2);
// return [];
kaiman.find(1);

// return ["Banana"]
```

Listing 1: Example of code

3 Theoretical background

Kaiman is a wrapper class to a balanced tree, with the purpose of abstracting from the user the business logic to store the data in the memory in an excellent way to save memory and performance. The use Kaiman recommended to use depends on the kind of use data in the application, in the moment Kaiman is constructed using the AVL Tree.

AVL Tree is a type of search binary tree. Perform some operations during the operations that update the tree structure, removing or adding a node. these operations allow the tree itself to adjust to permit the best time to access the data stored. Below are listed the complexity time of operations in this tree.

Operation/Case	Average-case	Worst case
Insertion	O(logN)	O(logN)
Deletion	O(logN)	O(logN)
Searching	O(logN)	O(logN)

Figure 1: Complexity

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Algoritmos: Teoria e Prática*. Elsevier, 3rd edition, 2009.

[2] Jayme Luiz Szwarcfiter and Lilian Markenzon. *Estruturas de Dados e Seus Algoritmos*. Editora LTC, 3rd edition, 2023.