University of São Paulo
Institute of Mathematics and Statistics
Bachelor's Degree in Computer Science

Final Report of MAC0215
Curricular Project in Research

Persistent and Retroactive Data Structures

Bruno Armond Braga

Advisor: Prof. Dr. Cristina Gomes Fernandes

São Paulo
December, 2023

Abstract

A conventional data structure (DS) generally supports query and modification operations. Normally, when a modification occurs, the DS is updated in such a way that we lose its previous state. Thus, we only have access to the current state of the DS.

The so-called persistent and retroactive data structures aim to provide the user with support for operations on all versions of the DS. They have applications in various contexts, such as in database systems, functional languages, version control systems, etc.

The concept of persistence was formally introduced in 1986 by Driscoll, Sarnak, Sleator, and Tarjan, while the idea of retroactivity was introduced by Demaine, Iacono, and Langerman in 2007. In both cases, the user can query a past version of the DS, with each version of the DS being linked to a point in time. What distinguishes these two types of DS is how they handle a modification applied to a version of the DS in the past.

In a persistent DS, a change at time t in the past causes a branching, starting a new branch of the DS state from that moment t. Queries to previous versions continue to give the same answers, and several "parallel" versions of the DS coexist.

In retroactivity, however, a modification to the DS at time t in the past affects the state of the DS at all times after t. Queries to the DS at a time after t may start to return different answers than the ones obtained before the modification occurred.

Achieved Results

This research demonstrates that temporal data structures are quite sophisticated and offer impressive operations.

Throughout the study, the following structures were explored and implemented:

- **Retroactivity**
  - Implementation of a stack with the operations: insert push, insert pop, delete push, delete pop, size, top, k-th and print (Chap. 3, [2]).
- **Persistence**
  - Implementation of a stack with the operations: push, pop, size, top, and with k-th operation consuming linear time in the number of elements in the stack (Chap. 3, [3]);
  - Implementation of an alternative stack with k-th operation using binary counters, consuming logarithmic time (Chap. 1, [3]);
  - Implementation of another alternative stack, this one with k-th operation using skew-binary representation, also with logarithmic time complexity (Chap. 2, [3]);
  - Implementation of a queue with the operations: enqueue, dequeue, size, first, last, and k-th (Chap. 3, [3]);
  - Implementation of a functional treap;
  - Implementation of a functional binary search tree;
  - Implementation of a deque with level ancestor (LA) and lowest common ancestor (LCA) (Chap. 4, [3]);
  - Implementation of a recursive deque proposed by Kaplan (Chap. 5, [3]);
  - Implementation of a more efficient deque proposed by Kaplan and Tarjan, with all operations having cost O(1) except for the k-th operation, which has a logarithmic cost (Chap. 6, [3]).

The implementation of all these structures can be found in the GitHub repository provided at the beginning of the research:

`https://github.com/BrunoArmondBraga/EstruturasTemporais`

History of Activities

- August (4 weeks): 22h
  - Initial study on temporal structures - 5 hours.
  - Study and implementation of different persistent stacks. The first with linear k-th operation, another with k-th using binary pointers, and a last one using jump pointers - 10 hours.
  - Study and implementation of a persistent queue - 4 hours.
  - Initial study of persistent deques - 3 hours.
- September (4 weeks): 23h
  - Implementation of a persistent treap using level ancestor and lowest common ancestor - 8 hours.
  - Study and implementation of a functional binary search tree - 7 hours.
  - Study and implementation of a retroactive stack - 8 hours.
- October (4 weeks): 23h
  - Study of the recursive persistent deque proposed by Kaplan - 10 hours.
  - Implementation of the recursive persistent deque proposed by Kaplan with void pointers - 13 hours.
- November (5 weeks): 26h
  - Study of the persistent deque proposed by Kaplan and Tarjan - 11 hours.
  - Implementation of the persistent deque proposed by Kaplan and Tarjan with void pointers - 15 hours.
- December (1 week): 11h
  - Development of the k-th operation in Kaplan and Tarjan's persistent deque - 5 hours.
  - Development of the report and final organization of the GitHub's repository - 6 hours.

References

[1] Cristina Gomes Fernandes, Lecture notes for MAC0385 - Advanced Data Structures, 2023. Available at: `https://www.ime.usp.br/~cris/aulas/23_2_6957`.

[2] Beatriz Figueiredo Marouelli, *A Study on Retroactive Data Structures*, Undergraduate Thesis, BCC-IME-USP, 2019. Available at: `https://bccdev.ime.usp.br/tccs/2019/bfm/`.

[3] Yan Soares Couto, *Persistent Data Structures*, Master's Thesis, IME-USP, 2018. Available at: `https://www.teses.usp.br/teses/disponiveis/45/45134/tde-24092019-181655/`.