Advanced LRU Cache using Treaps, HashMaps, and Linked Lists

Tejas Joshi, Kaustubh Kharat, Tejas Kolhe

March 27, 2025

Abstract

The Least Recently Used (LRU) Cache is a critical component in caching mechanisms, providing efficient data retrieval while managing limited memory. In this project, we implement an advanced LRU cache using Treaps, HashMaps, and Doubly Linked Lists to optimize data access time and maintain order dynamically. This report details the problem, objectives, data structures, and flowchart for the project.

1 Problem Statement

Efficient memory management is crucial in modern computing. Traditional LRU Cache implementations using Doubly Linked Lists and HashMaps provide fast operations but lack ordered retrieval capabilities. This project enhances the LRU Cache by incorporating a Treap, enabling ordered access while maintaining the standard O(1) get and put operations.

2 Aim

To develop an optimized LRU Cache utilizing Treaps for ordered access, HashMaps for fast lookup, and Doubly Linked Lists for efficient eviction policies.

3 Project Objectives

- Implement an LRU Cache with O(1) get and put operations.
- Integrate Treaps to maintain key ordering dynamically.
- Use HashMaps for quick key lookup.
- Utilize Doubly Linked Lists to track recently used elements.
- Develop a menu-driven C++ program for user interaction.

4 Illustration of Suitable Data Structures

The following data structures are used:

- **Treap**: A randomized binary search tree for ordered key access.
- **HashMap**: Provides O(1) time complexity for key lookup.
- **Doubly Linked List**: Enables quick insertion, deletion, and LRU eviction.

5 Flowchart

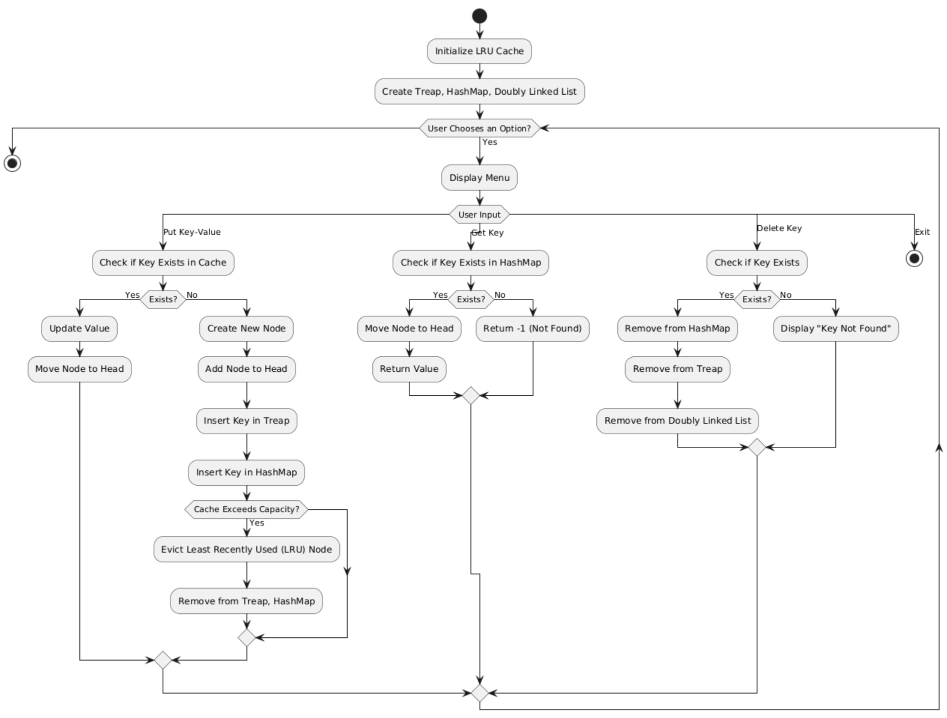

Figure 1: LRU Cache Flowchart

6 References

1. Tenenbaum, A. M. *Data Structures Using C and C++.* Pearson Education.
2. Goodrich, M. T., Tamassia, R. *Algorithm Design.* Wiley.
3. Wikipedia: Least Recently Used (LRU) Cache