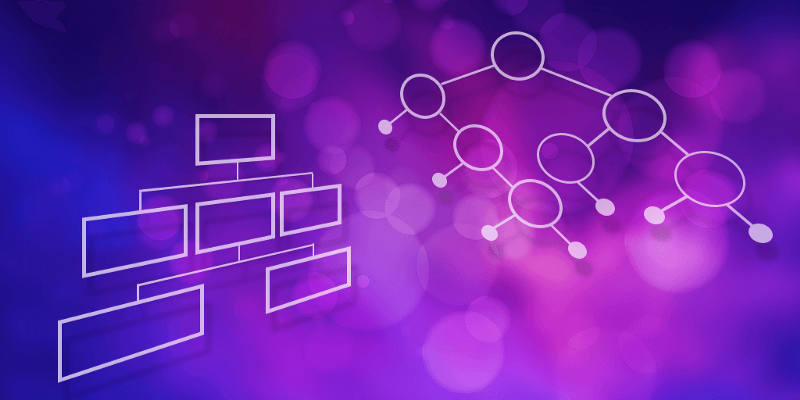
DSA PROJECT

Implementing LRU and FIFO replacement algorithms



**Ankit Jaiswal (191070081)**

**Tanish Sawant (191070064)**

**Aryan Khurana (191070008)**

**Aryan Devikar (191070007)**

# Problem Statement:

Design and implement LRU cache and FIFO cache using appropriate data structures, so that replacement can occur in Θ(1) time.

# Motivation:

Least Recently Used and First In First Out are the cache replacement algorithms that are included in the Computer Organization and Architecture course. We have learnt in a data structure course that a doubly linked list provides constant time deletion and hashtable provides nearly constant time access. This inspired us to combine the knowledge of these two courses and to implement LRU and FIFO cache.

# Methodology:

LRU (Least Recently Used) cache is a cache replacement algorithm that removes the least recently used data in order to make room for new data. According to this algorithm, the latest used data is considered to be important, so when the memory cache is full, it prioritizes removing the data that hasn't been used for long. This type of cache is built from a combination of two data structures, those being linked lists and hash tables.

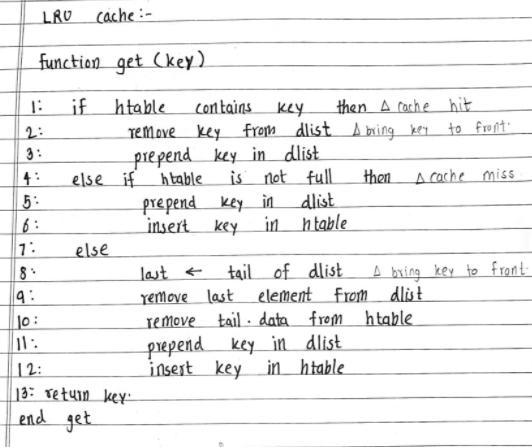
We need the entries to be maintained in a sorted list based on the order in which the entries were accessed. An alternate way to achieve this would be to maintain the entries in a doubly linked list using which every time an entry is accessed ( a cache lookup operation), the entry is also moved to the end of the list. When we need to purge the entries it is done from the top of the list. In an ArrayList when an element is removed the rest of the entries need to be moved by one to fill the gap. A doubly linked list does not have this issue.

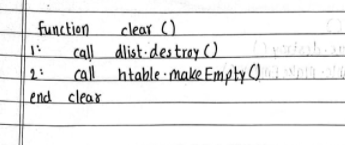
FIFO (First-In-First-Out) replaces the first page that was placed in the cache with the desired page, and the next replacement will be the second page placed in the cache and so on. This type of cache is built from the queue data structure.

We can use a map for the cache and a doubly linked list to implement the FIFO queue. When the cache is full, we remove the cached item that is at the front of the FIFO queue since it was added first and add the new item at the tail. Due to the usage of hash tables, the lookups, adds & deletes are fast and occur at a constant time.

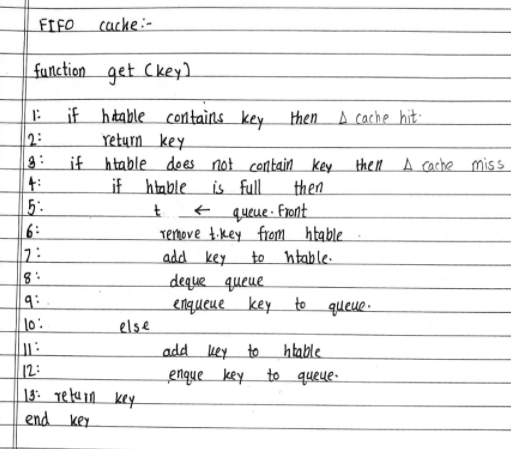
# Pseudocode:

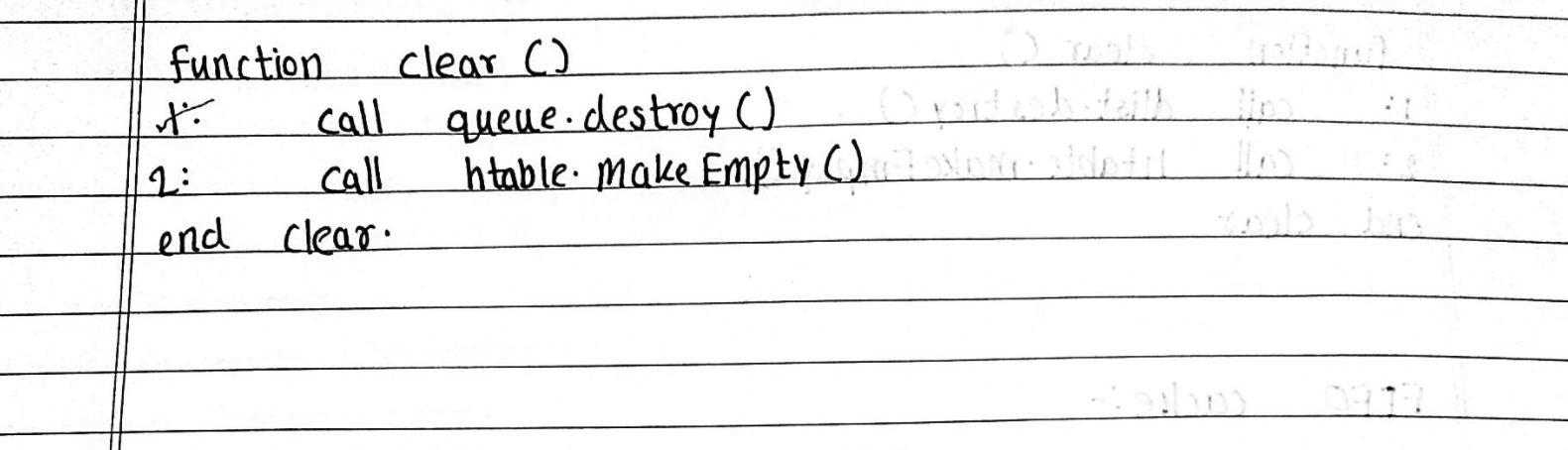
**LRU Cache:**





**FIFO Cache:**





# 

# Output:



# 

# Discussion:

In this experiment, we have implemented LRU and FIFO replacement algorithms in cache, the theory of which we studied under our Computer Organisation and Architecture course and the implementation part was studied under the Data Structure and Algorithms course.

We have used the concept of linked lists, hashing and standard data structures for FIFO i.e. queues. Whenever all the lines of cache are filled, we need to replace the block contained in one of the lines with another block containing the desired word in order to get the desired word by processor. LRU or Least Recently Used Algorithm involves replacing the block that has been in the cache line for the longest time and has least memory references. For this, we keep on inserting and deleting nodes in order to set priority. In any case, the word is not present in the cache line.

We use an efficient method of searching i.e. quadratic probing, to get the desired block in the line of cache. We implement a similar program in FIFO however with some changes, since we need to replace blocks that have been fetched for the longest time for which we use queue data structure.

Thus, we have successfully applied the theoretical concepts we have learnt in a practical manner.