Cache (Homework 4)

Naho Yokoyama

Overview

This program implements a data structure for managing a cache. The cache uses the Least Recently Used (LRU) algorithm to store the most recently accessed N pages, while removing older pages from the cache. By combining a hash table and an array, it performs search, insert, and delete operations in approximately O(1) time. However, for updating the access order array when we access the page that is already cache, it requires O(N) at worst case.

Important functions

1. Cache the page that is accessed (access_page)
   Argument(s): url, contents (any type)
   Return value(s): none

2. Get the list of pages that were cached (get_page)
   Argument(s): none
   Return value(s): List of pages that were cached (sorted according to the time when they were accessed)

Algorithm

1. How to cache
   The addition of data is divided into two cases based on whether the data already exists in the hash table. Both cases use the class variable “index”, which indicates the location of the oldest data or in other words, the location where the newly accessed page should be swapped in.

   1.1 When the data exists in the hash table
   To update the access order array, the following steps are performed:

   1. Save the data at the current index to another variable and swap the data with the accessed page
   2. Go through the array and continue swapping the saved variable with the current data until it reaches the position of the accessed page

   1.2 When the data does not exists in the hash table

To update the access order array and the hash table, the following steps are performed:

1. Get the data at the index position in the access order array
2. Delete the data corresponding to the obtained data in step 1 from the hash table (refer to the deletion method in the hash table documentation from Homework 1)
3. Insert the new data at the index position in the access order array
4. Add the new data to the hash table (refer to the addition method in the hash table documentation from Homework 1)

2. How to get the list of cached pages

   The access order array holds a list of page URLs, which are returned by tracing back from the index. However, before N additions of data has been performed, there will be None entries in the access order array. These entries would be eliminated before returning the result.

Key points to be considered

1. How to update the variable index

   By utilizing the fixed length of the cache (with a maximum size of N), the “index” is updated by incrementing by 1 and using the remainder when divided by N. This approach ensures that once the end of the array is reached, the “index” moves back to the beginning, allowing for seamless circular updates without the need for writing complicated programs.

What I discussed with the mentor

1. About the use of queue

   Using an access-order array and an index for processing essentially does the same thing as a queue. However, even in the case of a queue, updating the access order of already cached pages takes O(N) in the worst case, just like with the access-order array.

2. How to perform updates in O(1)

   By using a doubly linked list instead of an access-order array and maintaining pointers to the list in a hash table, it is possible to perform all operations, including addition, search, deletion, and update, in almost O(1) time.