Lab 5 LRU and Heaps

Introduction

This lab involves two problems, A and B. Problem A has you implement a data structure called LRU, Least Recently Used cache. Basically it's a hash table, the value points to a node that is in a doubly linked list. This list is capped at a certain size, meaning once it moves past this size then it sheds the last node in the list. If you insert a key and value that is already in the list then it moves to the head of the doubly linked list.

Problem B is much more straightforward, given list a list print the k most frequent elements in descending order. Print the word and the amount of times it appears in the list, it's a requirement to use a heap.

Design

Problem A

So I used implemented a hashtable to start with, this implementation comes from my professor. I then wrote a quick implementation of a doubly linked list, my node implementation I made sure to include a function called "remove()". This function will remove the node from the list, when a node needs to be moved to the head or deleted altogether.

When a key and value are added to the LRU a node is stored into the table, this node contains both value and key, the node is also added to the list. Now, before this happens we check if that value or key is in the table already, by searching the table itself, now this is done in constant time. If the node is discovered then the node is removed and the process begins again.

The only parameter the LRU takes is the max size.

Problem B

This was a bit more difficult but I got it done. We were given a basic heap implementation, I modified this implementation to receive a node or a Datawrapper as I called it. This Datawrapper is meant to take any data type along with a key. The data in this case would be the word, and the key would be the amount of occurrences in the list.

Before I load the heap, I insert all words in the list into a hash table using the word as the key and value as the amount of time its appeared in the list.

Results

Problem A is constant, passed all test cases.

Problem B is N + log.

Overall, the time to insert is quite fast, one of my test cases I used inserted 9000 words and it was done in 0.06322097778320312 seconds.

Conclusion

Overall, the main learning experience was implementing simple data structures with more of a twist. The LRU is simply a hash table with a doubly linked list attached that is capped by a certain size. You also need to be making rotations at all times. I had to edit the heap to be ready account for the datawrapper object that I created.