LRU Cache - Explanation

For this problem, I decided to use the following data structures:

- Hashtable (Dictionary)
- Doubly Linked List

A hash table is an unordered collection of key-value pairs, where each key is unique. It allows us to do fast lookups at constant time. The value of the key-value contains a reference to a node in the doubly linked list.

The second data structure chosen for the LRU cache is a doubly-linked list. The goal is to perform fast removal operations and append an element to the tail of the list. The least used element in the cache can be found at the head of the list, whereas the most recently used one is located at the tail.

Comment: A simpler solution in Python would be to use only `OrderedDict` *. I've decided against it because I wanted to take a more general and language-agnostic approach to solve this task. In a real-world scenario, I would have asked the interviewer if it is okay to solve this task with* `OrderedDict`

Summary

Operation	Goal	Data Structure	Time Complexity	Space Complexity
set()	Fast Removal	Doubly Linked List	O(1)	O(n)
get()	Fast Lookup	Hash Table	O(1)	O(n)