Least Recently Used (LRU) Cache Visualizer - Documentation

1. Introduction

The Least Recently Used (LRU) Cache Visualizer is a Python-based tool that provides a graphical representation of how the LRU cache algorithm works. It allows users to interact with a cache, adding and retrieving items, and visualize how the cache manages its contents based on usage.

2. Core Functionality

2.1 LRU Cache Mechanism

The LRU cache stores a limited number of key-value pairs. When the cache reaches its capacity, adding a new item triggers the eviction of the *least recently used* item. This ensures that the most frequently accessed items remain in the cache, improving retrieval efficiency.

Description:

- Maintains a cache of a fixed size.
- Evicts the least recently used item when the cache is full.
- Uses a data structure (e.g., a doubly linked list combined with a hash map) to track item usage and facilitate efficient eviction.

2.2 Operations

- Put (Key, Value): Adds a new key-value pair to the cache. If the key already exists, its value is updated, and it's moved to the most recently used position. If the cache is full, the LRU item is evicted before the new item is added.
- Get (Key): Retrieves the value associated with a given key. If the key is found, the item is moved to the most recently used position. If the key is not found, it returns a "Cache Miss" indicator (e.g., None).

3. How to Use

1. Set Cache Size: Specify the maximum number of items the cache can hold.
2. Input Key and Value: Enter the key and value you want to add to the cache.
3. Put: Click the "Put" button to add/update the key-value pair in the cache.
4. Input Key (for Get): Enter the key you want to retrieve from the cache.
5. Get: Click the "Get" button to retrieve the value associated with the key.
6. View Cache State: The current contents of the cache are displayed, showing the key-value pairs and their order (from most recently used to least recently used).

4. Visualization

The visualization area displays the cache's contents. A clear representation of the order of items (indicating recency of use) is provided. When an item is added or retrieved, the visualization updates to reflect the change in the cache's state and the movement of items based on the LRU algorithm. Evictions are visually highlighted.

5. Example

- Cache Size: 3
- Put (A, 1): Cache: {A: 1}
- Put (B, 2): Cache: {A: 1, B: 2}
- Put (C, 3): Cache: {A: 1, B: 2, C: 3}
- Put (D, 4): Cache: {B: 2, C: 3, D: 4} (A is evicted as it's the LRU)
- Get (C): Cache: {B: 2, D: 4, C: 3} (C is moved to the most recently used position)

The visualization would show these transitions, clearly indicating the LRU item being evicted and the changes in item order.

6. Notes

- The visualization aims to provide a clear understanding of the LRU cache's behavior.
- The implementation may use animations or other visual cues to enhance the visualization.
- Error handling (e.g., for invalid input) is included.