

Tanish Mittal

IT-C

2000290130174

Problem 2: Design a data structure that follows the constraints of Least Recently Used (LRU) cache

class LRUCache:

class Node:

def __init__(self, key=None, value=None):

self.key = key

self.value = value

self.prev = None

self.next = None

def __init__(self, capacity):

self.capacity = capacity

self.cache = {}

self.head = self.Node()

self.tail = self.Node()

self.head.next = self.tail

self.tail.prev = self.head

def _add_node(self, node):

Add a node to the front of the linked list

node.prev = self.head

node.next = self.head.next

self.head.next.prev = node

self.head.next = node

def _remove_node(self, node):

Remove a node from the linked list

prev = node.prev

next = node.next

```
prev.next = next
```

```
next.prev = prev
```

```

def _move_to_front(self, node):

    # Move a node to the front of the linked list

    self._remove_node(node)

    self._add_node(node)


def _pop_tail(self):

    # Remove and return the tail node from the linked list

    tail_node = self.tail.prev

    self._remove_node(tail_node)

    return tail_node


def get(self, key):

    if key in self.cache:

        node = self.cache[key]

        self._move_to_front(node)

        return node.value

    else:

        return -1


def put(self, key, value):

    if key in self.cache:

        node = self.cache[key]

        node.value = value

        self._move_to_front(node)

    else:

        new_node = self.Node(key, value)

        self.cache[key] = new_node

        self._add_node(new_node)

        if len(self.cache) > self.capacity:

            tail_node = self._pop_tail()

            del self.cache[tail_node.key]

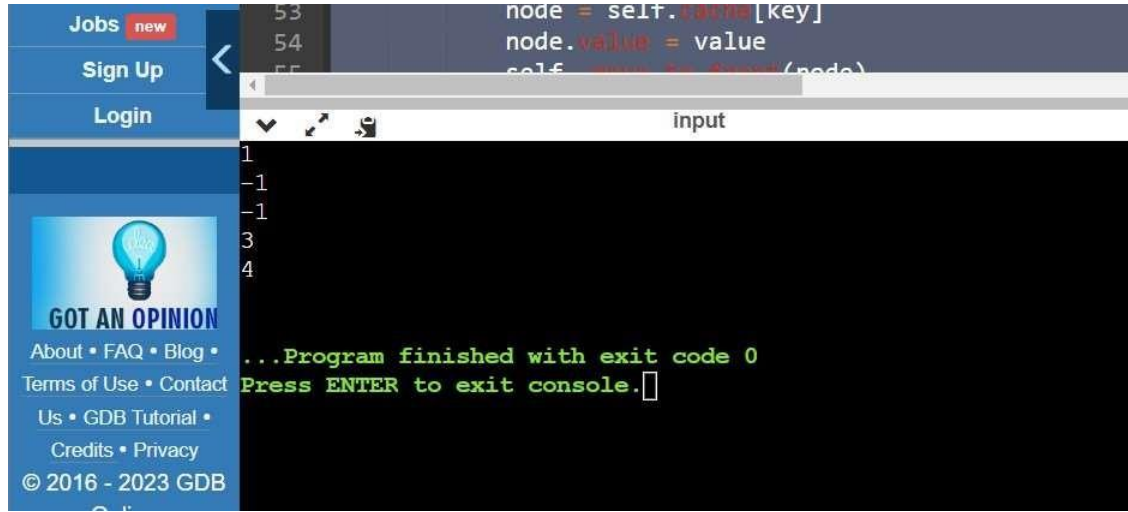

lru_cache = LRUCache(2)

lru_cache.put(1, 1)

lru_cache.put(2, 2)

```

```
print(lru_cache.get(1))  
lru_cache.put(3, 3)  
print(lru_cache.get(2))  
lru_cache.put(4, 4)  
print(lru_cache.get(1))  
print(lru_cache.get(3))  
print(lru_cache.get(4))
```



The screenshot shows a web application interface. On the left is a blue sidebar with navigation links: 'Jobs' (with a 'new' badge), 'Sign Up', 'Login', and a 'GOT AN OPINION' section with links to 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact Us', 'GDB Tutorial', 'Credits', and 'Privacy'. The main content area has a dark background. At the top, there's a code editor showing Python code for an LRU cache. Below the code editor is a terminal window with the following output:

```
1  
-1  
-1  
3  
4  
...Program finished with exit code 0  
Press ENTER to exit console.
```