Dash
All
Articles
Videos
Problems
Quiz
Contest
« Prev
Next »

LRU Cache Design

Design a data structure for LRU Cache. It should support the following operations: **get** and **set**.

- **get(key)** – Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.
- **set(key, value)** – Set or insert the value if the key is not already present. When the cache reached its capacity, it should invalidate the least recently used item before inserting a new item.

Examples:

```
// Let's say we have a LRU cache of capacity 2.
LRUCache cache = new LRUCache(2);
cache.set(1, 10); // it will store a key (1) with value 10 in the cache.
cache.set(2, 20); // it will store a key (2) with value 20 in the cache.
cache.get(1); // returns 10
cache.set(3, 30); // evicts key 2 and store a key (3) with value 30 in the cache.
cache.get(2); // returns -1 (not found)
cache.set(4, 40); // evicts key 1 and store a key (4) with value 40 in the cache.
cache.get(1); // returns -1 (not found)
cache.get(3); // returns 30
cache.get(4); // returns 40
```

Asked In: Adobe, Hike and many more companies.

Implementation:

Dash
All
Articles
Videos
Problems
Quiz
Contest
«
»

C++ Java

```
import java.io.*;
import java.lang.*;
import java.util.*;
import java.util.*;

class Node {
    int key;
    int value;
    Node pre;
    Node next;

    public Node(int key, int value)
    {
        this.key = key;
        this.value = value;
    }
}

class LRUCache {
    private HashMap<Integer, Node> map;
    private int capacity, count;
    private Node head, tail;

    public LRUCache(int capacity)
    {
        this.capacity = capacity;
        map = new HashMap<>();
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

```
        head = new Node(0, 0);
        tail = new Node(0, 0);
        head.next = tail;
        tail.pre = head;
        head.pre = null;
        tail.next = null;
        count = 0;
    }

    public void deleteNode(Node node)
    {
        node.pre.next = node.next;
        node.next.pre = node.pre;
    }

    public void addToHead(Node node)
    {
        node.next = head.next;
        node.next.pre = node;
        node.pre = head;
        head.next = node;
    }

    public int get(int key)
    {
        if (map.get(key) != null) {
            Node node = map.get(key);
            int result = node.value;
            deleteNode(node);
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

```
            addToHead(node);
            System.out.println("Got the value : " + result
                               + " for the key: " + key);
            return result;
        }
        System.out.println("Did not get any value"
                           + " for the key: " + key);
        return -1;
    }

    public void set(int key, int value)
    {
        System.out.println("Going to set the (key, "
                           + "value) : (" + key + ", "
                           + value + ")");
        if (map.get(key) != null) {
            Node node = map.get(key);
            node.value = value;
            deleteNode(node);
            addToHead(node);
        }
        else {
            Node node = new Node(key, value);
            map.put(key, node);
            if (count < capacity) {
                count++;
                addToHead(node);
            }
            else {
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

```
                map.remove(tail.pre.key);
                deleteNode(tail.pre);
                addToHead(node);
            }
        }
    }
}

public class TestLRUCache {
    public static void main(String[] args)
    {

        LRUCache cache = new LRUCache(2);

        // it will store a key (1) with value
        // 10 in the cache.
        cache.set(1, 10);

        // it will store a key (2) with value 20 in the
        // cache.
        cache.set(2, 20);
        System.out.println("Value for the key: 1 is "
                           + cache.get(1)); // returns 10

        // removing key 2 and store a key (3) with value 30
        // in the cache.
        cache.set(3, 30);

        System.out.println(
            "Value for the key: 2 is "
            + cache.get(2)); // returns -1 (not found)

        // removing key 1 and store a key (4) with value 40
        // in the cache.
        cache.set(4, 40);
        System.out.println(
            "Value for the key: 1 is "
            + cache.get(1)); // returns -1 (not found)
        System.out.println("Value for the key: 3 is "
                           + cache.get(3)); // returns 30
        System.out.println("Value for the key: 4 is "
                           + cache.get(4)); // return 40
```

Dash
All
Articles
Get 90% Refund!
Courses Tutorials Jobs Practice Contests
P
Problems
Quiz
Contest
«
»

Output:

```
Going to set the (key, value) : (1, 10)
Going to set the (key, value) : (2, 20)
Got the value : 10 for the key: 1
Value for the key: 1 is 10
Going to set the (key, value) : (3, 30)
Did not get any value for the key: 2
Value for the key: 2 is -1
Going to set the (key, value) : (4, 40)
Did not get any value for the key: 1
Value for the key: 1 is -1
```

Dash
All
Articles
Videos
Problems
Quiz
Contest
«
»

```
Got the value : 30 for the key: 3
Value for the key: 3 is 30
Got the value : 40 for the key: 4
Value for the key: 4 is 40
```

Time Complexity :-

get(key) - O(1)

set(key, value) - O(1)

Mark as Read

Report An Issue

If you are facing any issue on this page. Please let us know.