706. Design HashMap

Easy 2914 297 Add to List Share

Design a HashMap without using any built-in hash table libraries.

Implement the `MyHashMap` class:

- `MyHashMap()` initializes the object with an empty map.
- `void put(int key, int value)` inserts a `(key, value)` pair into the HashMap. If the `key` already exists in the map, update the corresponding `value`.
- `int get(int key)` returns the `value` to which the specified `key` is mapped, or `-1` if this map contains no mapping for the `key`.
- `void remove(key)` removes the `key` and its corresponding `value` if the map contains the mapping for the `key`.

Example 1:

```
Input
["MyHashMap", "put", "put", "get", "get", "put", "get", "remove", "get"]
[[], [1, 1], [2, 2], [1], [3], [2, 1], [2], [2], [2]]
Output
[null, null, null, 1, -1, null, 1, null, -1]

Explanation
MyHashMap myHashMap = new MyHashMap();
myHashMap.put(1, 1); // The map is now [[1,1]]
myHashMap.put(2, 2); // The map is now [[1,1], [2,2]]
myHashMap.get(1);    // return 1, The map is now [[1,1], [2,2]]
myHashMap.get(3);    // return -1 (i.e., not found), The map is now [[1,1], [2,2]]
myHashMap.put(2, 1); // The map is now [[1,1], [2,1]] (i.e., update the existing value)
myHashMap.get(2);    // return 1, The map is now [[1,1], [2,1]]
myHashMap.remove(2); // remove the mapping for 2, The map is now [[1,1]]
myHashMap.get(2);    // return -1 (i.e., not found), The map is now [[1,1]]
```

Constraints:

- $0 <= key, value <= 10^6$
- At most 10^4 calls will be made to `put`, `get`, and `remove`.

Accepted 305,347 Submissions 471,937

Seen this question in a real interview before? Yes No

Companies

Related Topics

Similar Questions

Java Autocomplete

```java
class MyHashMap {

    class HashTableEntries {

        private Integer key;
        private Integer value;

        public HashTableEntries( Integer key, Integer value ) {
            this.key = key;
            this.value = value;
        }

        public Integer getKey() {
            return key;
        }

        public void setKey( Integer key ) {
            this.key = key;
        }

        public Integer getValue() {
            return value;
        }

        public void setValue( Integer value ) {
            this.value = value;
        }
    }

    private final int HASH_MAP_SIZE = 100000;
    private final HashTableEntries[] hashTableEntries;

    public MyHashMap() {
        hashTableEntries = new HashTableEntries[HASH_MAP_SIZE];
    }

    public void put( int key, int value ) {

        int hash1 = hashFunc1(key);
        int hash2 = hashFunc2(key);

        int counter = 0;
        int hash = hash1 % HASH_MAP_SIZE;

        while (hashTableEntries[hash] != null && !hashTableEntries[hash].key.equals(key)) {
            counter++;
            hash = (hash1 + counter * hash2) % HASH_MAP_SIZE;
        }

        hashTableEntries[hash] = new HashTableEntries(key, value);

    }

    public int get( int key ) {

        int hash1 = hashFunc1(key);
        int hash2 = hashFunc2(key);

        int counter = 0;
        int hash = hash1 % HASH_MAP_SIZE;

        while (hashTableEntries[hash] != null && !hashTableEntries[hash].key.equals(key)) {
            counter++;
            hash = (hash1 + counter * hash2) % HASH_MAP_SIZE;
        }

        return hashTableEntries[hash] == null ? -1 : hashTableEntries[hash].value;
    }

```