LRU cache

https://www.geeksforgeeks.org/lru-cache-implementation/

class LRUCache

{

private Queue<Integer> q=new LinkedList();

private HashMap<Integer,Integer> map=new HashMap<>();

private int capacity;

public LRUCache(int capacity)

{

this.capacity=capacity;

}

public int get(int key)

{

if(map.containsKey(key))

{

removefromqueue(key);

q.add(key);

return map.get(key);

}

else

{

return -1;

}

}

public void put(int key, int value)

{

if(map.containsKey(key))

{

removefromqueue(key);

}

q.add(value);

map.put(key,value);

trimqueue();

}

public void trimqueue()

{

if(q.size()>capacity)

{

int temp=q.poll();

map.remove(temp);

}

}

public void removefromqueue(int key)

{

q.remove(key);

}

}

/\*\*

\* Your LRUCache object will be instantiated and called as such:

\* LRUCache obj = new LRUCache(capacity);

\* int param\_1 = obj.get(key);

\* obj.put(key,value);

\*/