→ LRU Cache in Linked List

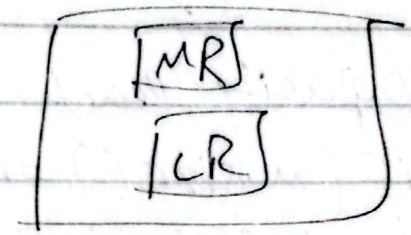
Cachee

→ Part of memory to store some data for fast processes.

→ Temperory data → store in 'key, val'

LRU

(Least Recently Used)



LRU Cache(2)
put(1,1)
put(2,2)
get(1)
put(3,3)
get(2)
put(4,4)
get(1)
get(3)
get(4)

Use:-

- DLL
  - Head
  - MR
- Map
  - Tail
  - LR

Enables (Functionalize get()

```
class Node{
  public
  int key,val;
  Node* prev;
  Node* Next;
  Constructor()
```

Functions Logic

```
Put(key,val){
 if(m[key] != m.end()){
    oldNode = m[key]
    delNode(oldNode)
    m.erase(key)
        }
Capacity Reach
 if(m.size() == limit){
    m.erase(tail->prev->key)
     delNode(tail->Prev)
    }
 Node* newNode = newNode(key,val)
    addNode(newNode)
    m[key] = newNode
 delNode(oldNode){
    oldprev = oldNode->prev
 oldNext = oldNode->next
    oldprev->next = oldNext
     oldNext->prev = oldprev
 addNode(newNode){
          L insert at head next
 Node* oldNode = head->next
 head->next = NewNode
 oldNext->prev = newNode
 newNode->next = oldnext
 newNode->prev = head
```

```
PUT(key){
if(m.find(key) == m.end())
    return -1
    ans = m[key]->val
ansNode = m[key]
  m.erase(key)
delNode = (ansNode)
oldNode(ansNode)
m[key] = ansNode
    return -1
```

TC = O(n)