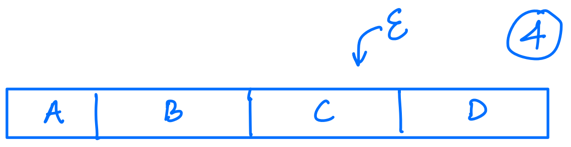
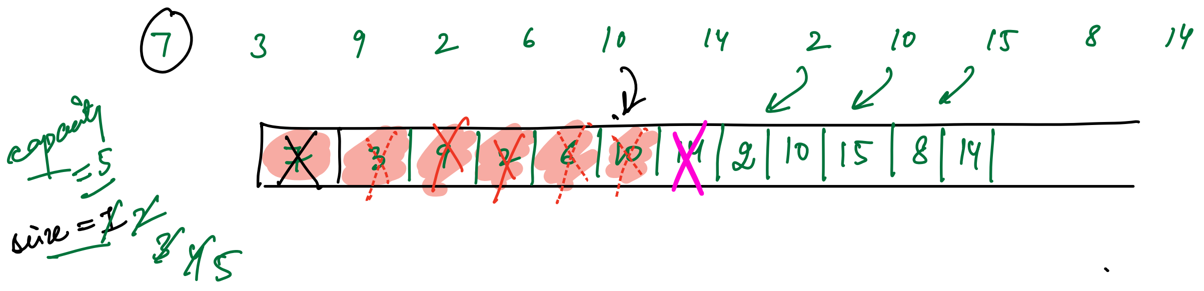
Cache ? – storage space

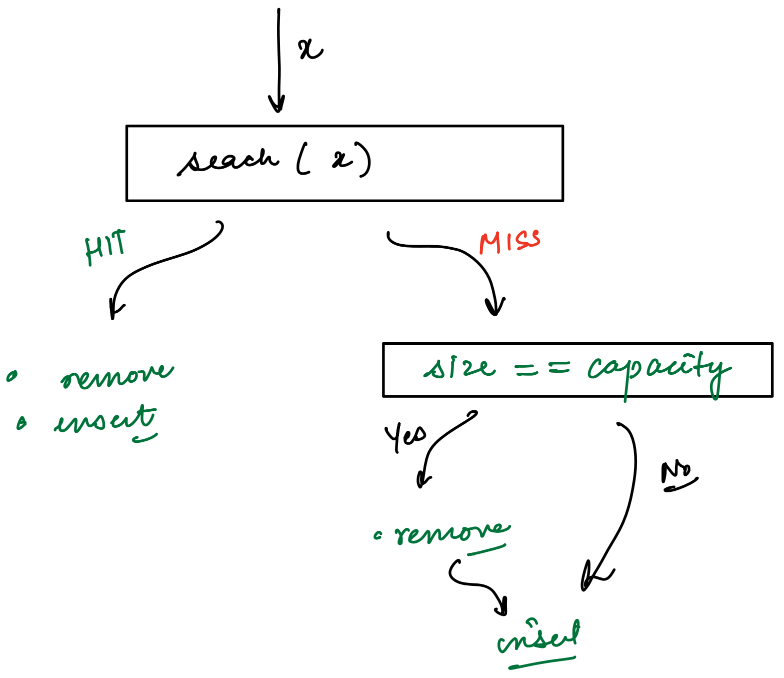
Secondary .
RAM
cache



LRU cache
↓
least Recently used



- search
- delete
- insert

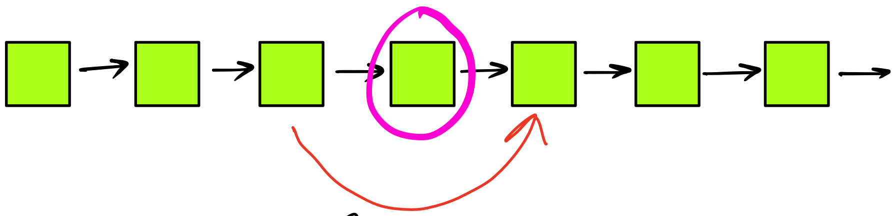


	array	single LL	SLL + Hashmap
search	O(n)	O(n)	O(1)
insert at end	O(1)	O(1)	O(1)
deletion	O(n)	O(1)	O(N)

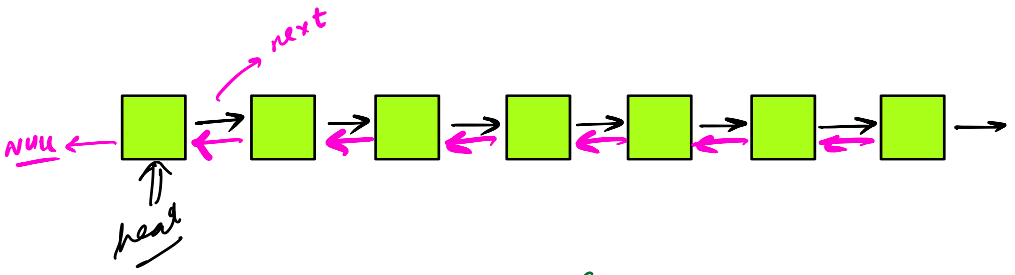
→ <data, address of node>

maintain tail

help from search



DLL : Doubly Linked List



```
class Node {
    int data;
    Node next;
    Node prev;
}
```

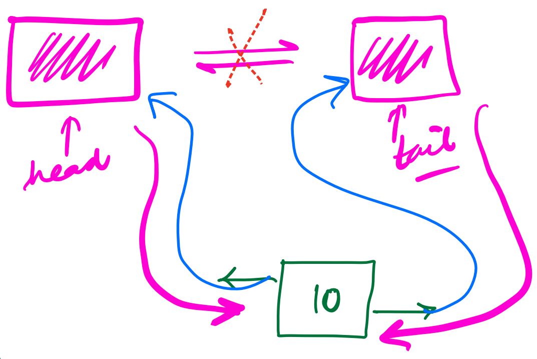
int key;
int value;

DS: DLL + Hashmap

<10, —> (15, <>)

10 ↑ 15 ↑ 19 ↑ 20 ↑ 15 ↑ 18 ↑ 23 ↑ 20 19 17 17

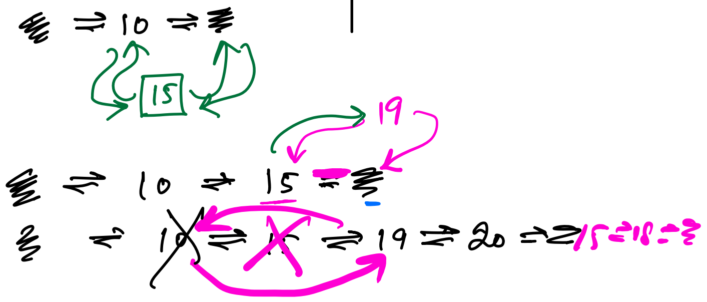
Cap=5
size=0



HM

<10, addes>
<15, add>
<19, add>
<20, add>
<18, add>

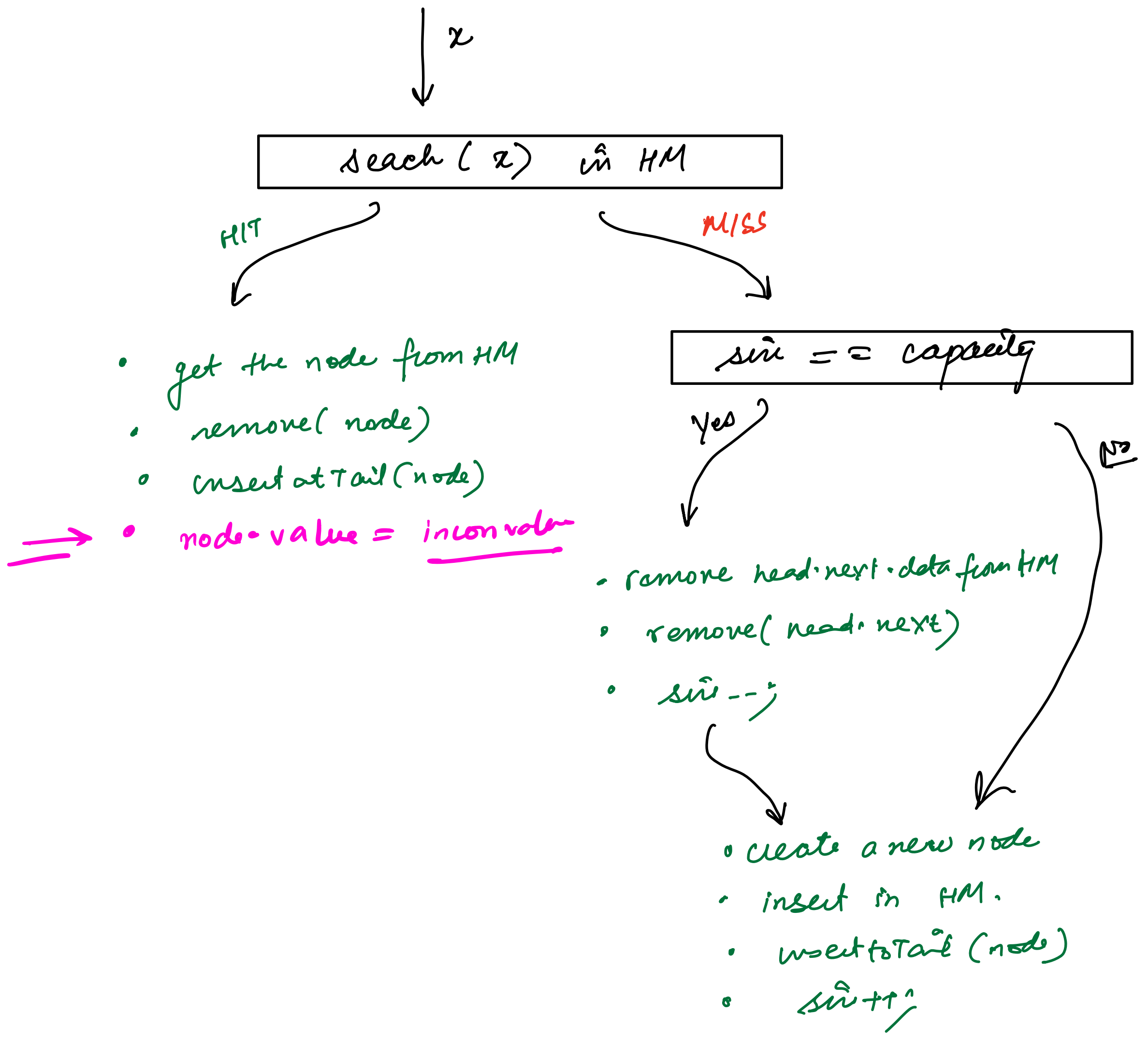
```
insertatTail(x)
{
    x.next = tail
    x.prev = tail.prev
    x.prev.next = x
    tail.prev = x
}
```



```
temp = HM[15]
remove(temp)
{
    temp.prev.next = temp.next;
    temp.next.prev = temp.prev;
}
```

remove from HM(head.next.data)
remove(head.next)

⇌ 19 ⇌ 20 ⇌ 15 ⇌ 18 ⇌ 23 ⇌



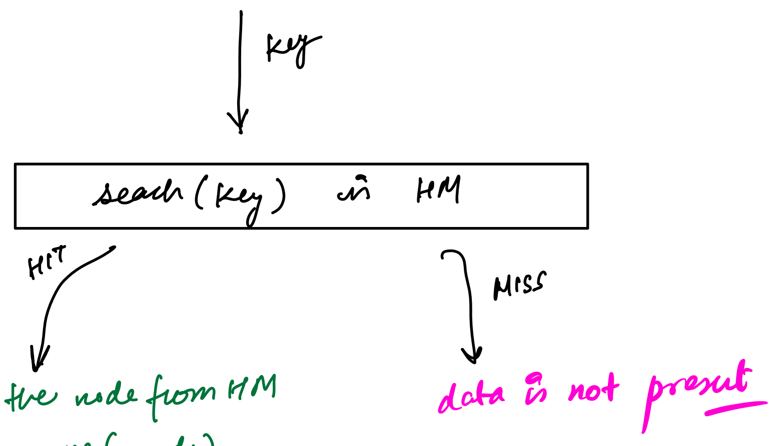
<Key, value>

get (key)

it takes a key & return the value associated

put (key, value)

get ( ) ~~te~~ key



- get the node from HM
- remove (node)
- insertAtTail (node)
- return node.value;