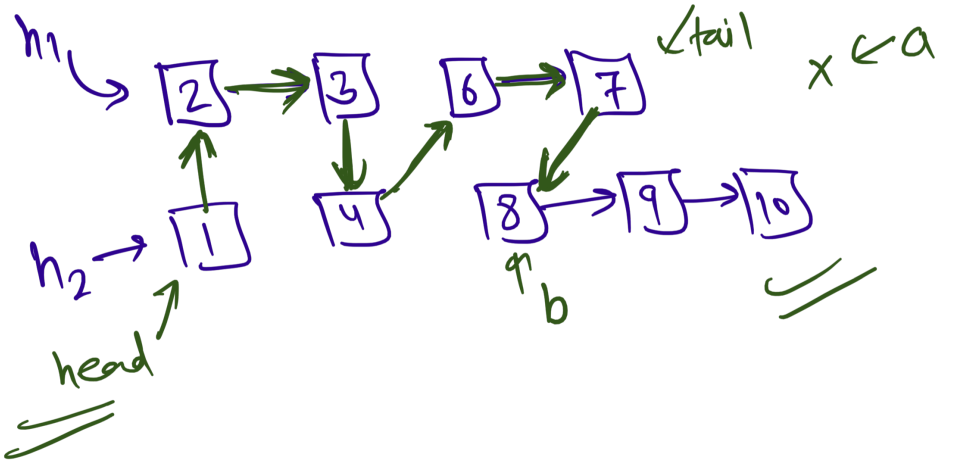
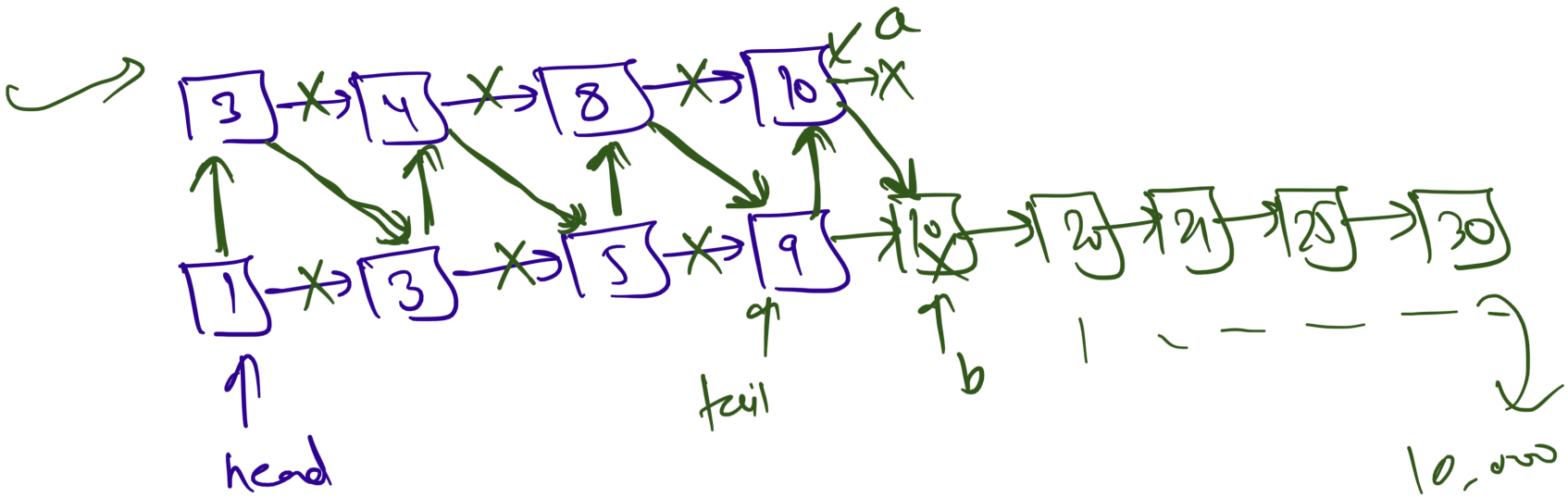
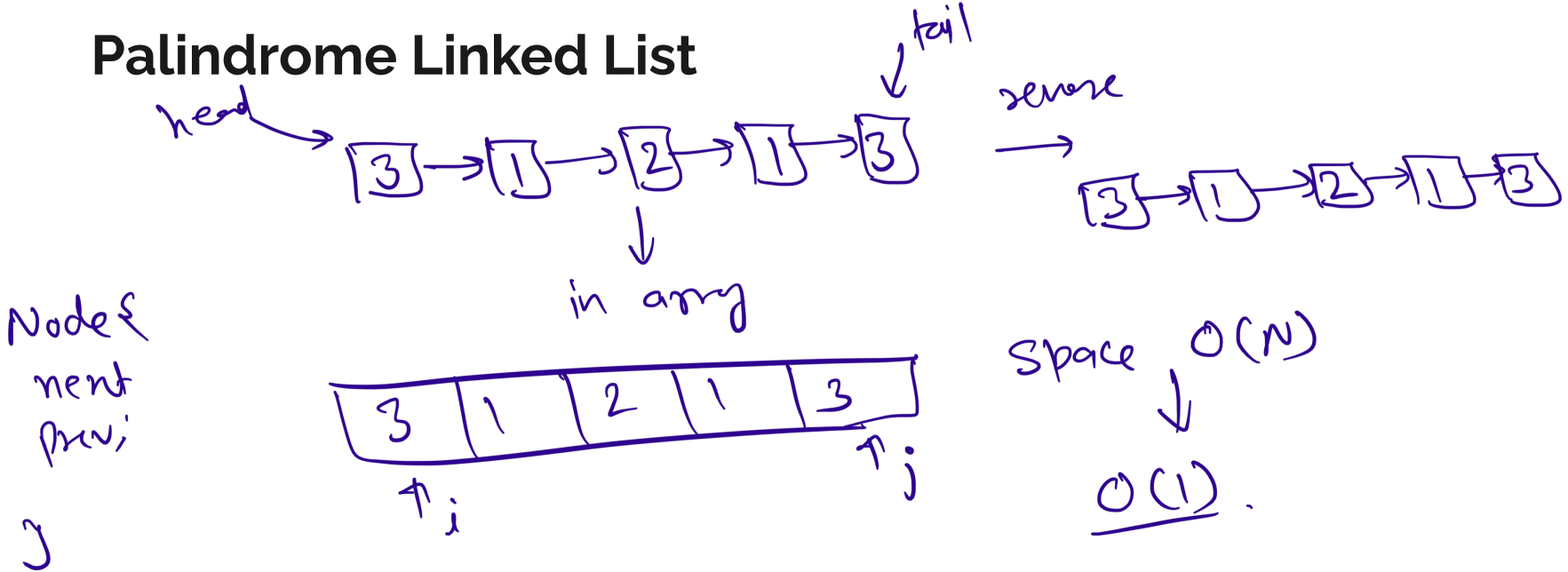
Linked List Problems - II

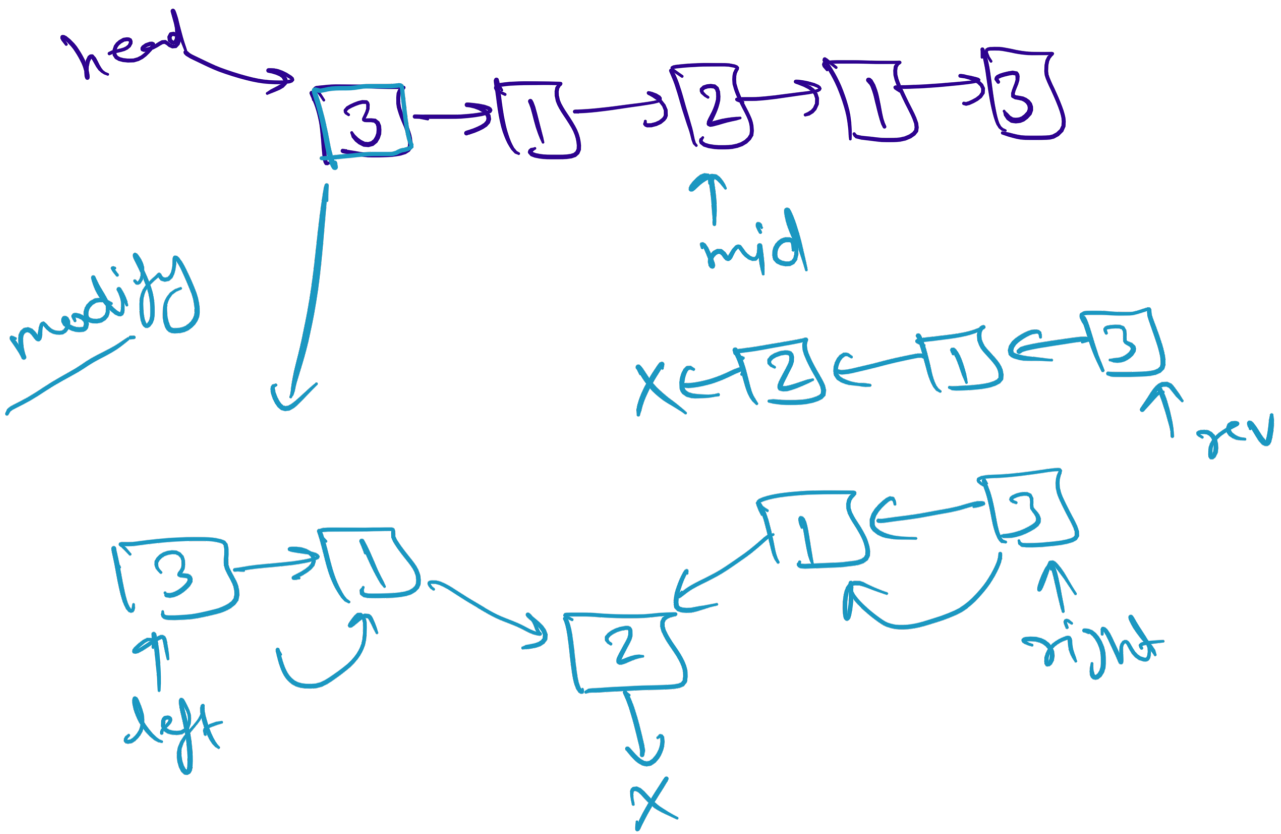
Merge two sorted Linked List in Place

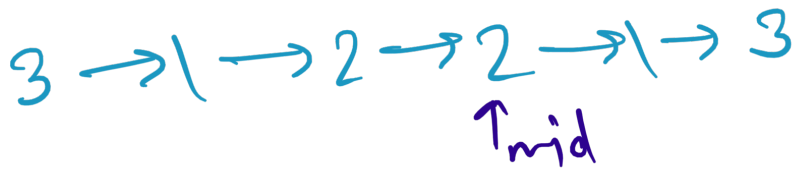


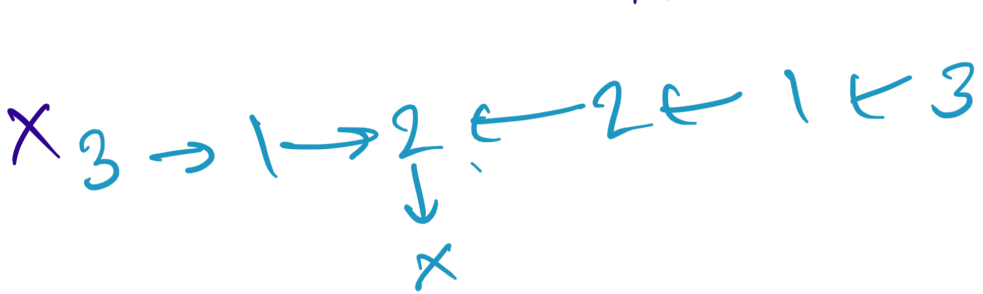

3
4
8
10
a
1
3
5
9
10
23
21
25
30
head
tail
b
10,000

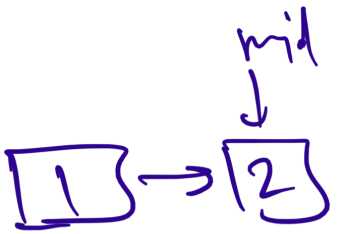
Palindrome Linked List

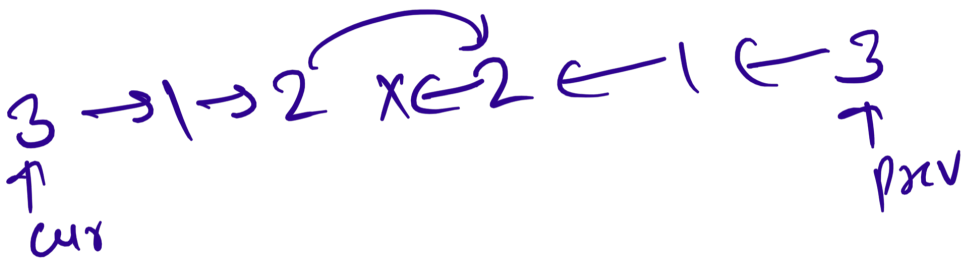


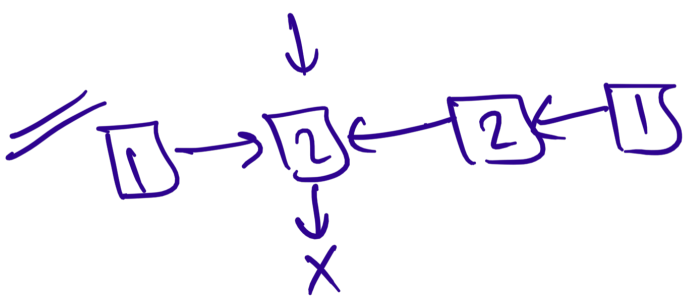

head
3
1
2
1
3
mid
modify
X
2
1
3
rev
3
1
2
1
3
left
right
X

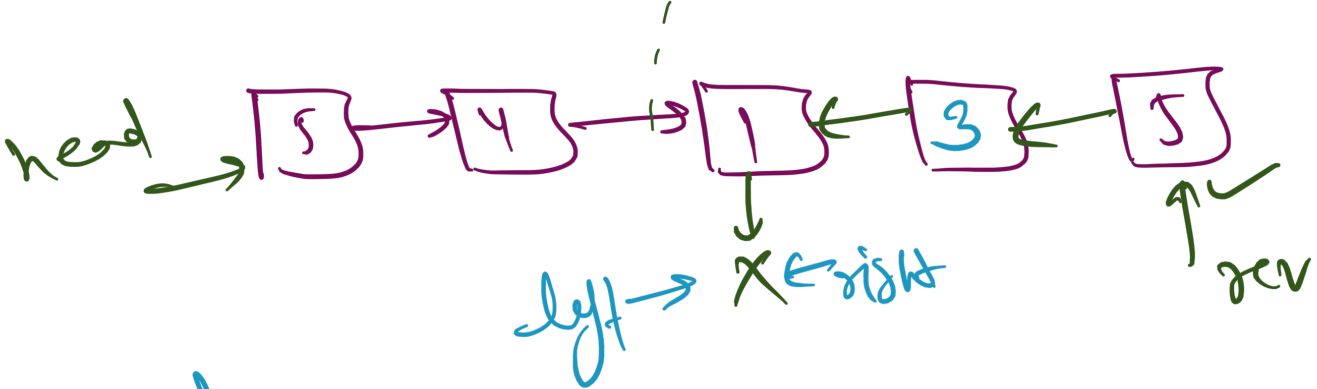

3 → 1 → 2 → 2 → 1 → 3
mid

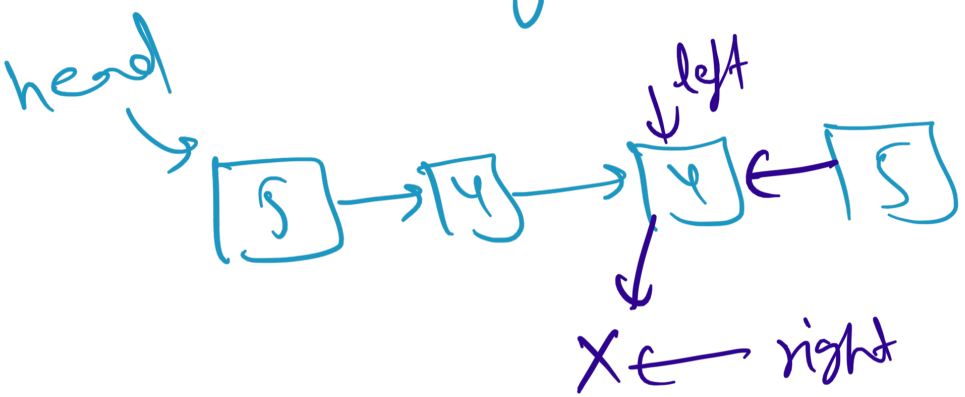

3 → 1 → 2 ← 2 ← 1 ← 3
x


mid
1
2

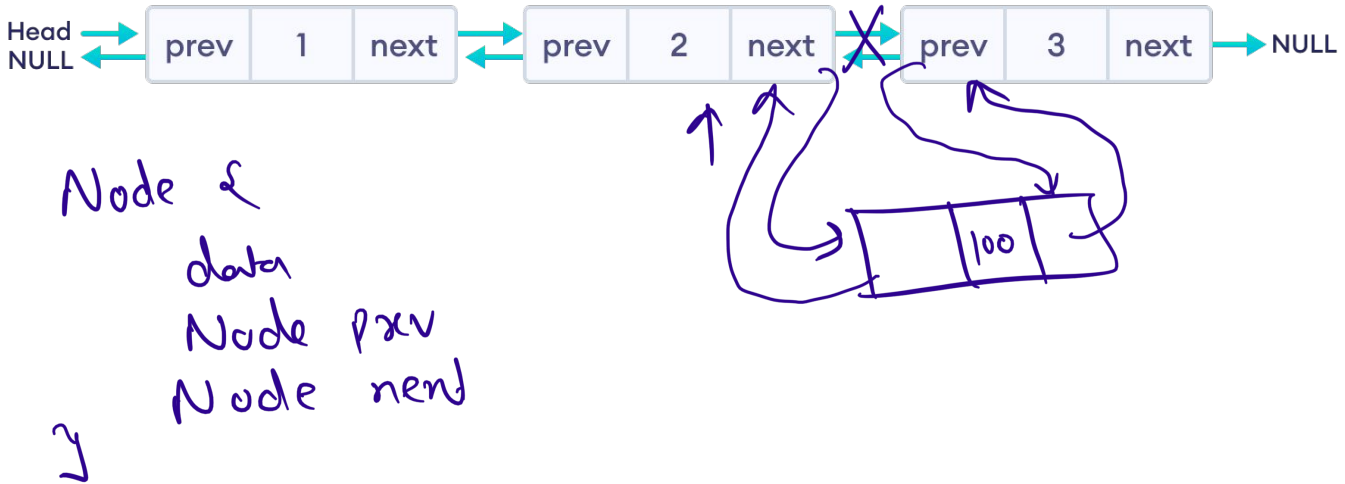

3 → 1 → 2 x ← 2 ← 1 ← 3
cur
prv


1
2
2
1
x


head
5
4
1
3
5
left
X
right
rev


head
left
5
4
4
5
X
right

Doubly Linked List Implementation



- Insert after a given Node
- Delete after a given Node

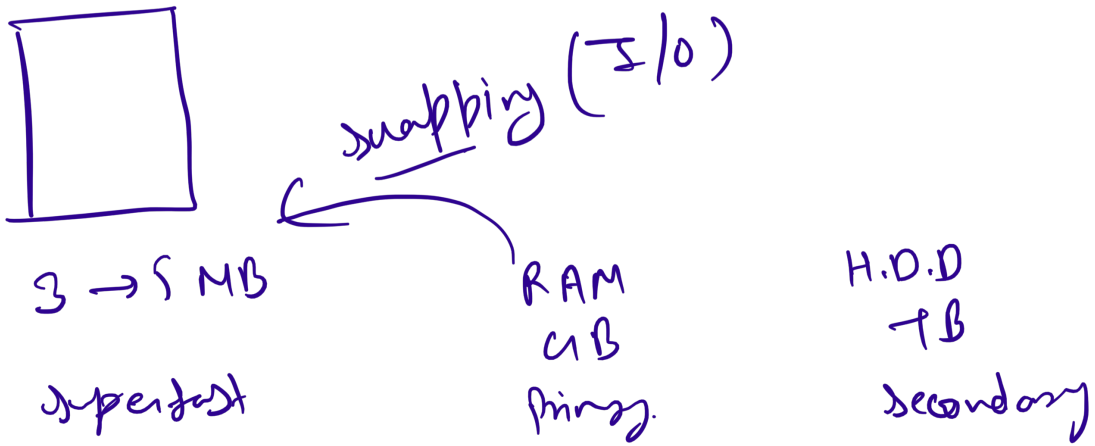
O(1).

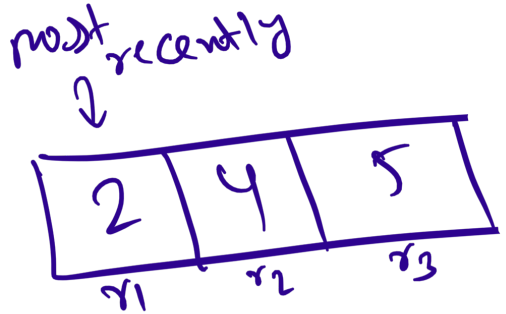
- Delete first
- Insert at Head
- Insert at a given position → O(n)

Least Recently used

Implement LRU Cache

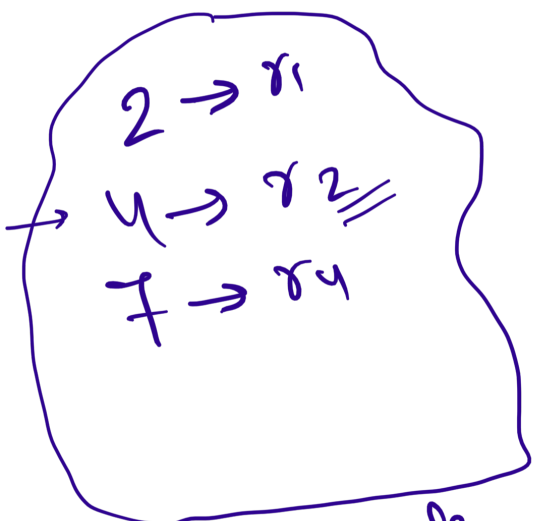
We are given the total possible page numbers that can be referred. We are also given a cache (or memory) size (The number of page frames that the cache can hold at a time). The LRU caching scheme is to remove the least recently used frame when the cache is full and a new page is referenced which is not there in the cache.



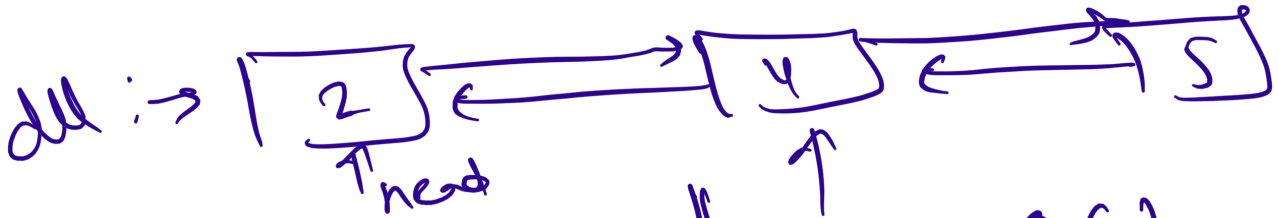


a[] = {1, 2, 3, 4, 5, 6, 7, 8}

refer(1) → miss
refer(4) → miss
refer(3) → miss
refer(5) → miss
refer(4) → hit
refer(2) → miss
refer(4) → hit



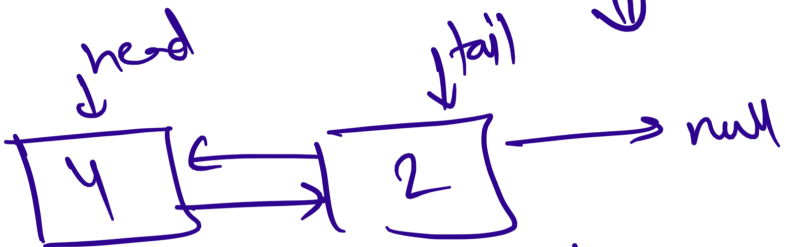
HashMap
<Data, Node>



① Delete using reference

O(1)

② insert at head



refer(7) → miss

```
refer ( int data) {
    isPresent = map.getKey(data);
    if ( isPresent ) {
        dll.remove ( data );
        dll.insertAtHead ( data );
    } else {
        Node newNode = new data
        dll.removeLast ( );
        map.remove ( lastKey);
        dll.insertAtHead ( data );
        map.put (data, data );
    }
```

Practice Problems

1. Clone a Linked List with next and random pointer.
2. Given a linked list A , reverse the order of all nodes at even positions.
    - `Input =  1 -> 2 -> 3 -> 4`
    - `Output = 1 -> 4 -> 3 -> 2`
3. https://www.interviewbit.com/courses/programming/linked-lists