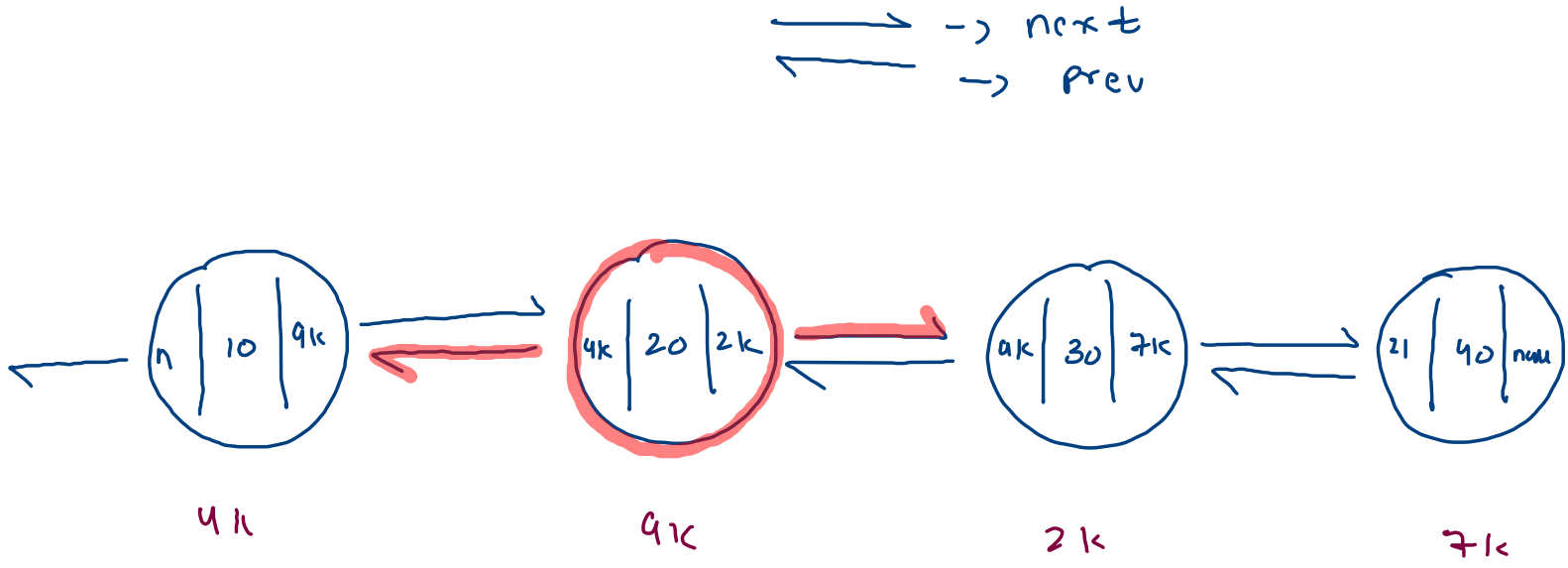
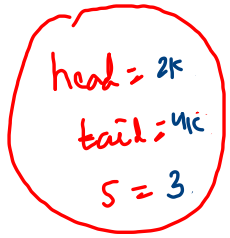
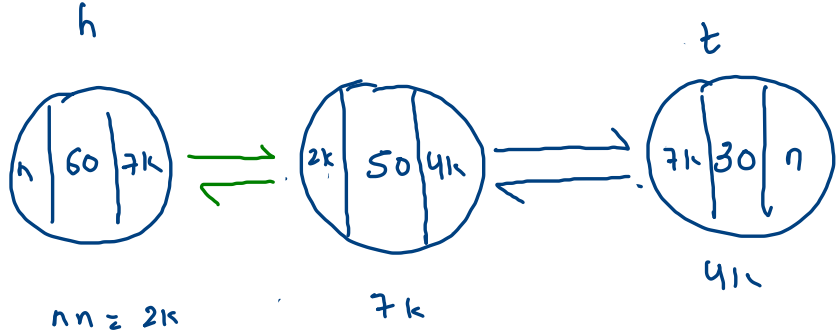
```
Node {
    int val;
    Node next;
    Node prev;
}

LL {
    Node head;
    Node tail;
    int size;
}
```



addfirst



DLL



dll.af.(30)

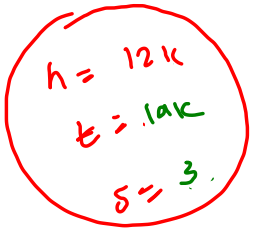
dll.af.(50)

dll.af(60)

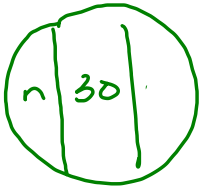
nn.next = head

head.prev = nn

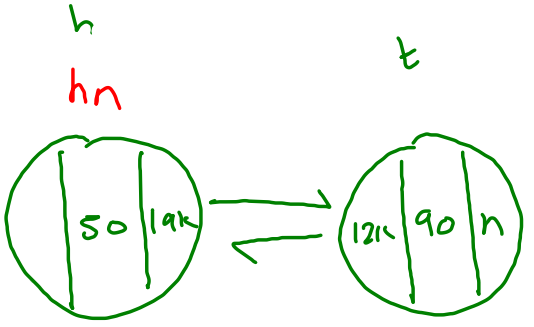
head = nn



DLL



4K



12K 19K

DLL.aL(30)

dll.aL(50)

dll.aL(90)

dll.rF()

size = 1

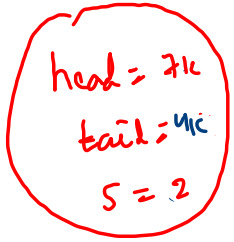
h = t = null

h.next = null

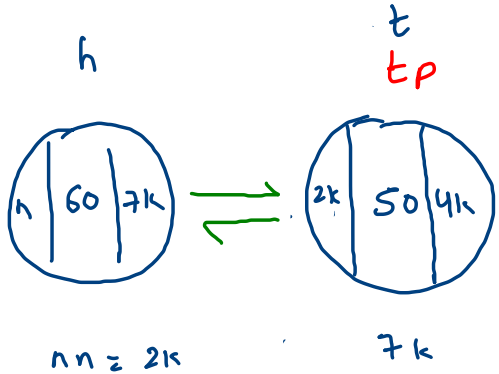
hn.prev = null

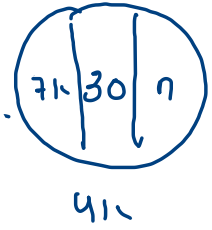
h = hn

} size ≥ 2



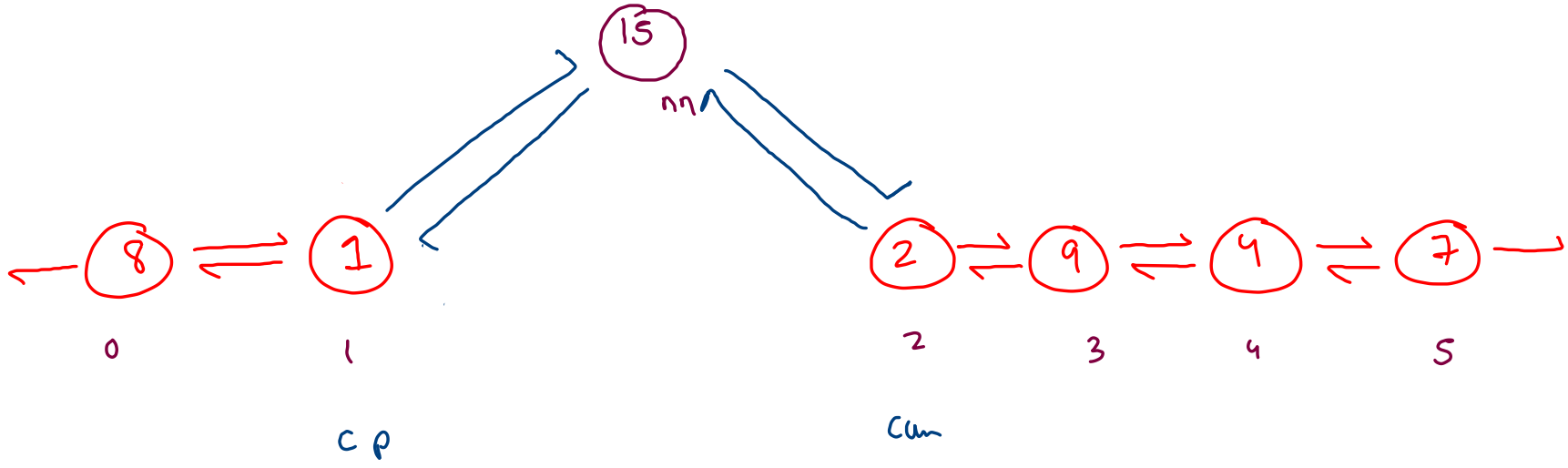
DLL





DLL.xL()

O(1)



addAt (2, 15)

idx, val

idx == 0 -> ar

idx == size -> al

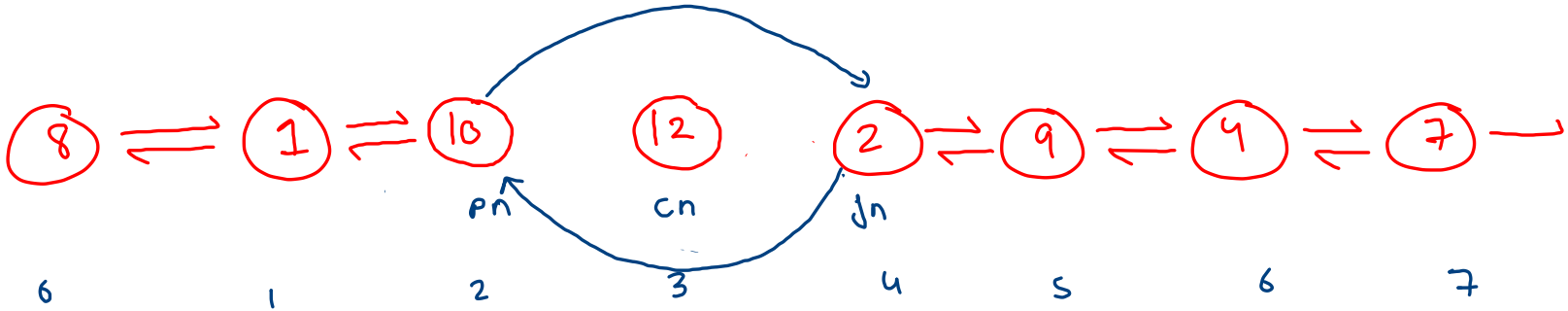
invalid idx : idx < 0 || idx > size

cp.next = nn

nn.prev = cp

cur.prev = nn

nn.next = cur

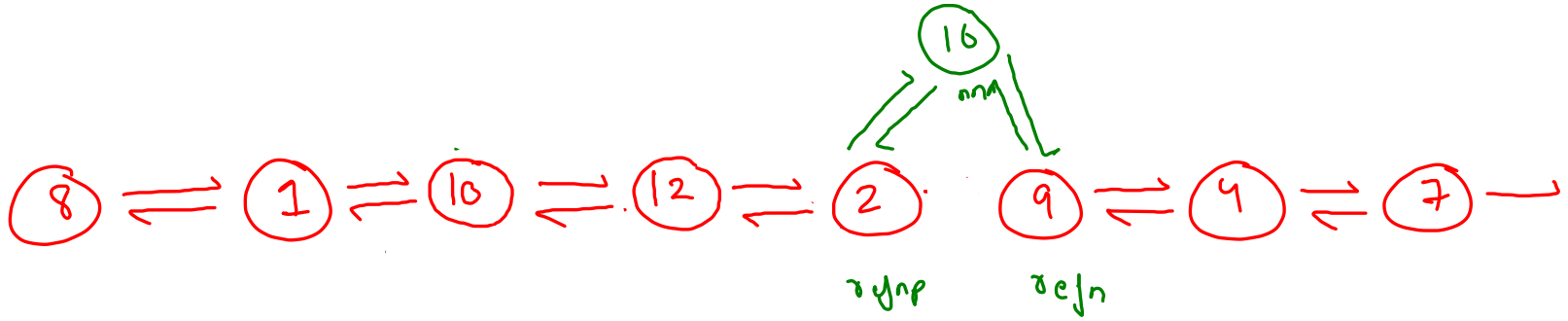


cn = gnA(idx)

pn.next = jn

jn.prev = pn

cn.next = cn.prev = null




```
Node nn = new Node(data);
Node rfnp = refNode.prev; //reference node prev'

rfnp.next = nn;
nn.prev = rfnp;
nn.next = refNode;
refNode.prev = nn;

size++;
```

addBefore(9, 16)

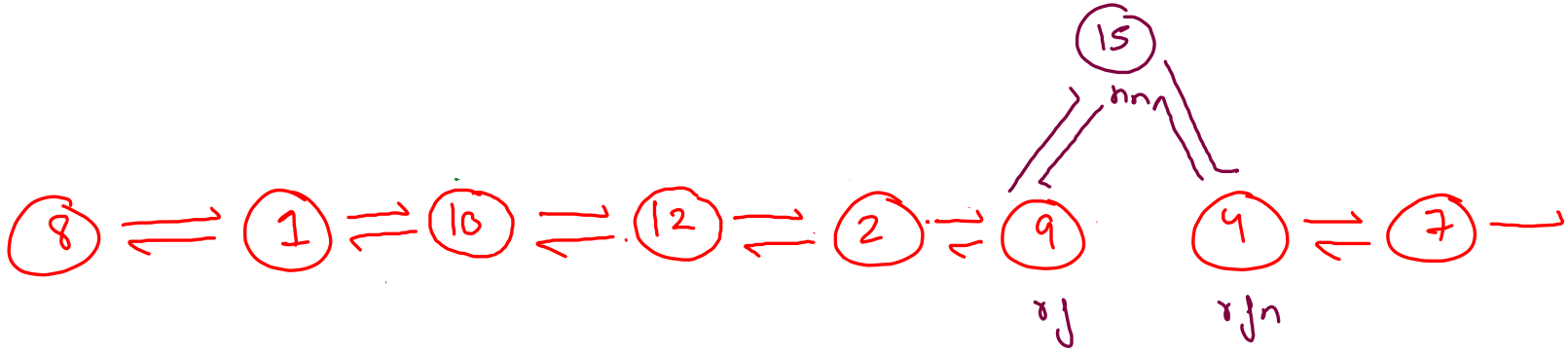
rn == head {

addFirst();

}

nn != head

- rnp.next = nn
- nn.prev = rnp
- nn.next = rn
- rn.prev = nn

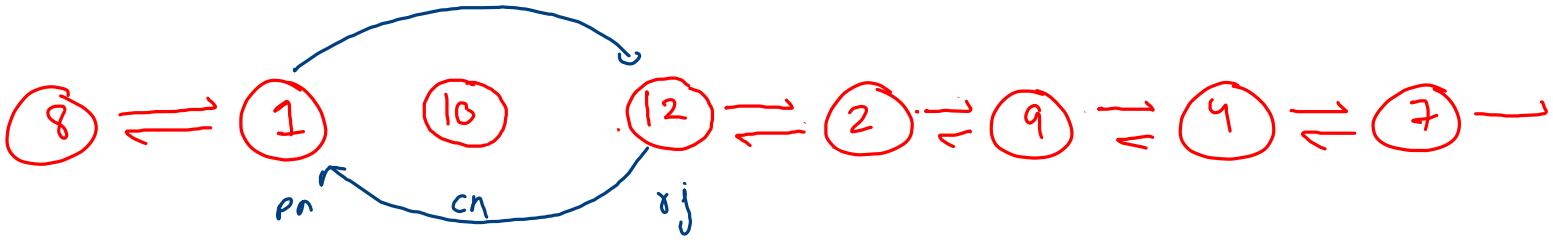


addAfter(9, 15)

```
if (rj == tail) {
    AddLast();
}
```

```
{ rj.next = nn
  nn.prev = rj
  nn.next = rjn
  rjn.prev = nn
```

rj != tail

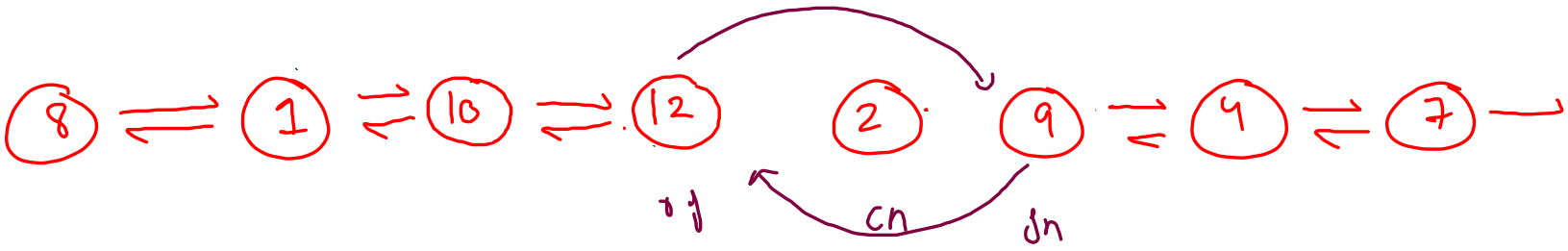



```
if (rj == head) {
    // do nothing
}
else if (rj == head.next) {
    removeFirst();
}
```

```
pn.next = rj
rj.prev = pn
cn.next = cn.prev = null
```

remove before (12)

rj



removeAfter (12)

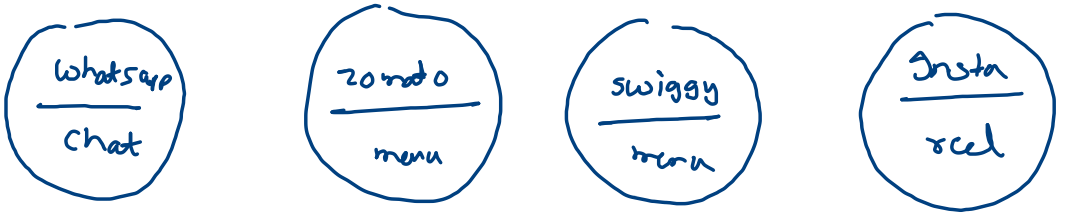
```
if(rj == tail) {
    // nothing
}
if( rj == tail.prev){
    removeLast();
}
```

```
rj.next = dn
dn.prev = rj
cn.next = cn.prev = null
```

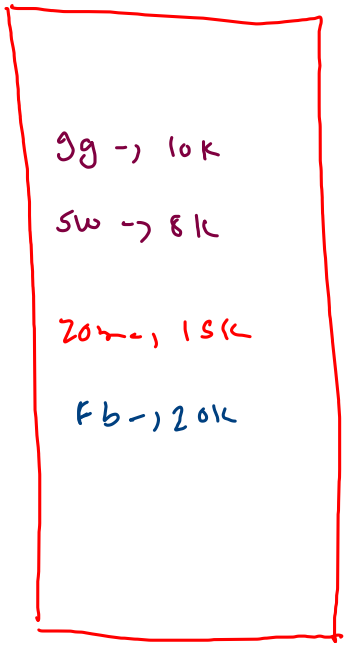
LRU cache

least recently used

limit = 4

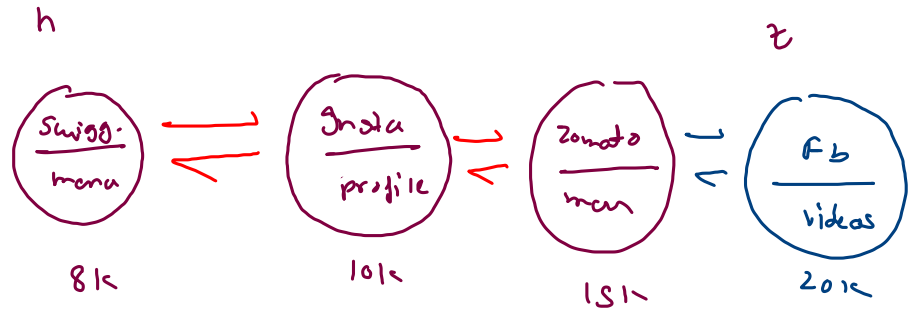


limit = 4



Key vs val

appname reference



already present [removeNode(10K)
AddLastNode(10K)

new app [addLastNode()
hm.size() > limit {
removeFirst();
}

Node {
appname
app state
Node prev
Node next
}

Input

```
["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]
[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]
```

Output

```
[null, null, null, 1, null, -1, null, -1, 3, 4]
```

[3,7]

put put get get

[9,15) [3] [4]

limit = 2