script

0 1 2 3 4

0 — 1 — 2 — 3 — 4 → x

kthFromLast

4 3 2 1 0

↑

* 0

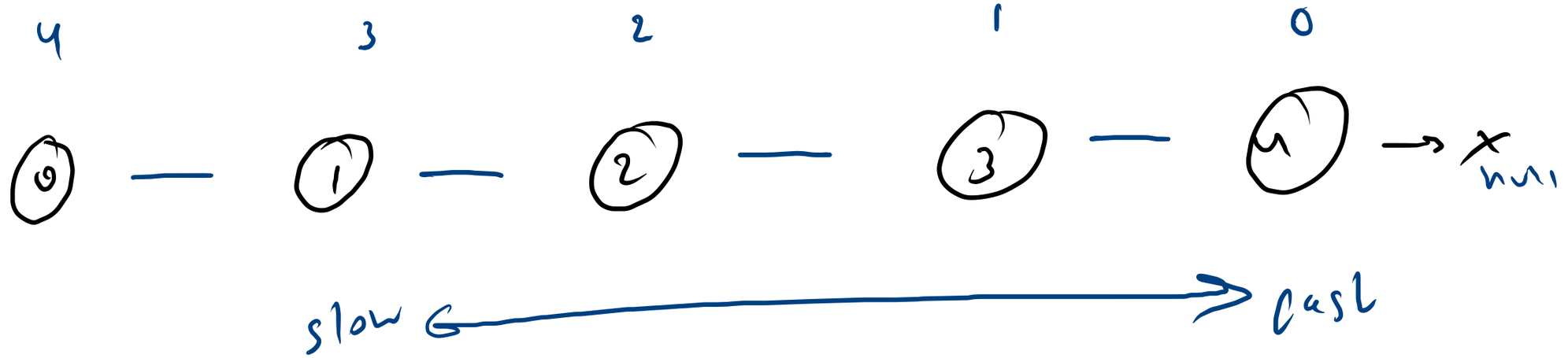
kthFromLast(0) → 4

3 → 1

head 0

tail 4

size 5 O(1)



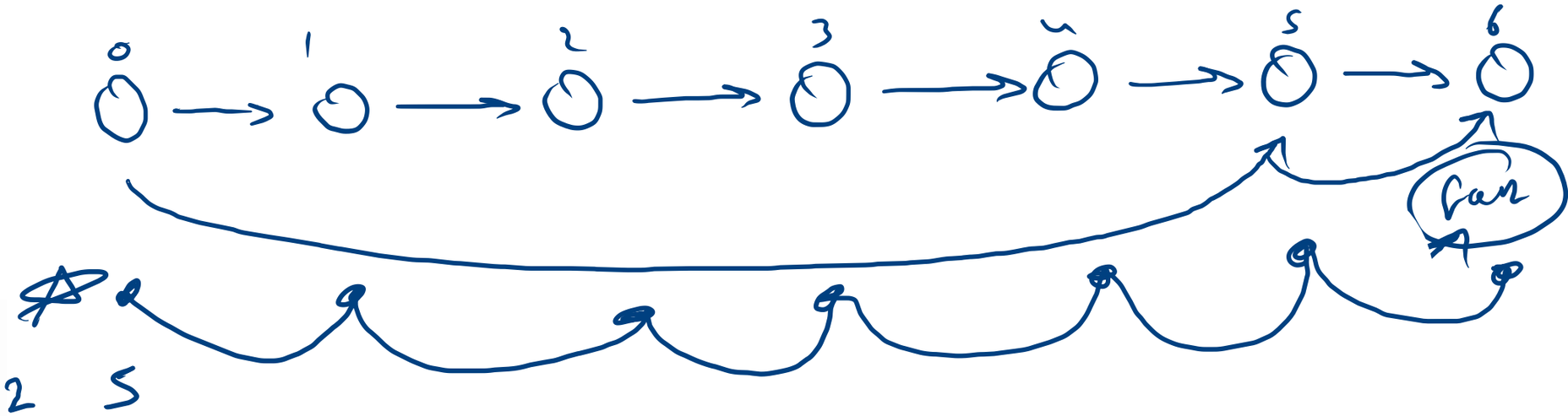
idx = 3

fast = head

[Fast move idx times

slow = head

slow, fast move




```
Node fast = head;
for(int i=0;i<k;i++){
    fast = fast.next;
}

Node slow = head;
while(fast != tail){
    fast = fast.next;
    slow = slow.next;
}

return slow.data;
```

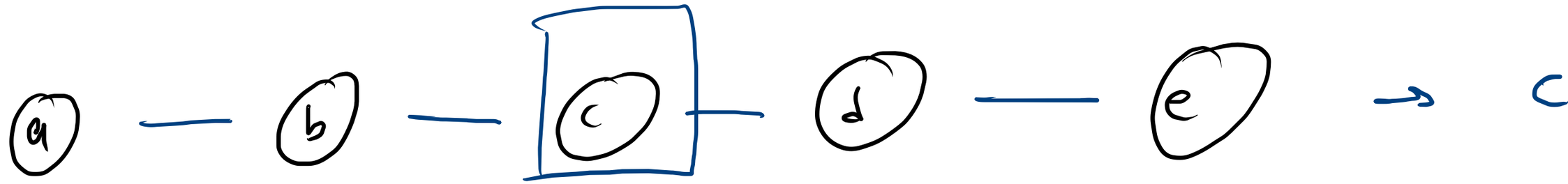
k

k

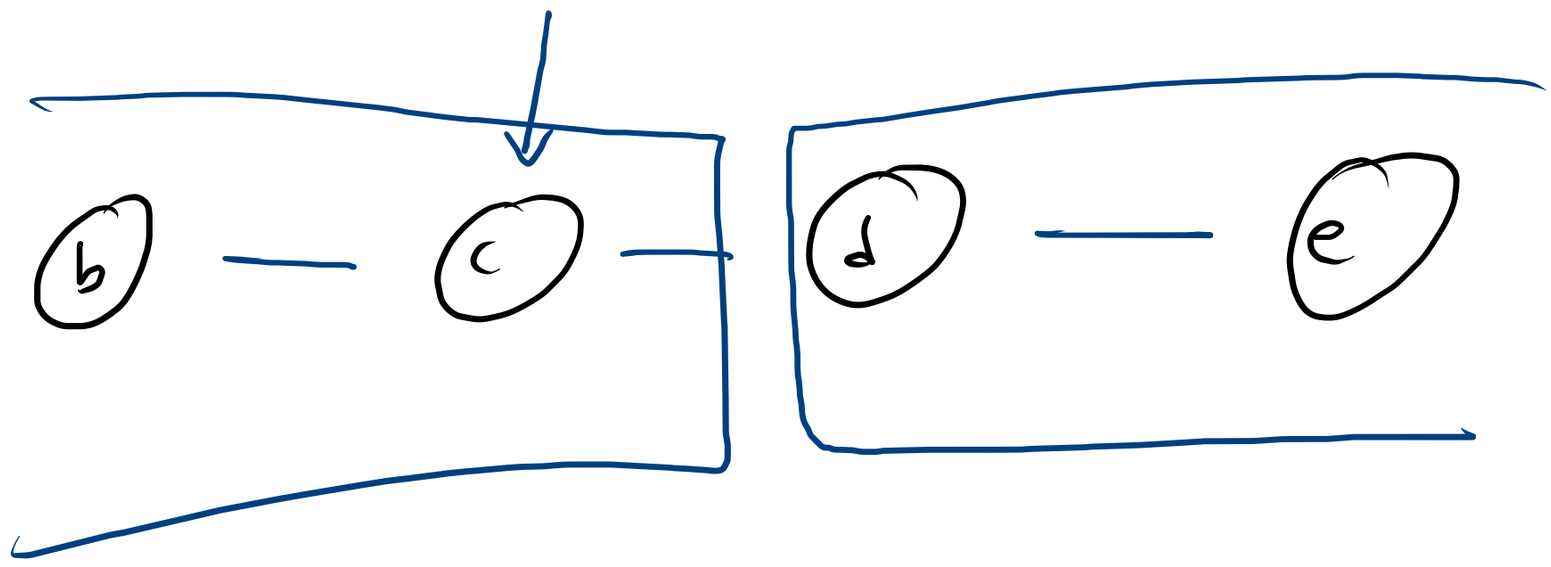
idx = 2, 3, 5

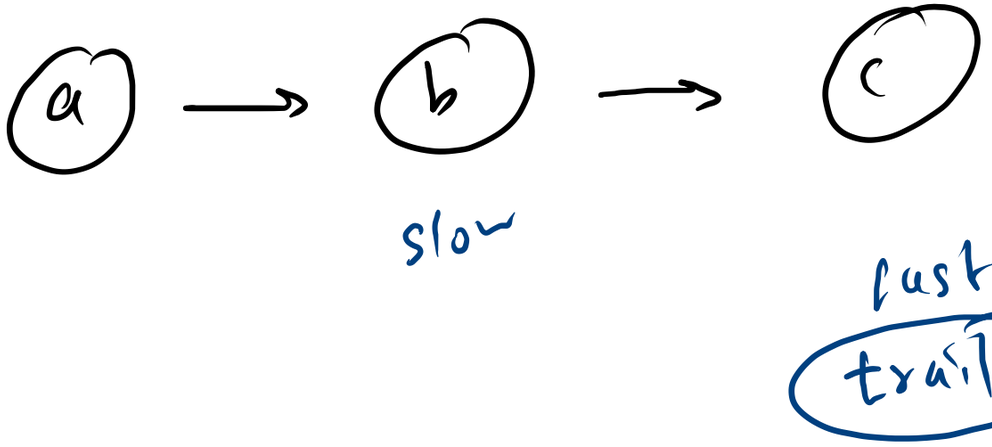
★ O(k)

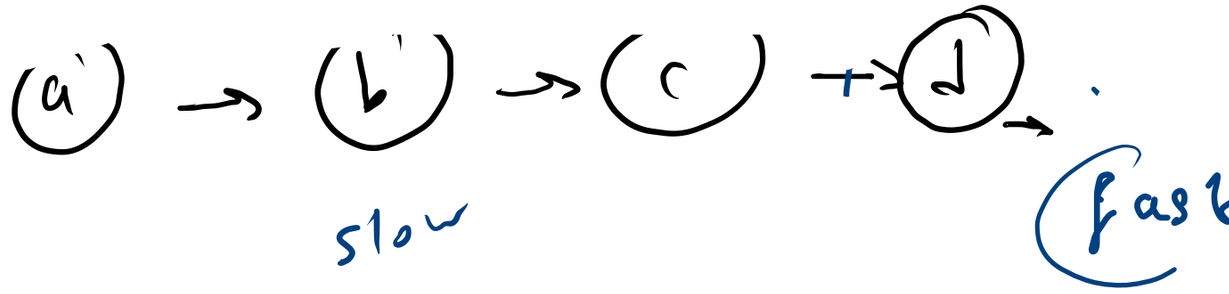
old



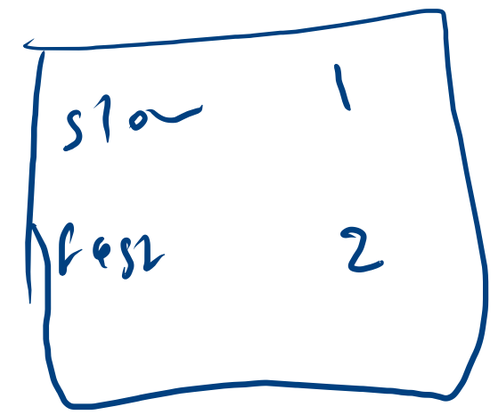
even

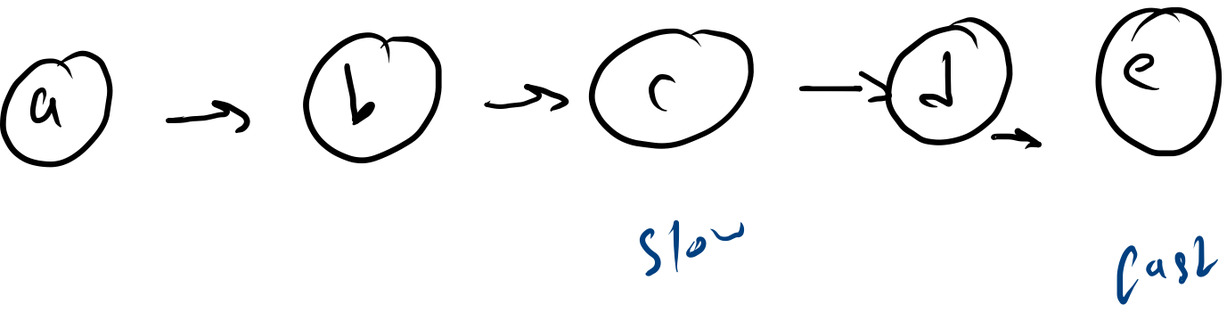







```
while ( fast != last )
    fast = fast.next.next;
    if( fast != null)
        slow = slow.next;
}
```



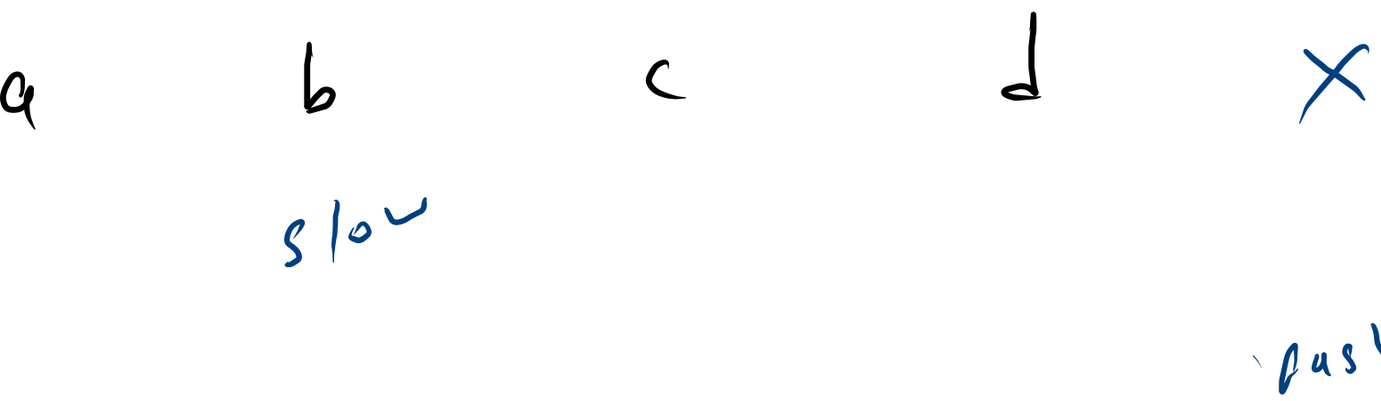


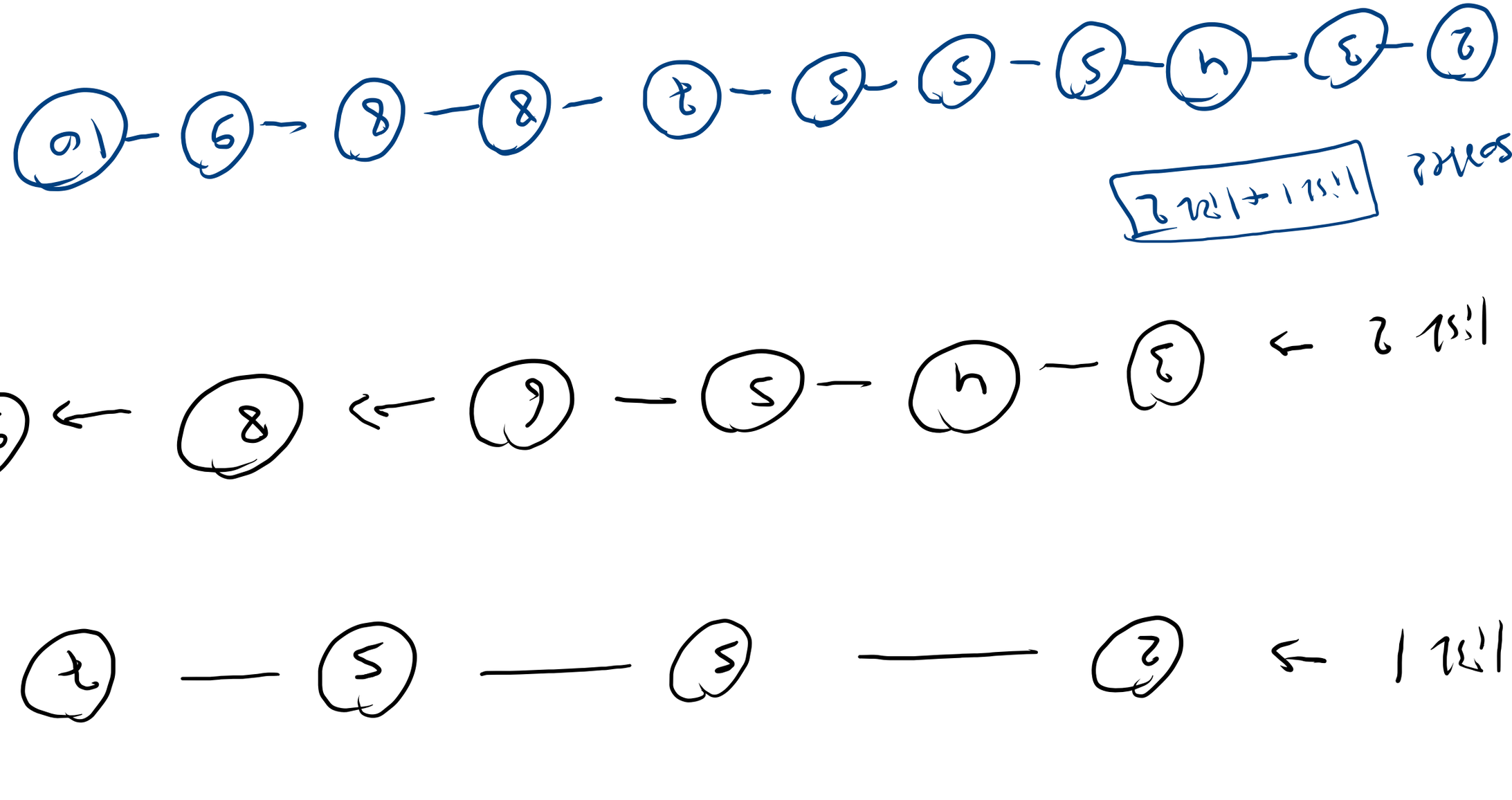

```
while ( fast != last && fast.next != last )
    fast = fast.next.next;
    if ( fast != null )
        slow = slow.next;
}
```

even

odd

even →





A

list 1 → (2) —— (5) —— (5) — (7) — (8)

B

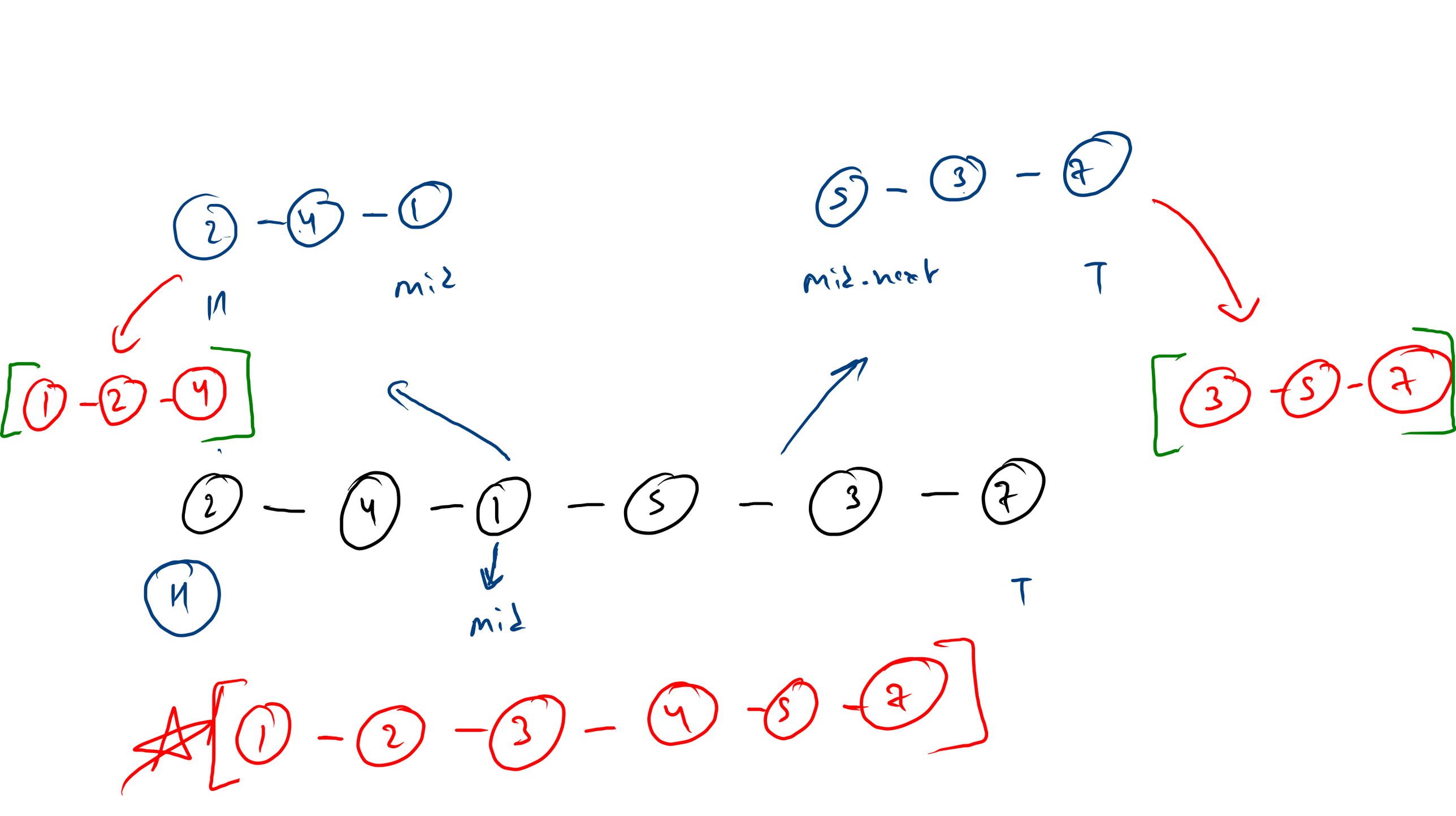
list 2 → (3) — (4) — (5) — (6) → (8) → (9) → (10)

merge 2 → (2) — (3) — (4) — (5) — (5) — (5) — (6) → (7) → (8) — (9) → (10)

addLast

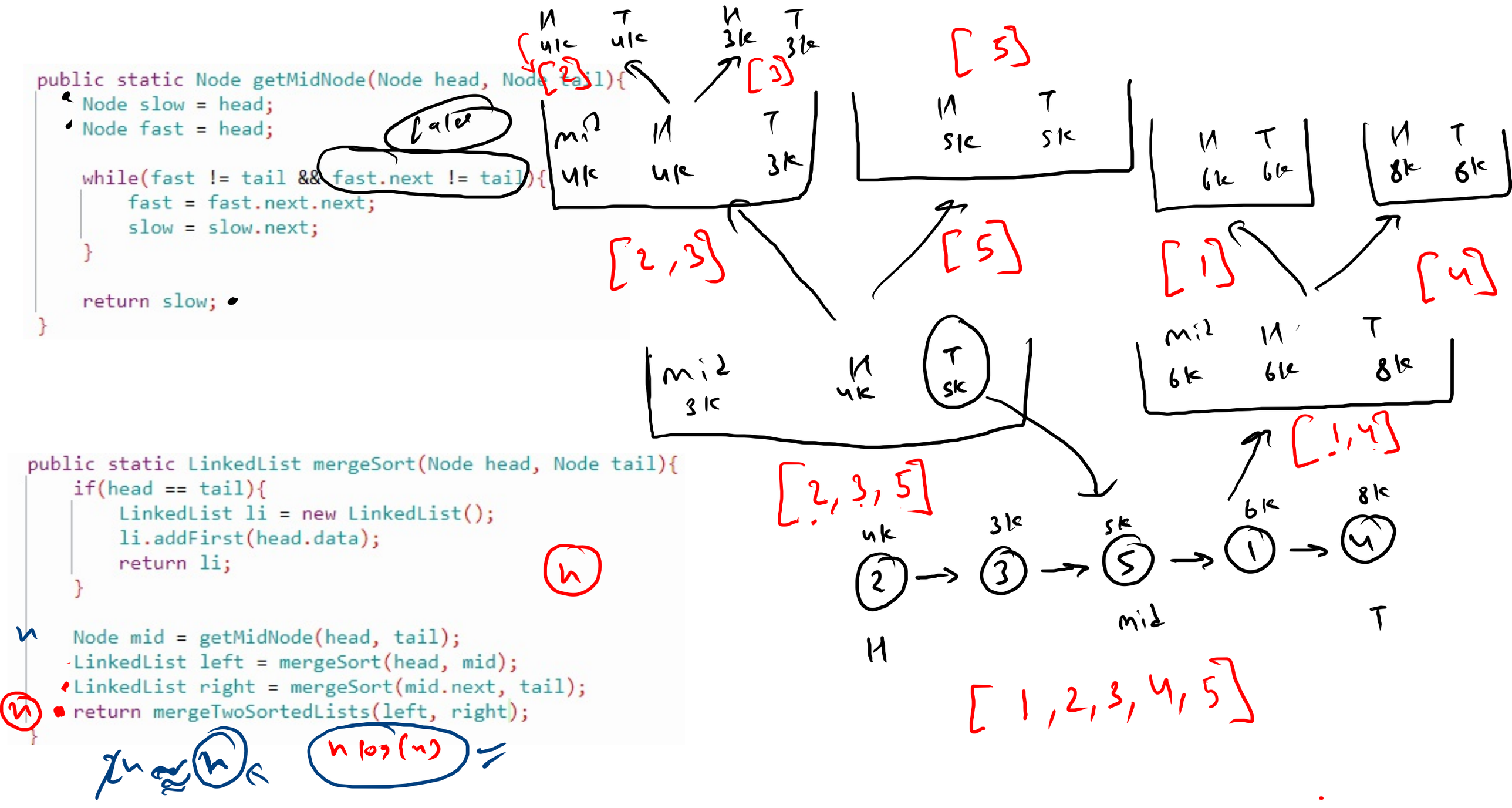
update 2

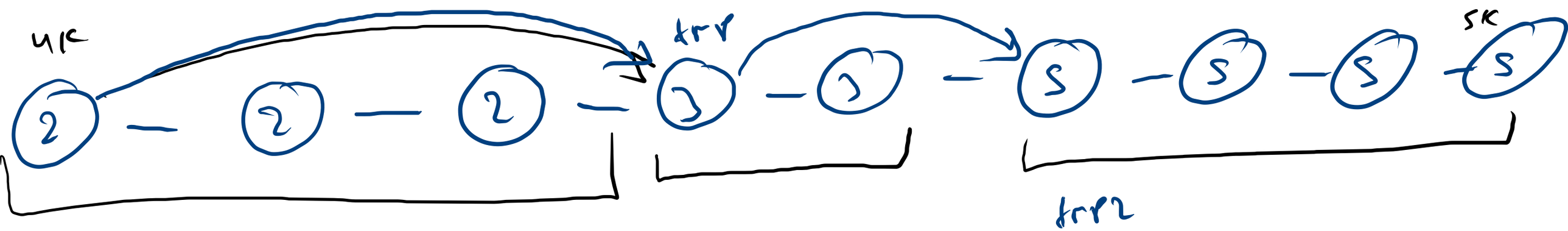
tail X
size 0

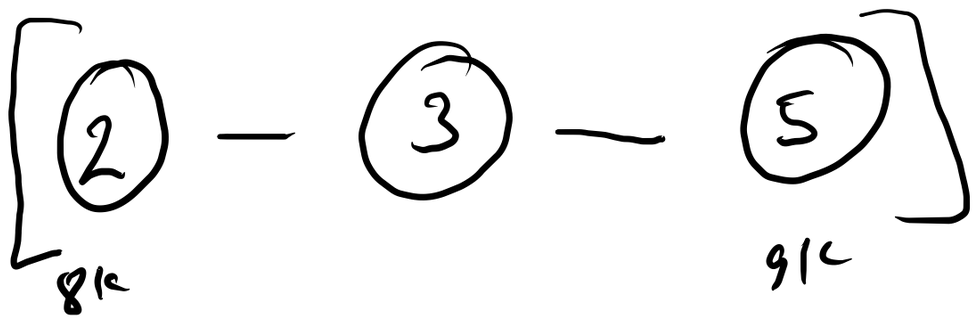

2 - 4 - 1
H
mid
5 - 3 - 7
mid.next
T
[1 - 2 - 4]
[3 - 5 - 7]
2 - 4 - 1 - 5 - 3 - 7
H
mid
T
[1 - 2 - 3 - 4 - 5 - 7]

```java
public static Node getMidNode(Node head, Node tail){
    Node slow = head;
    Node fast = head;

    while(fast != tail && fast.next != tail){
        fast = fast.next.next;
        slow = slow.next;
    }

    return slow;
}
```

```java
public static LinkedList mergeSort(Node head, Node tail){
    if(head == tail){
        LinkedList li = new LinkedList();
        li.addFirst(head.data);
        return li;
    }

    Node mid = getMidNode(head, tail);
    LinkedList left = mergeSort(head, mid);
    LinkedList right = mergeSort(mid.next, tail);
    return mergeTwoSortedLists(left, right);
}
```




4K
2
2
2
trv
1
1
5
5
5
5K
5
trv2

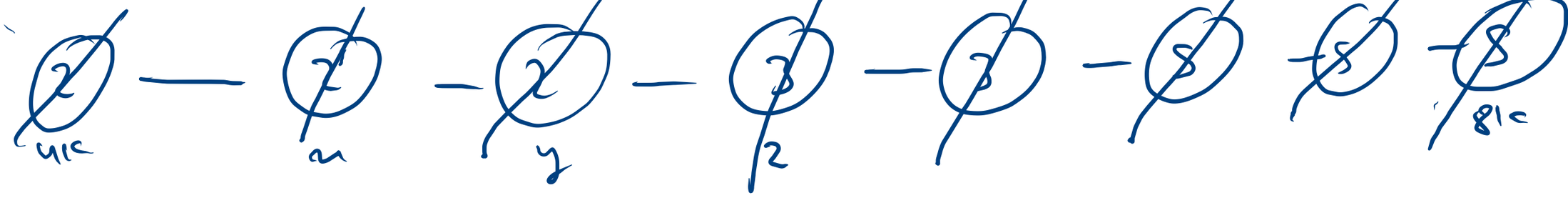

2
3
5
8K
9K


head
8K
tail
9K
size
3
head →
4K
tail →
5K
size →
9

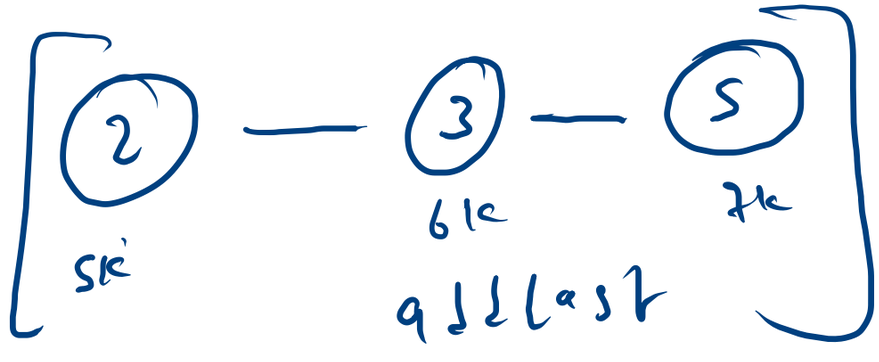
node

reuse

this



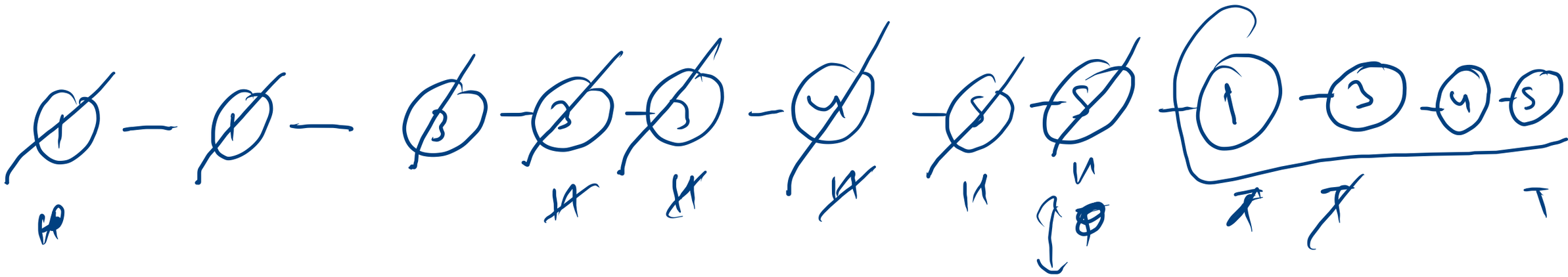
head	5lc
tail	7lc
size	3



nl →

head	5lc
tail	~~5lc~~ ~~6lc~~
size	~~0~~ ~~1~~ ~~2~~ 3

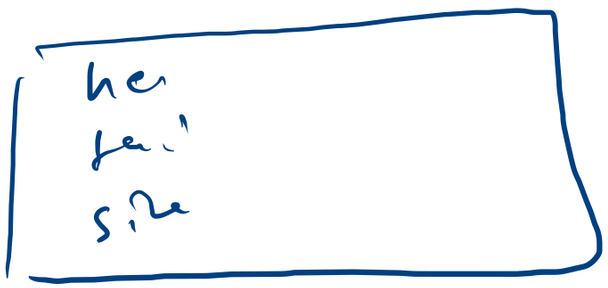
objs

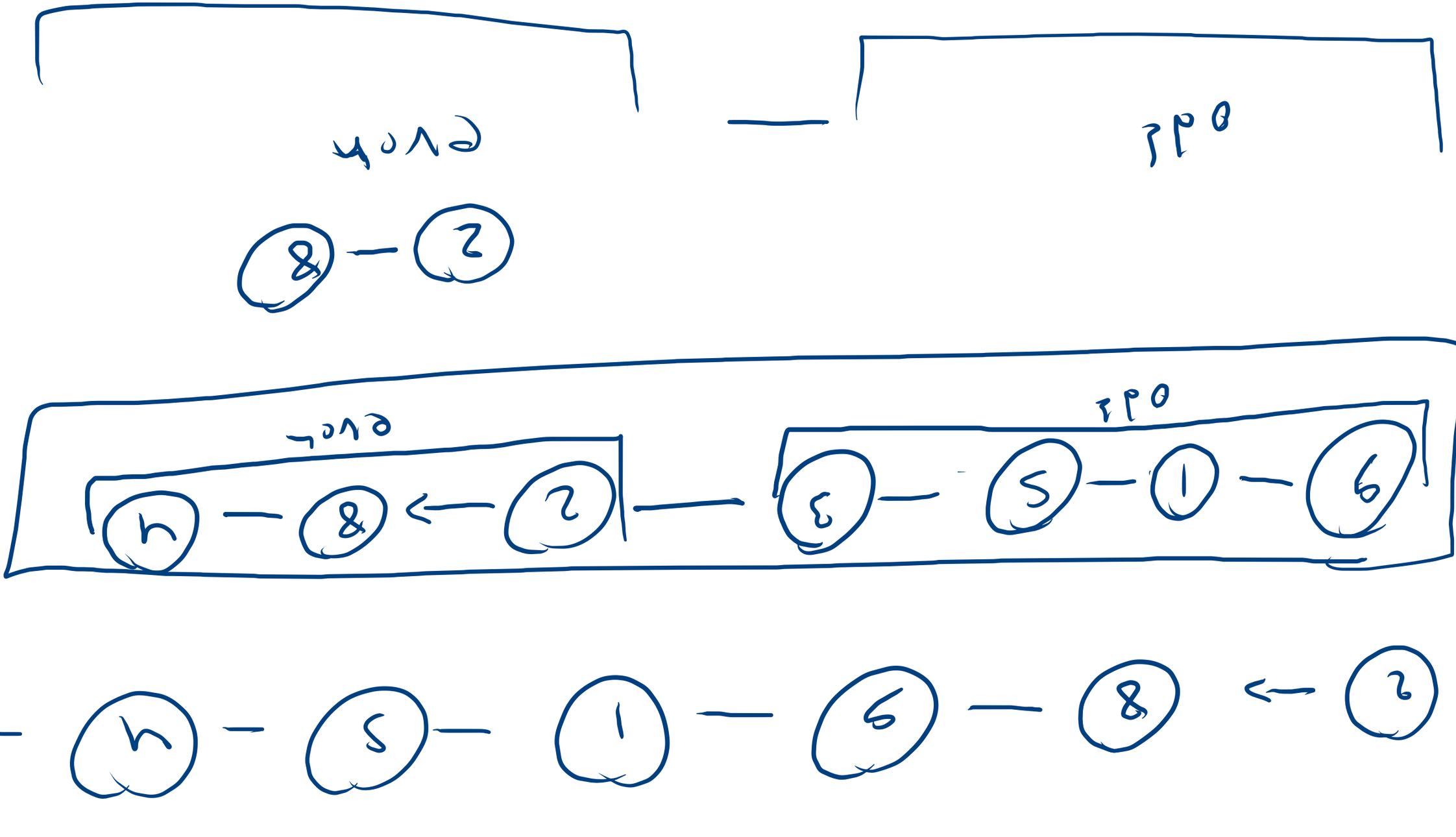


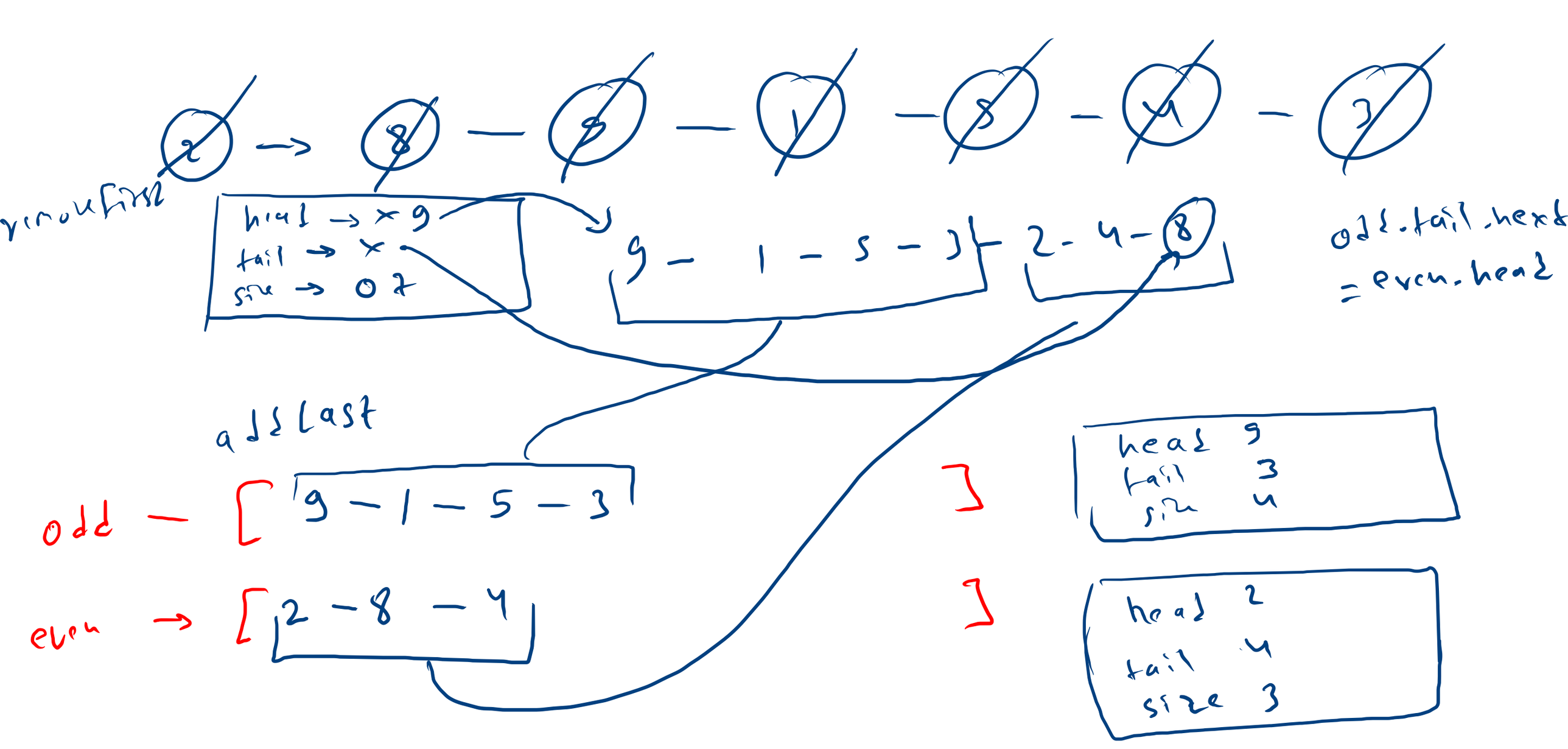
OT

removeAll O(1)

allLast O(1)






head → x 9
tail → x
size → 0 4
9 - 1 - 5 - 3
2 - 4 - 8
odd.tail.next
= even.head
addLast
odd — [9 - 1 - 5 - 3]
even → [2 - 8 - 4]
head 9
tail 3
size 4
head 2
tail 4
size 3

```java
LinkedList odd = new LinkedList();
LinkedList even = new LinkedList();

while(size > 0){
    int val = head.data;
    removeFirst();
    if(val %2 ==0){
        even.addLast(val);
    }else{
        odd.addLast(val);
    }
}

odd.tail.next = even.head;
head = odd.head;
tail = even.tail;
size = odd.size() + even.size();
```