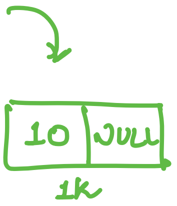
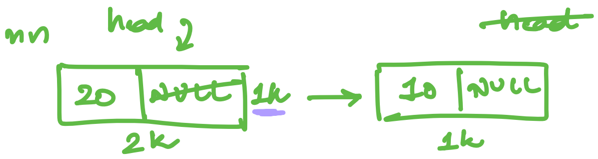
Add first

additions: ll empty, 1st node add

add first (10)
add first (20)

head = ~~NULL~~ 1k

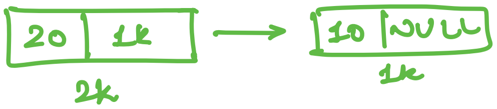




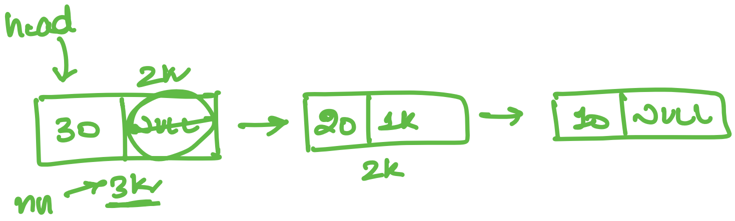
nn → next = head
1k

head = nn
2k

head = 2k



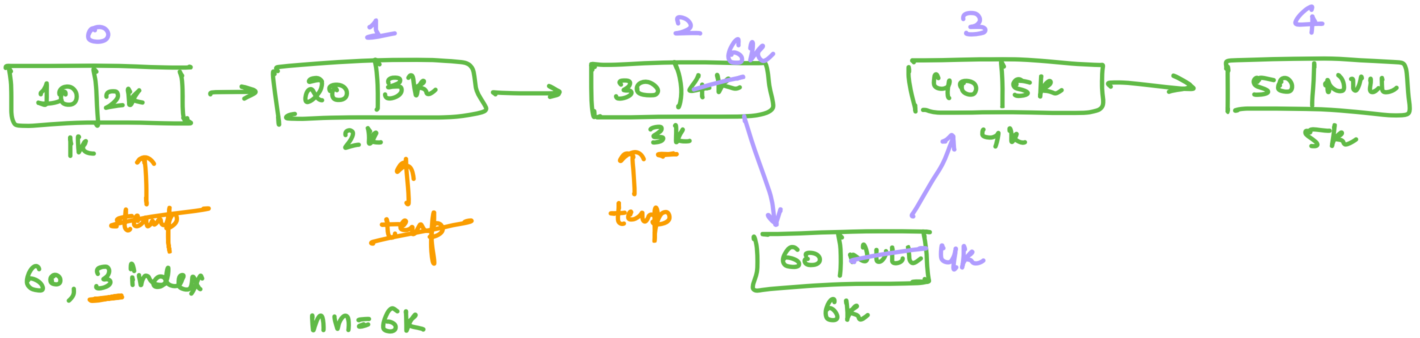
add first (30)



nn → next = head
2k

head = nn
3k

Add At



nn = 6K

nn → next = temp → next

temp → next = nn

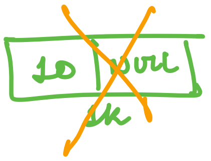
6k → next = 4k

nn → next = temp → next
4k

Remove Last

Border Case: ll 1 node, removing that one node

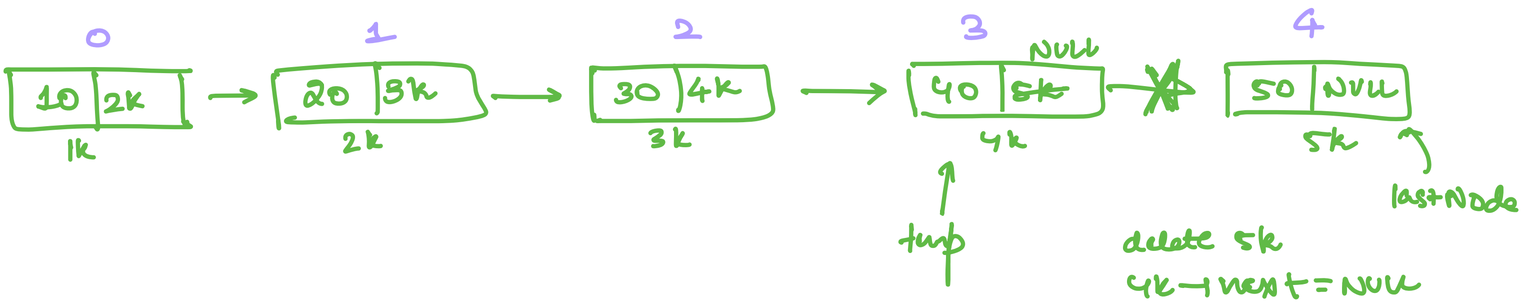
② head = ~~1k~~ NULL



① delete ma
1k

4k→n→n=NULL

5k→ next= NULL

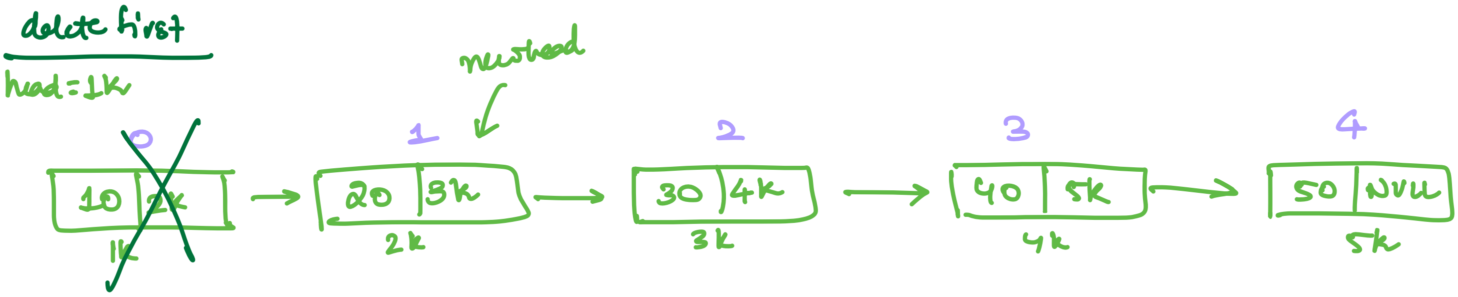


delete 5k

4k→next=NULL

delete first

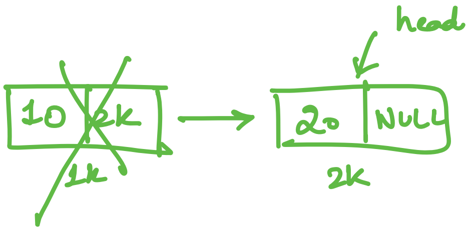
head=1k



2k access

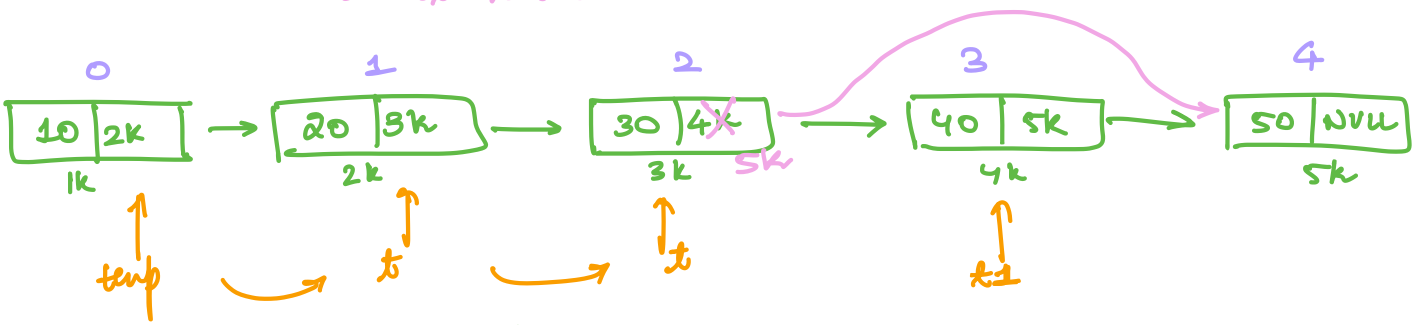
delete 1k

head = 2k



remove At

3 index remove



t access

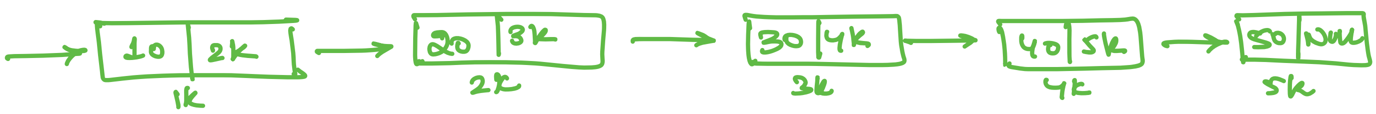
t1 access

$t \rightarrow next = \underbrace{t1 \rightarrow next}_{5k}$

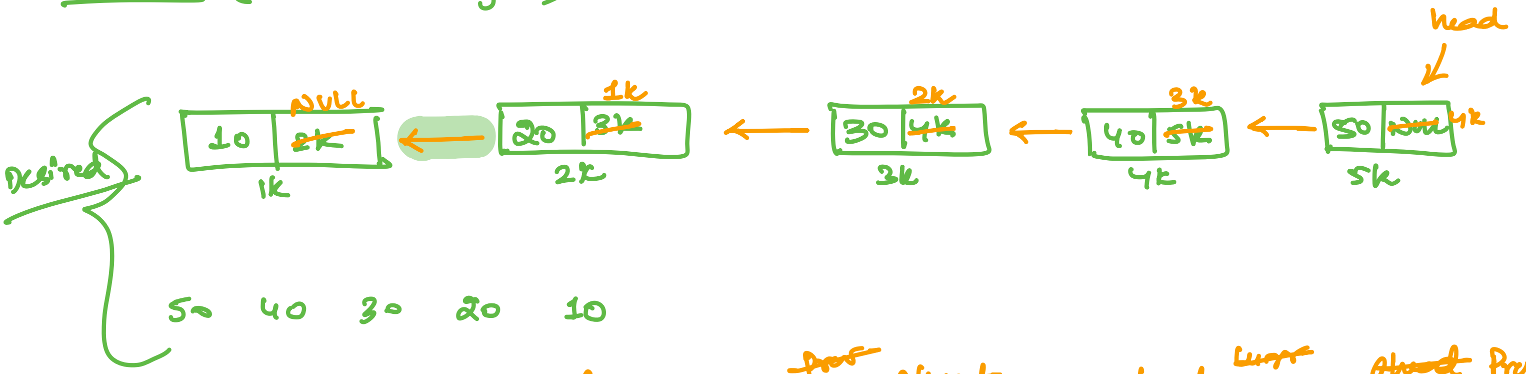
delete $\underline{4k}$

t1

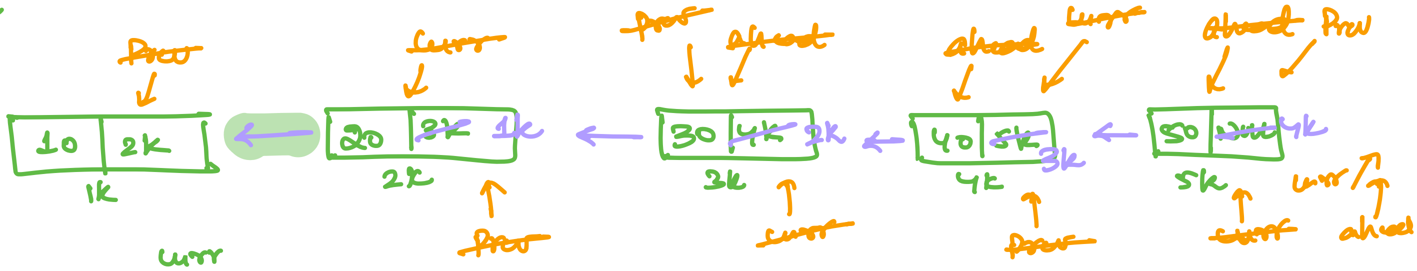
Reverse



Reverse (Pointers Change)



head = 1k



curr
ahead = 2k→next

2k→next = 1k
curr   Prev

prev = curr
2k

curr = ahead
3k

ahead = 3k→next
4k

3k→next = 2k
curr   Prev

prev = curr
3k

curr = ahead
4k

ahead = 4k→next

4k→next = 3k

prev = curr
4k

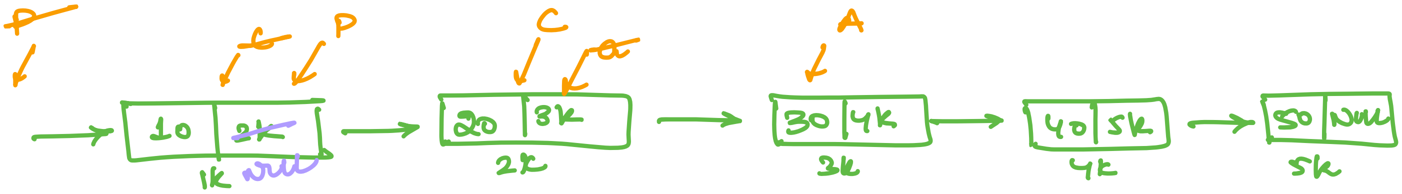
curr = ahead
5k

ahead = NULL

5k→next = 4k

prev = curr
5k

curr = ahead
NULL

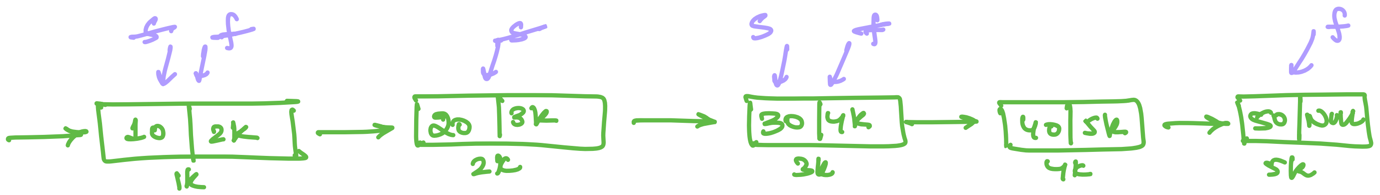


head = 1k

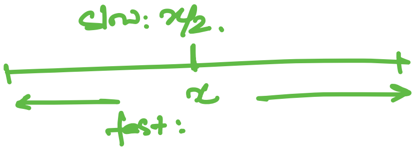
Prev = NULL;
curr = 1k

ahead = 2k
curr→next = prev
1k   null
P = C
C = A

Mid

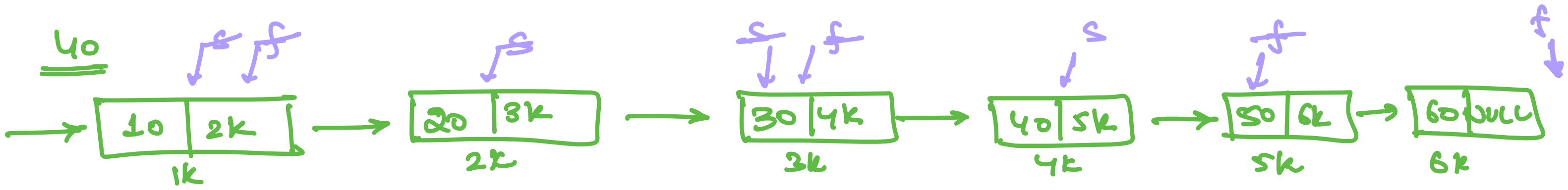


odd elements: fast → next == NULL
stop



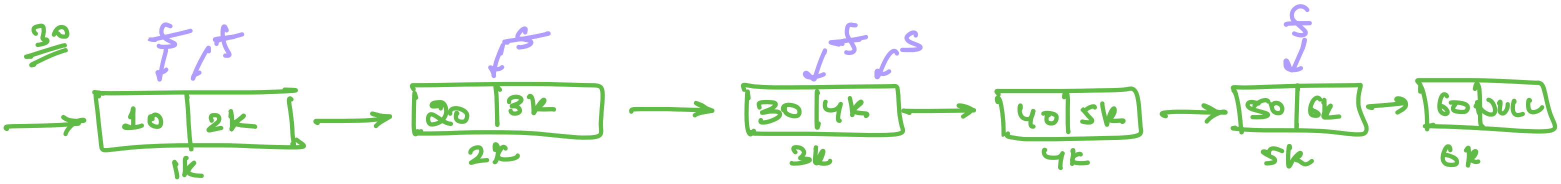
size() → 5
5/2 = 2
loop 2 times

40



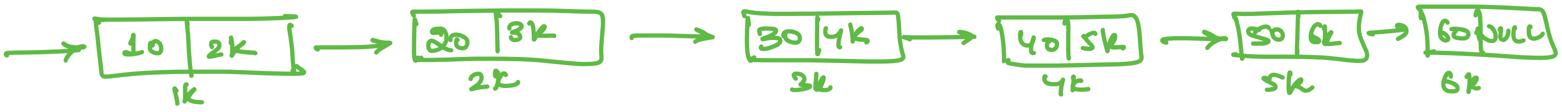
fast == NULL stop

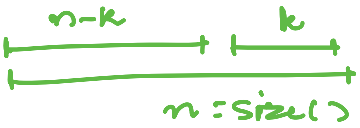
30



fast → next → next == NULL stop.

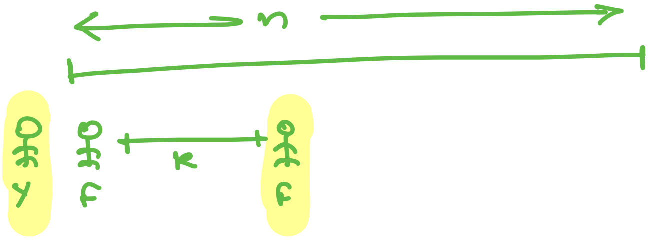
kth from last

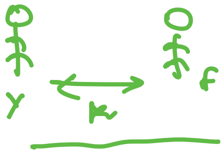




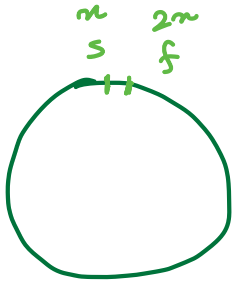
last (k+th): start (n-k)th

k=1	→	60
k=2	→	50
k=3	→	40
k=4	→	30
		⋮

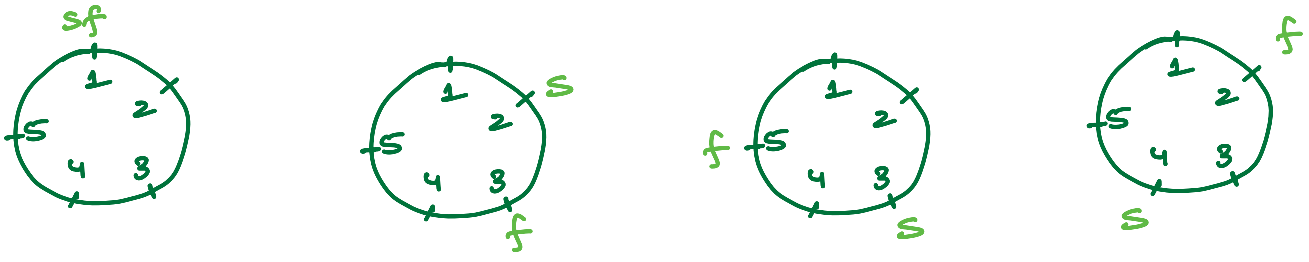


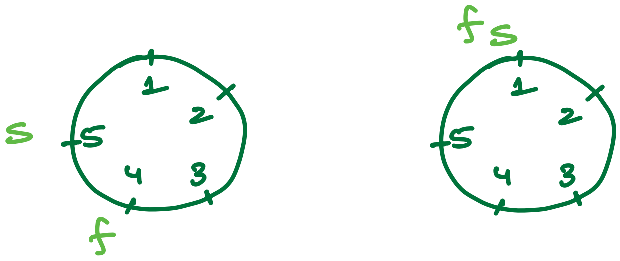


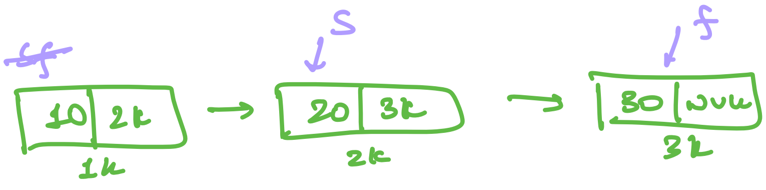
Detect loop in linked list



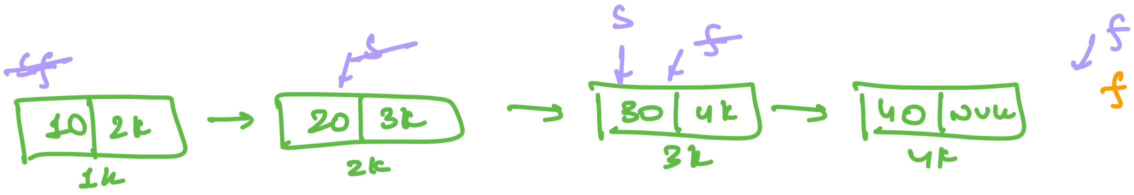
cycle: meet
no cycle: fast null.







f→n = NULL



f = NULL

Remove Loop → Video (Java)