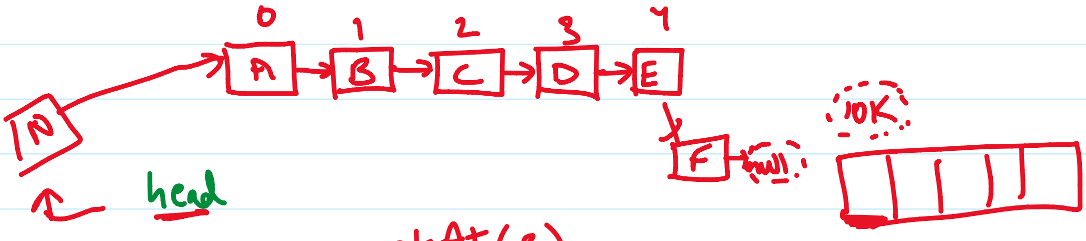
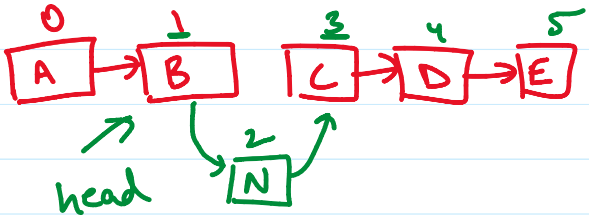
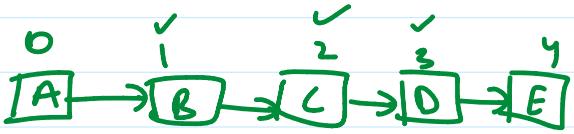
05 March 2023 09:48

LL



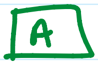
getAt(3)





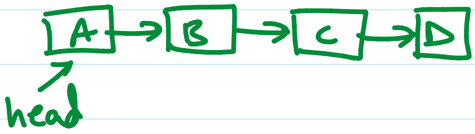
```
Node nn = new Node(data);
        Node prev = getNodeAt(idx - 1);
        Node curr = getNodeAt(idx);
        nn.next = curr;
        prev.next = nn;
```

idx=1

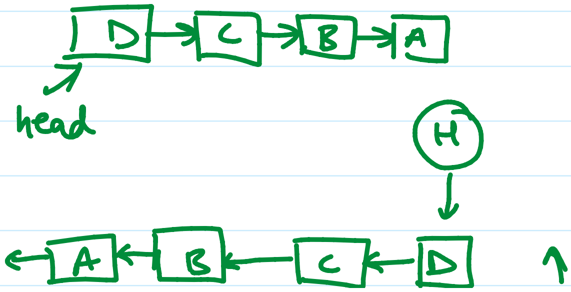


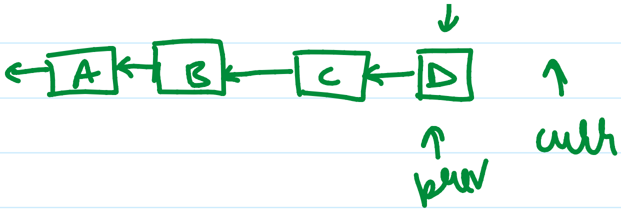
1) Reverse a LL

Iterative



addlast ✗

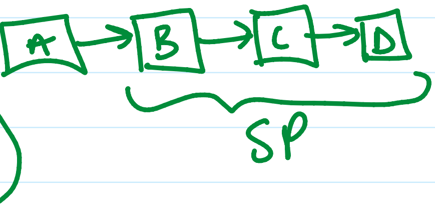




curr.next=p

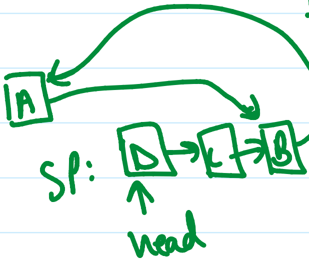
2) Reverse L

→ Recursion



Node rev(Node n){

//



A.next.next=A →

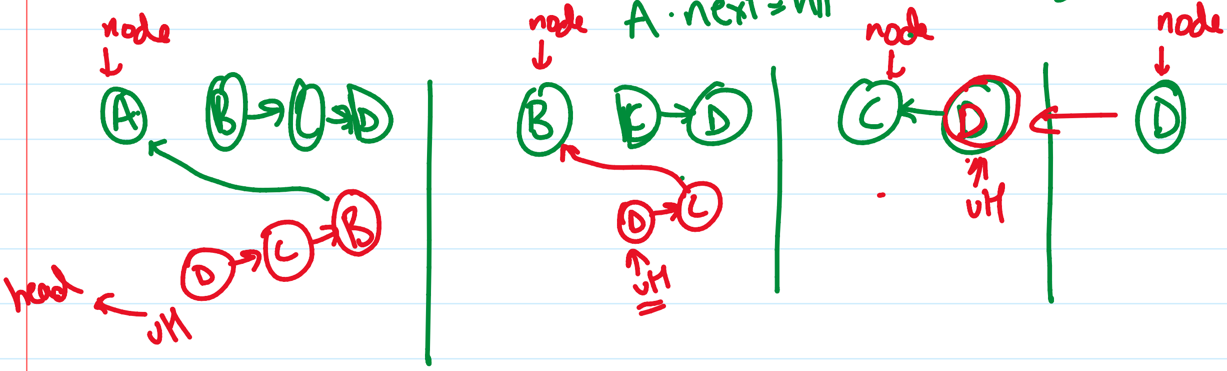
A.next=null →

}

B.C

↳ node.next =null

node=null

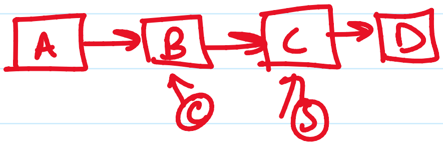


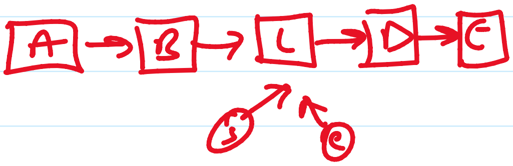
~~size~~, ~~getAt~~

→ Middle of a LL

M1) → get At

T(n) : O(N^2)

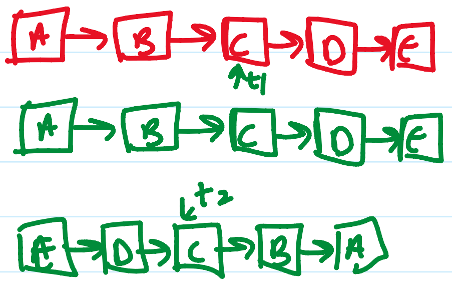




M2) O.G
R.LL (c.G)

$T(N) = O(N)$

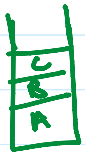
$S(N) = O(N)$

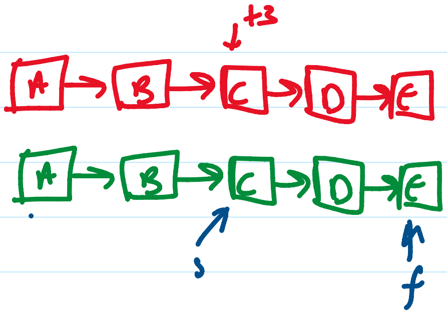


Copy +
>ReversedLL

M3) $T(N) = O(N)$

$S(N) = O(N)$

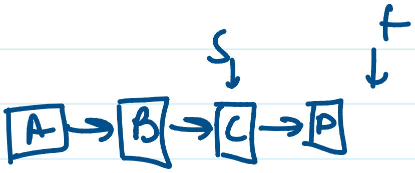


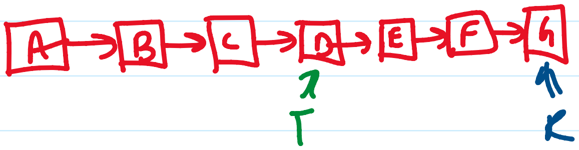


Copy
+
Stack

M4) $T(N) = O(N)$

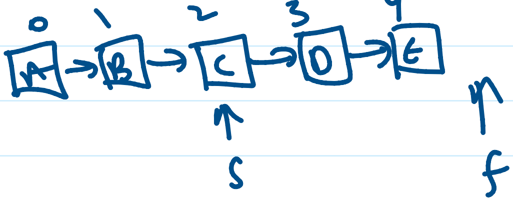
$S = O(1)$





2 ptr ✓
↳ s .next
↳ f .next.next

→ Kth from last



K=2

size x

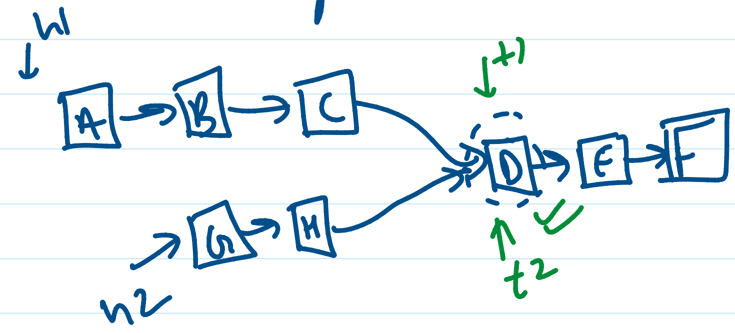
5-2

K=2

10->20->30

10 → 20 → 30 → 40 → 50 → 60

→ Intersection of LL



len(L1) = 6

len(L2) = 5

→ size1, 2

→ Headstart

→ compare