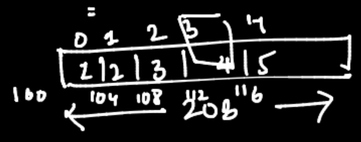
Linked List

Array → linear DS

contiguous ✓

int a[5] = {1, 2, 3, 4, 5}



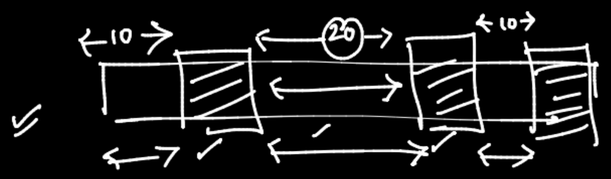
3rd index ⇒ 100 + 3(4) (addr)

= 112

a[3] = *(a+3)

max size of a vector / array is limited by the

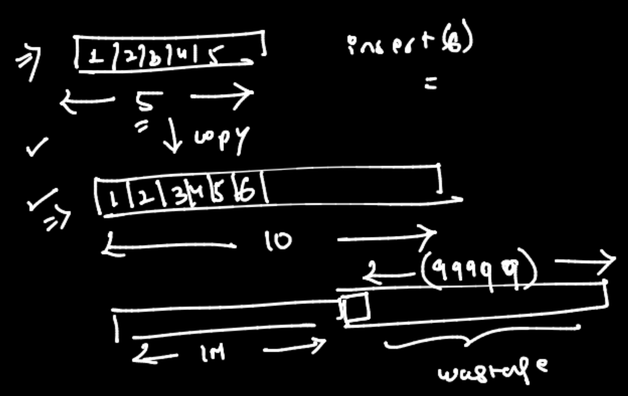
max. conti memory available

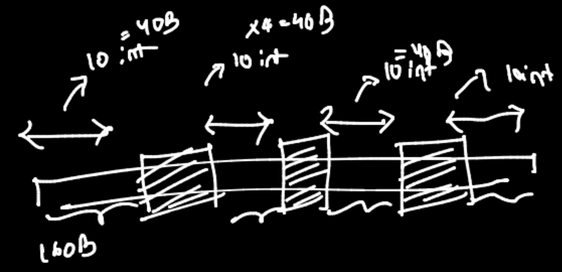


Vectors - dynamic arrays

resize → expensive

insert(_)
erase()



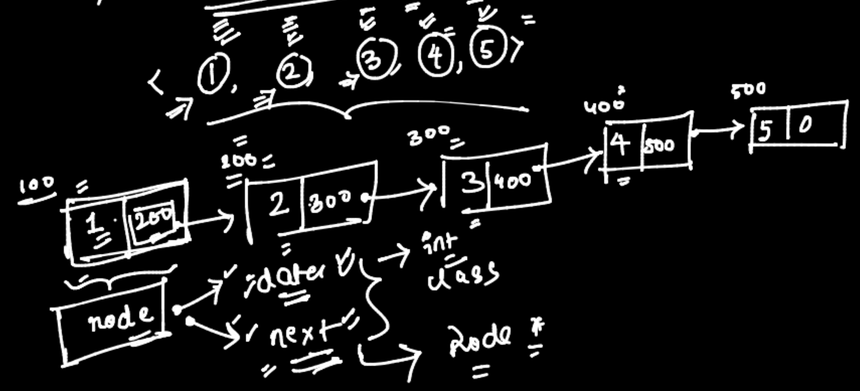


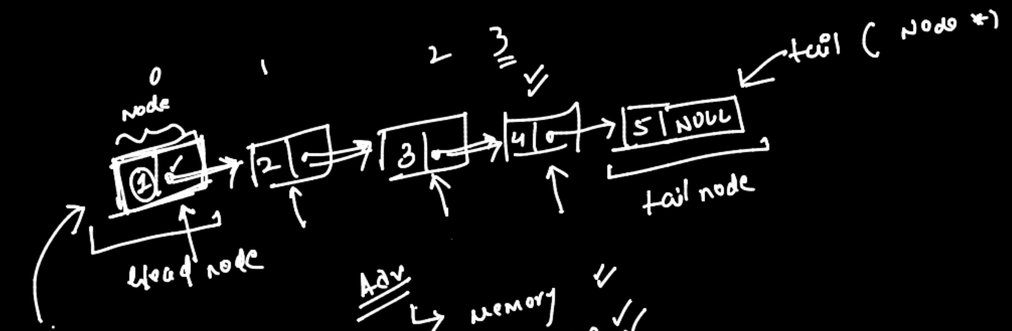
vector ⇒ 40 x 4 = 160 B

/ array

Linked List

- linear DS
- non-contiguous



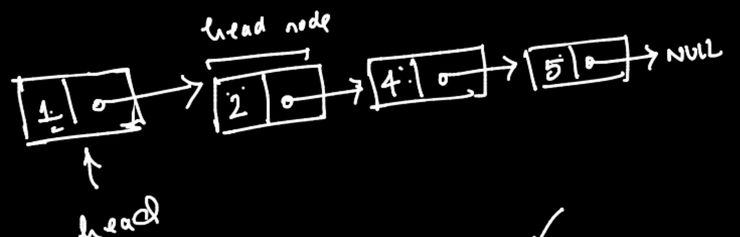


head → node *

Adv
- ↳ memory ✓
  - ↳ no resize ✓
  - ↳ no frag. issue ✓

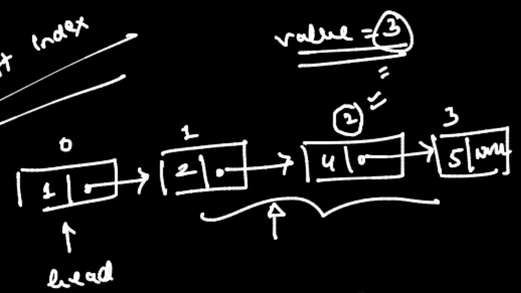
Disadv
- Random access is no longer const.

1. Insert
   - at the beginning (at head)
   - at the end (at tail)
   - at an index

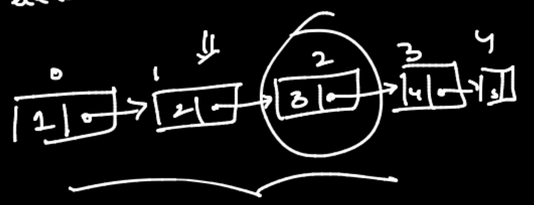


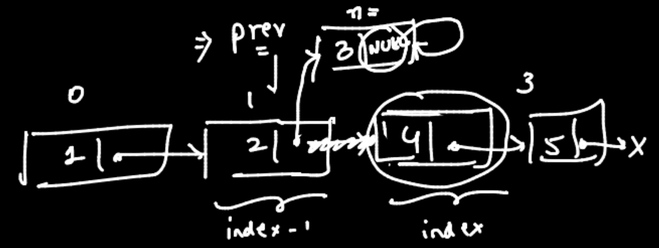
1. Create node with val. 1 ✓
2. assign its next ptr to point to head of LL ✓
3. reassign head to new node ✓

2. Insert At index



⇒ Insert(3, 2)



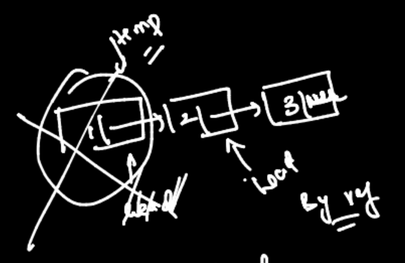


insert (3, at 2) index = 2 ⇒

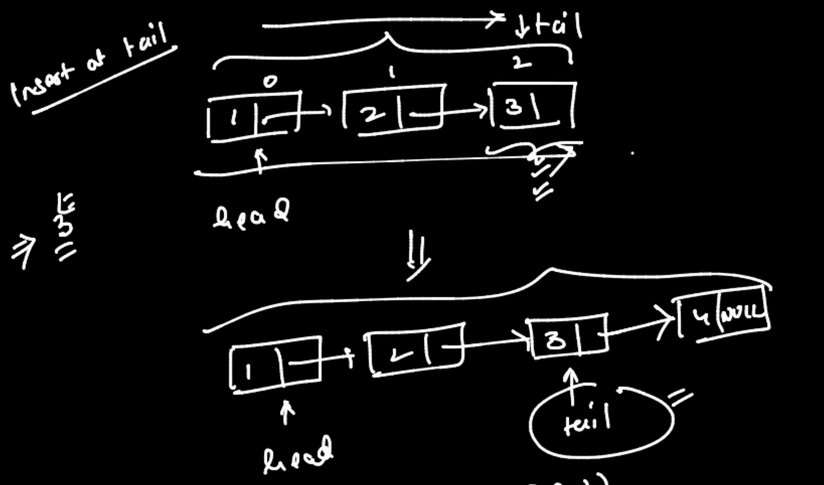
1. Get prev → ptr to node at index - 1

2. Create a node with value 3

3. ⇒ { n → next = prev → next
   − { prev → next = n



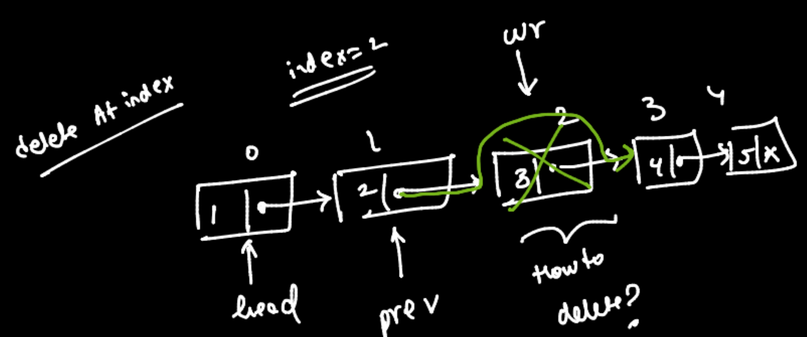
⇒ { temp = head
  { head = head → next
  { delete temp



⇒ 1. get the tail node = O(N)

{ 2. Create a new node

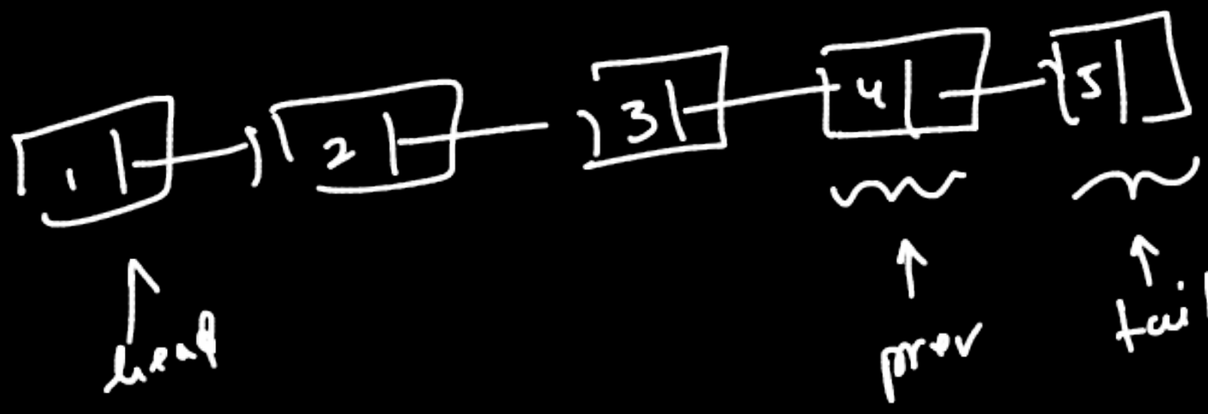
{ 3. Set next of tail to new node



find curr
find pre

- prev → next = curr → next
- delete curr

Delete from tail



prev->next = NULL

delete tail