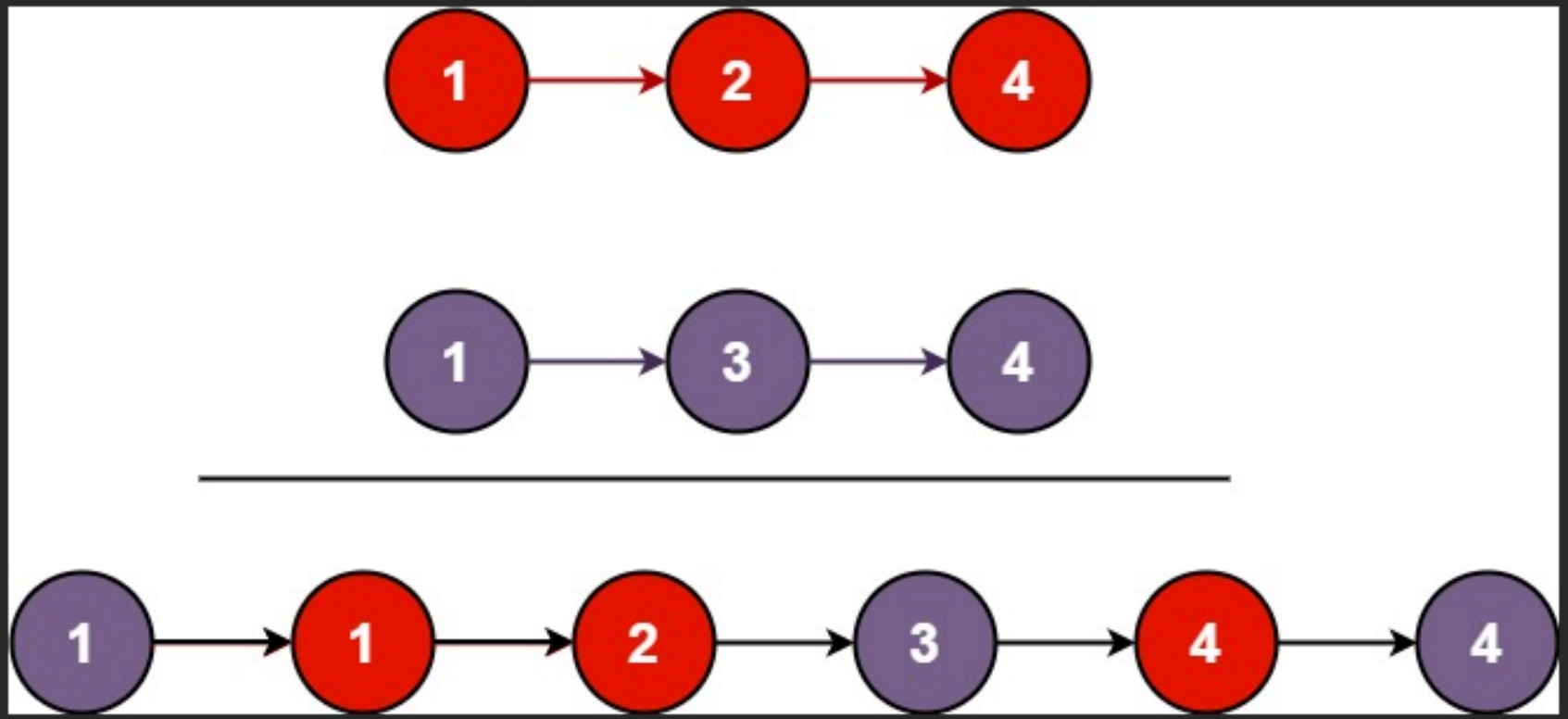
Merge Sorted Linked List

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list.*

Example 1:



> **Input:** list1 = [1,2,4], list2 = [1,3,4]
> **Output:** [1,1,2,3,4,4]

Linked List 1 → sorted

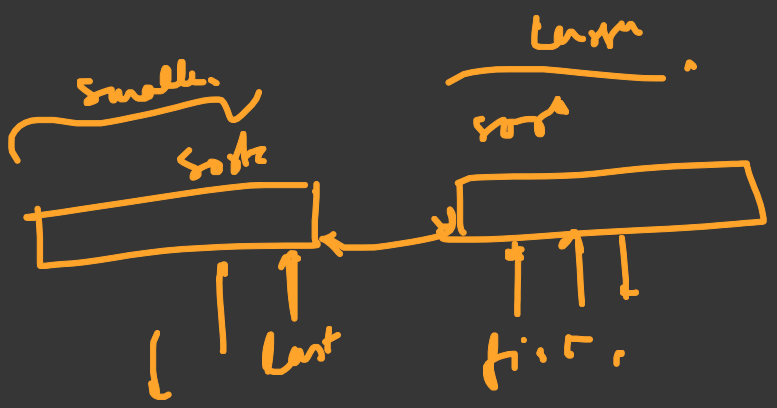
3 → 7 → 10

Linked List 2

1 → 2 → 5 → 8 → 10

head is given

Brute → merge sorted in array.



3 → 7 → 10

head

3

1 → 2 → 5 → 8 → 10

head

newhead → return

1 → 2 → 3

Idea

create a new head

create a new sorted linked list

Naive / Brute Approach

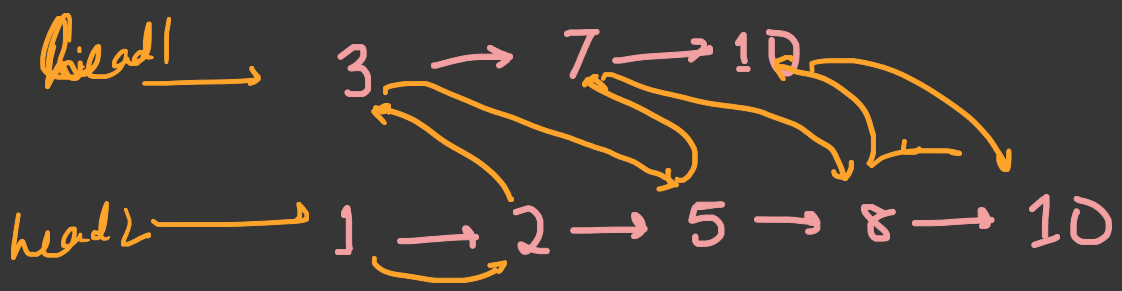
We create a dummy nelinked list and while iterating on the linked list we check if whichever is smaller we push in it the dummy linked list and we return the head of dummy linked list.

T.C. = O(N+M)

S.C = O(N+M) ⟶ Optimise

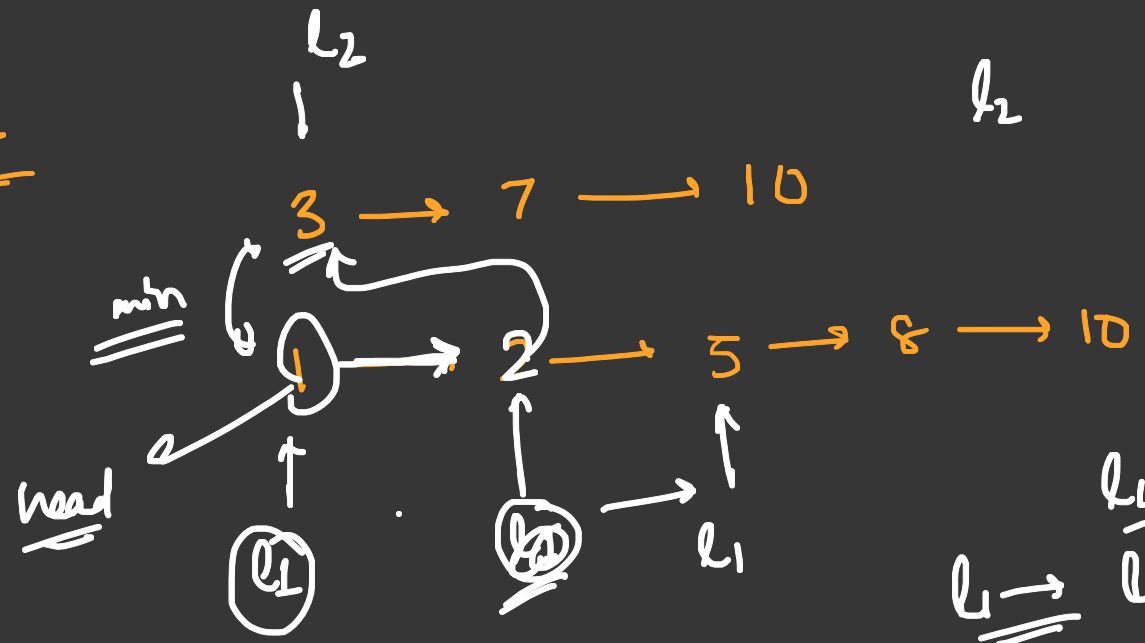
Optimised approach (In place) :

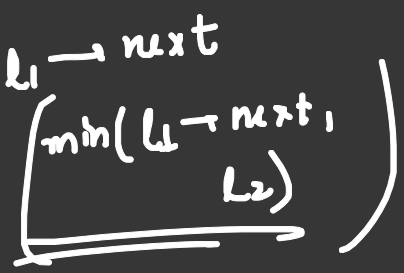
- We do not use any dummy linked list.



} Idea is to inplace arrange the sorted array.

- l1 l2





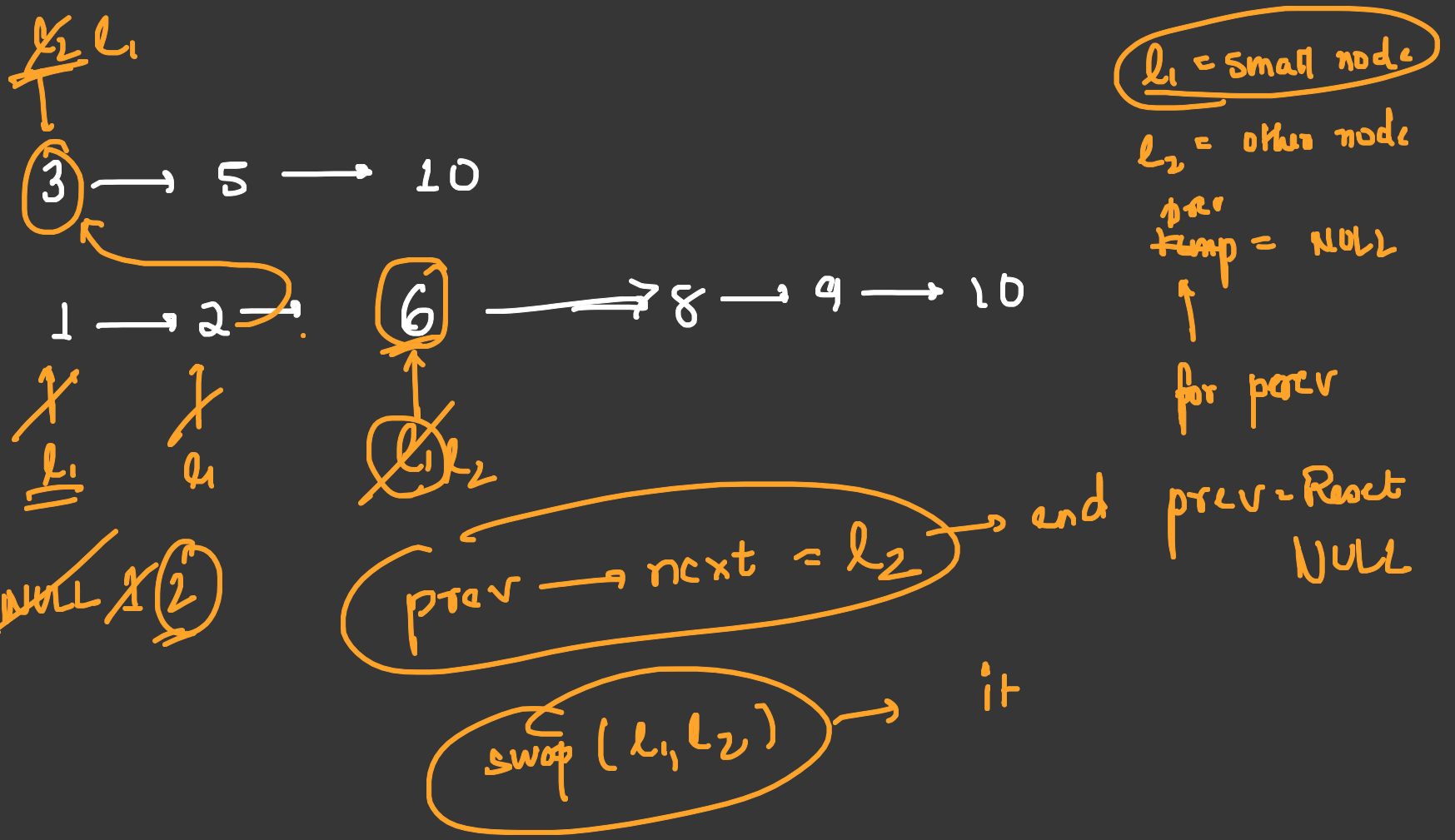
$l_1 = l_1 \rightarrow next$

$l_1 \rightarrow l_2$

$l_1 \rightarrow l_1 \rightarrow next$

$l_1 \rightarrow next = l_2$

Dry Run

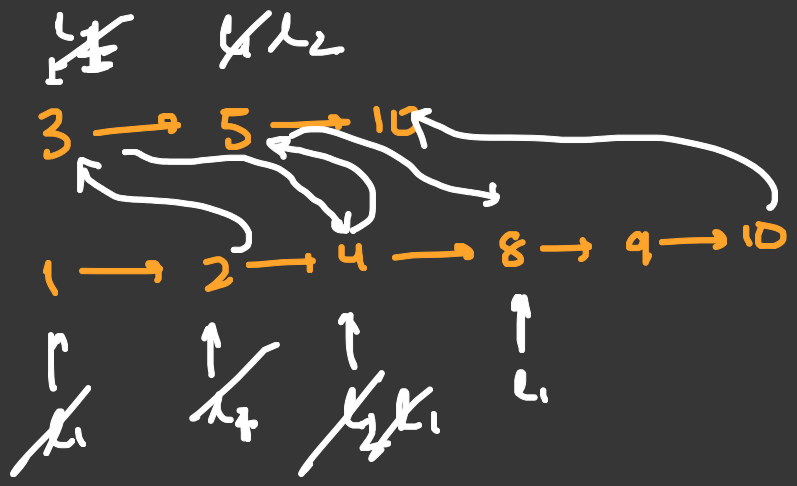

l2 l1
3 → 5 → 10
1 → 2 → 6 → 8 → 9 → 10
l1
l1 l2
l1 = small node
l2 = other node
prev
temp = NULL
for prev
prev = Head
NULL
prev → next = l2
end
swap (l1, l2)
it
prev = NULL 1 2
N L

Dry Run

l_1 = ~~1 2 3 4~~

l_2 = ~~3 5~~

prev = ~~NULL 1 2 NULL~~ 3 NULL



Pseudocode:

~~if~~ if(head1 → data < head2 → data) l_1 → head1 ; l_2 = head2

else l_1 = head2 , l_2 = head1;

prev = NULL

while(l_1 != NULL)

· while(l_1 → data < l_2 data) ~~l_1 = l_1 → next~~ prev = l_1

prev = l_1

l_1 = l_1 → next;

prev → next = l_2

prev = NULL

swap(l_1, l_2);

Code

```
if(l1 == NULL)  return l2;
    if(l2 == NULL)  return l1;
    if(l1->data > l2->data) swap(l1,l2);
    Node* res = l1;
    while(l1 != NULL && l2 != NULL){
        Node* temp = NULL;
        while(l1 != NULL && l1->data <= l2->data){
            temp = l1;
            l1 = l1->next;
        }
        temp->next = l2;
        swap(l1,l2);
    }
    return res;
```

T.C = O(N+M)

S.C = O(1)