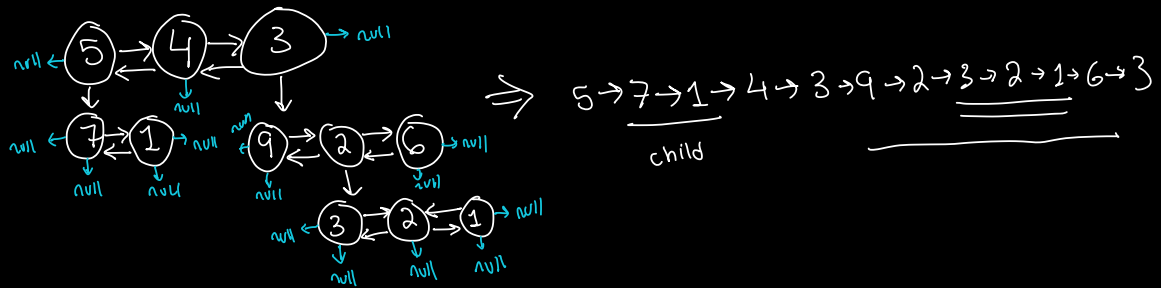


Problem: Given doubly linked list, list nodes have child property that can point to separate doubly linked list. Child lists can have own doubly linked lists. Return list as single level, flattened doubly linked list



Intro:

- Verify Constraints
 - every list node level can multiple children
 - child properties after flattening? set to null ✱
- Create Testcases

1 2 3 4 5 6
 |
 7 8 9 12-13
 |
 10 11

⇒ 1 2 7 8 10 11 9 3 4 5 12 13 6

Brute Force:

- Brainstorming & Pattern Observations

seems easier

① Merge
 Top-Down
 1 2 3
 |
 4 5 6
 |
 7
 ⇒ 1 2 4 5 6 3
 |
 7
 ⇒ 1 2 4 5 7 6 3

② Merge
 Bottom-Up
 1 2 3
 |
 4 5 6
 |
 7
 ⇒ 1 2 3
 4 5 7 6
 ⇒ 1 2 4 5 7 6 3

Gee

- Pseudocode

traverse thru first level

if curr.child != null :
 save curr = curr
 temp = curr.next
 curr.next = curr.child
 curr.child = null

curr temp
 1 2 3 → null
 |
 null ← 4 5 6 → null

1 2 4 5 6 → null

curr.
curr.next.prev = curr
// now we need tail

1 2 3 4 5 6 → null
↑
3

Traverse
slow and
level → while curr.next != null:
if curr.next.next == null:
tail = curr.next
curr.next = curr.next.next

1 2 3 4 5 6
|
7 8 9 10
|
11 12

tail.next = temp
temp.prev = tail
tail = curr
curr = save curr
curr = curr.next

curr
1 2 4 5 6 3
|
7

- Write code
- Run through testcases
- Analyze time and space complexity
Time: $O(n)$: touch every node at least once
Space: $O(1)$

Optimal:

- Brainstorming & Pattern Observations
- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity