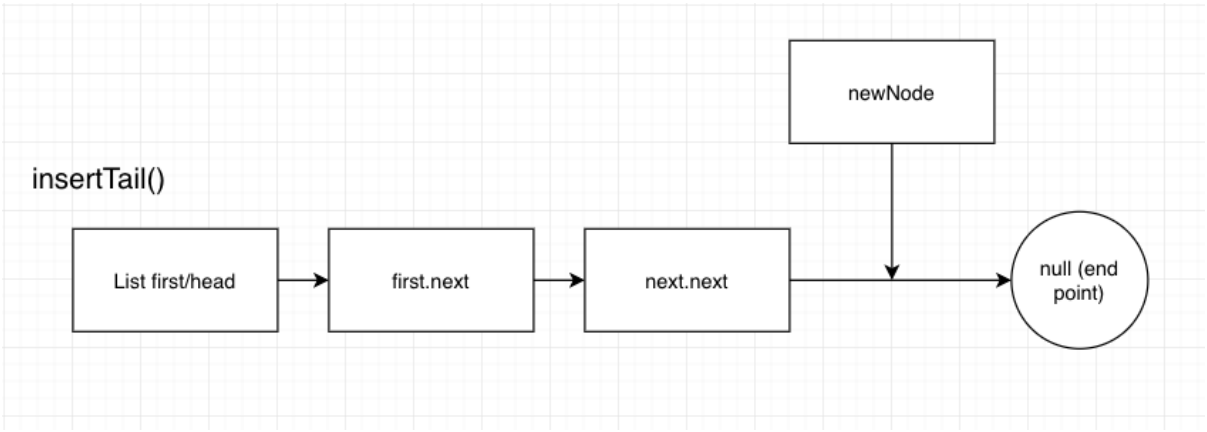
2460681S – Wesley Scott

insertTail

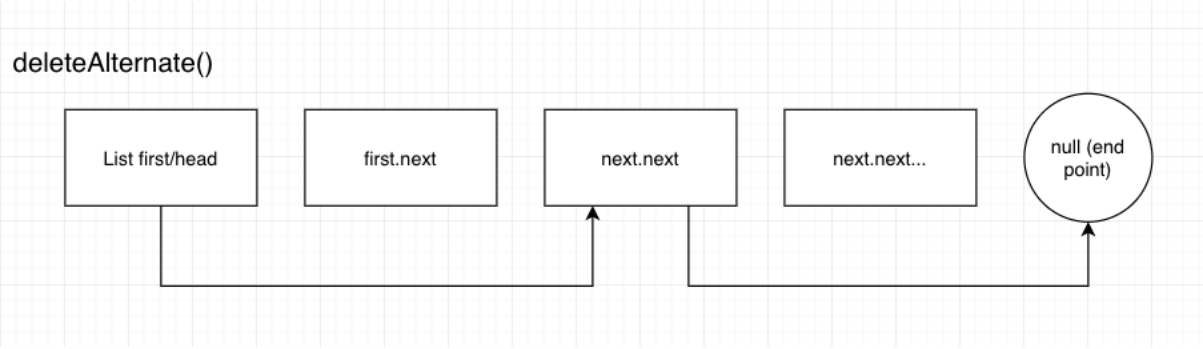


algorithm: no return, takes E elem

1. If the current list is empty, insert the element at the head [O(1)]
2. Prepare a new node. [O(1)]
3. Set the new node's next pointer to null. [O(1)]
4. Select the first node of the list [O(1)]
5. Check if the selected node's next node is null [O(n)]
    a. If that node is not null, select it as the new current node and repeat 5 [O(1)]
6. Once the currently selected node's next node is null, set the new node to be the next node. [O(1)]

Time complexity = O(N) overall

deleteAlternate



algorithm: no return, no args

1. If the first element in list is null there nothing to delete so return yielding nothing [O(1)]

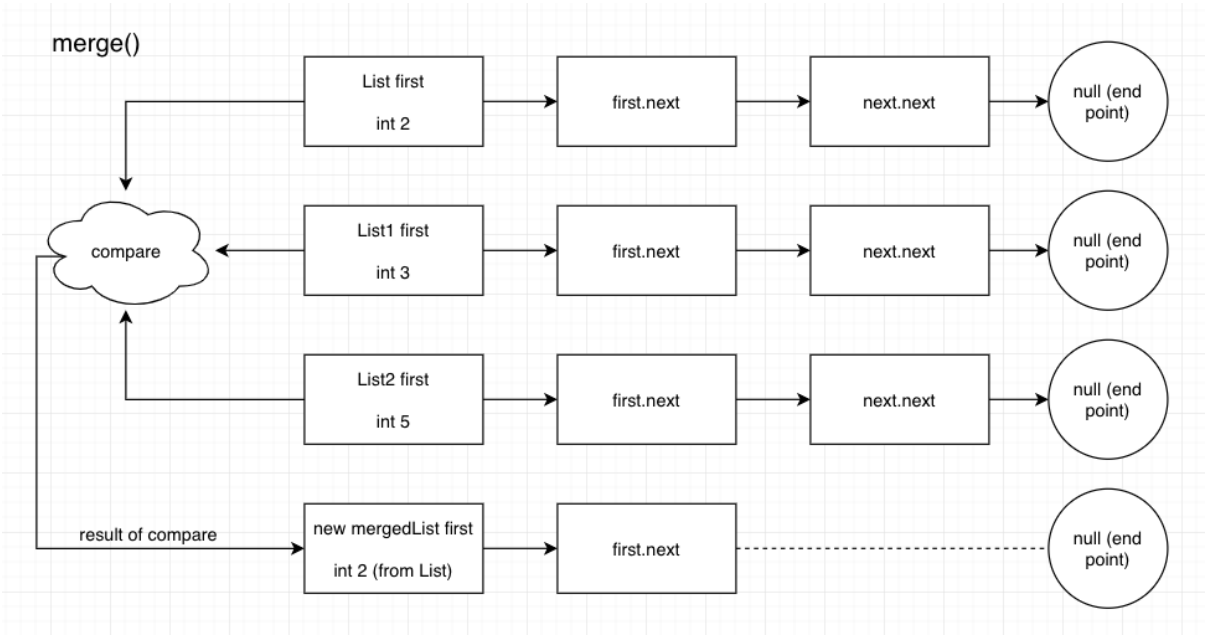
2. Select the first node as predecessor and the node after it as current [O(1)]

3. If neither of the nodes are null [O(N)]

    a. Change the link of the pred. node to point to current node's next node [O(1)]
    b. Nullify the current node [O(1)]
    c. Advance the pred. node to its next node [O(1)]
    d. If the new pred. is not null, set the current node to the next node of pred. [O(1)]
    e. Repeat 3 with new values [O(1)]

Time complexity = O(N) overall

merge



algorithm: returns linked list, takes 2 lists

1. Create a new empty linked list to store the merged values [O(1)]
2. Select the firsts/heads of each of the three lists to be merged [O(1)]
3. While the three lists are not empty [O(N)] N = sum lengths of lists
    a. Compare head one with head two. If comparing with null, take the value that's not null [O(1)]
    b. Compare head two with head three. If comparing with null, take the value that's not null [O(1)]
    c. Copy the smallest head into the new list as first node [O(1)]
    d. Advance the smallest head to its next node. If any of the other heads are equal to this head, advance them also [O(1)]
        i. When advancing, if the next node is the same as the previous, keep advancing until it's not
    e. Repeat 3 with new values [O(1)]
4. Return the merged list [O(1)]

Time complexity = O(N) overall (for sorted lists)