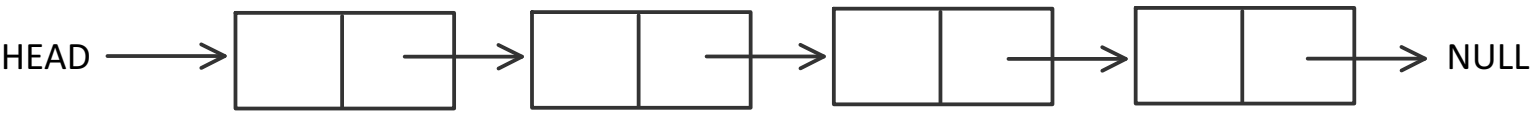


Singly linkedlist

Tuesday, May 3, 2022 7:11 PM

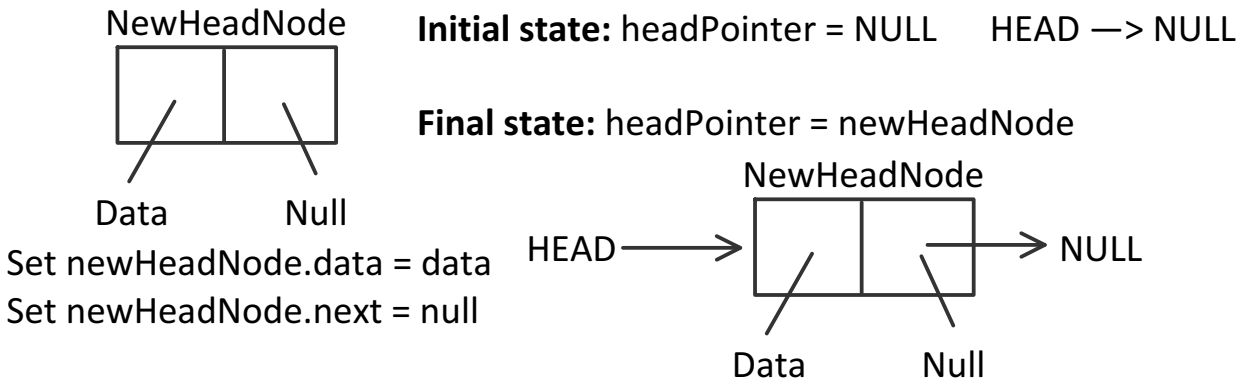
Singly linkedlist - Data structure & algorithms



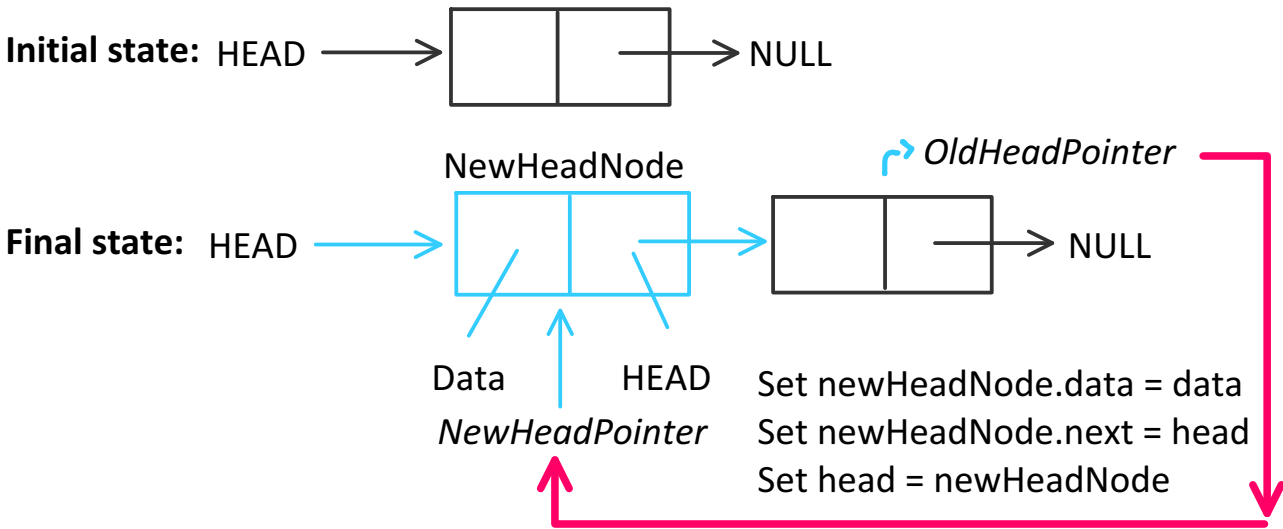
Initial state: headPointer = NULL
HEAD → NULL

InsertAtHead(int data)

- Check if the singly linkedlist is uninitialized. We can check this by verifying if head == null.
- If the headPointer is null, we will need to create a new singly linkedlist node and set head equal to it.

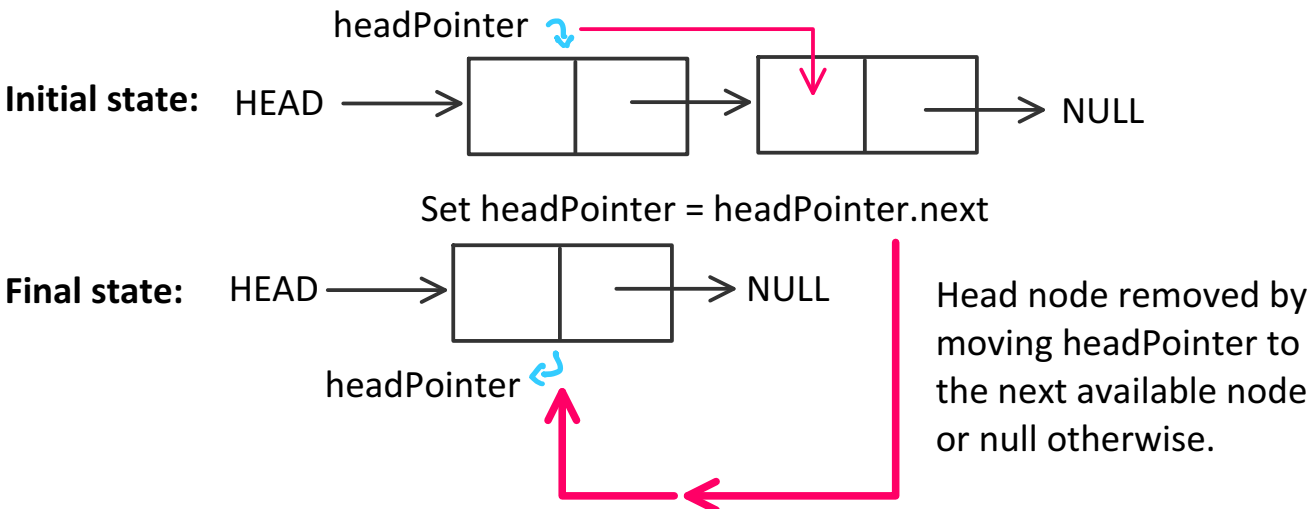


- If the headPointer is not null, we will still need to create a new singly linkedlist node and set its next pointer equal to the newHeadNode. Then we will assign the headPointer to the newHeadNode.



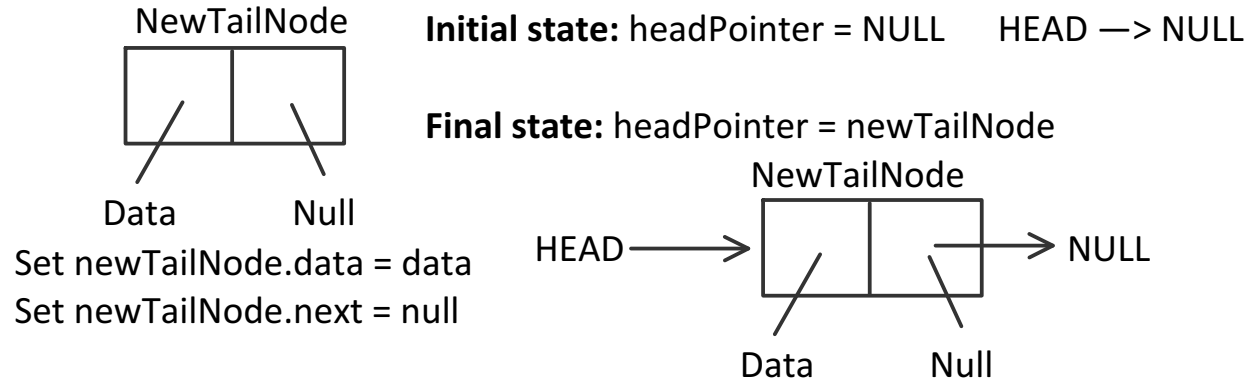
RemoveFromHead()

- Check if the singly linkedlist is uninitialized or empty. We can check this by verifying if head == null.
- If the headPointer is not null OR the singly linkedlist is initialized, we simply need to set the headPointer to the next singly linkedlist node in order to “removeFromHead”.

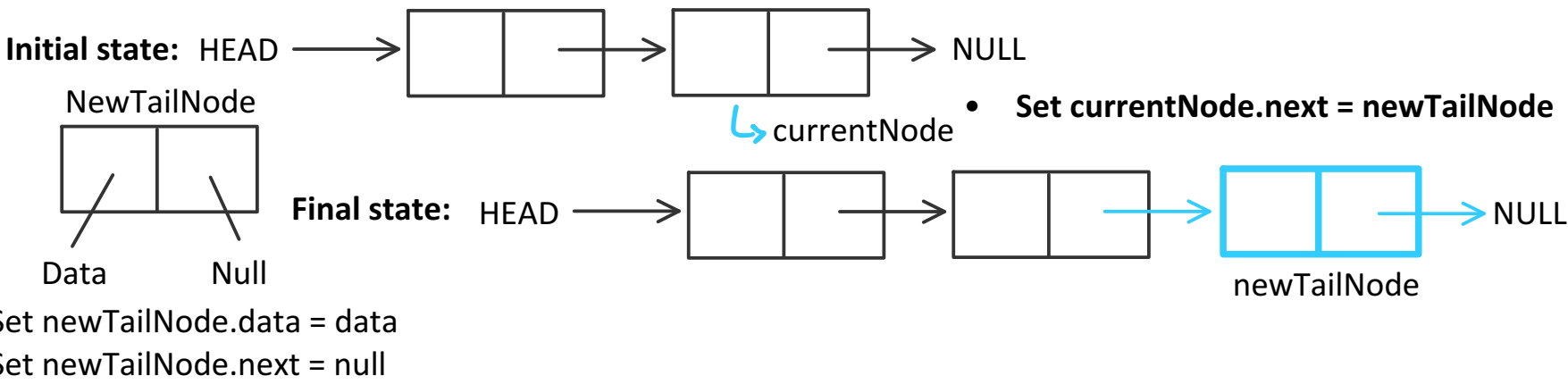


InsertAtTail(int data)

- Check if the singly linkedlist is uninitialized. We can check this by verifying if head == null.
- If the headPointer is null, we will need to create a new singly linkedlist node and set head equal to it.



- If the headPointer is not null, we will still need to create a new singly linkedlist node and set its next pointer equal to null. We need to traverse from the the head of the linkedlist to the last node and set its nextPointer to the new singly linkedlist node we just created.



RemoveFromTail()

- Check if the singly linkedlist is uninitialized. We can check this by verifying if head == null.
- If the singly linkedlist is initialized, we can removeFromTail(). We can do this by traversing from the head node all the way to the 2nd to last tail node! Then we can set that currentNode (that we traversed to) → currentNode.next = null. This will “remove” the tail node from our linkedlist.

