

Reverse a LinkedList (easy)

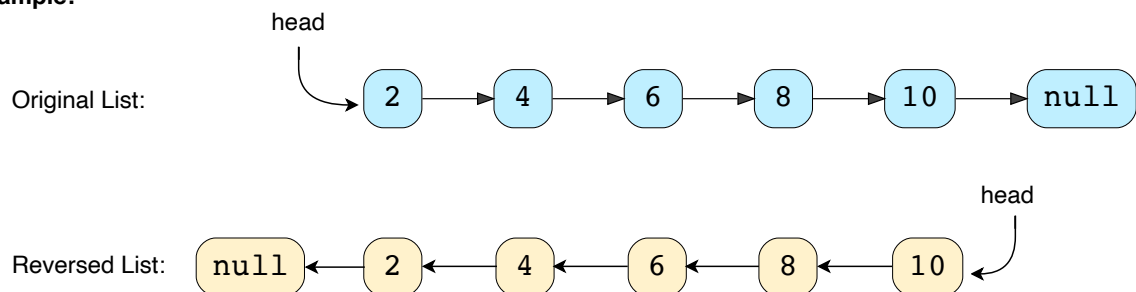
We'll cover the following ^

- Problem Statement
- Try it yourself
- Solution
 - Code
 - Time complexity
 - Space complexity

Problem Statement


Given the head of a Singly LinkedList, reverse the LinkedList. Write a function to return the new head of the reversed LinkedList.


Example:





Try it yourself

Try solving this question here:

 Java

 Python3

 JS

 C++

```
1 from __future__ import print_function
2
3
4 class Node:
5     def __init__(self, value, next=None):
6         self.value = value
7         self.next = next
8
9     def print_list(self):
10        temp = self
11        while temp is not None:
12            print(temp.value, end=" ")
13            temp = temp.next
14        print()
15
```

```

16
17 def reverse(head):
18     # TODO: Write your code here
19     return head
20
21
22 def main():
23     head = Node(2)
24     head.next = Node(4)
25     head.next.next = Node(6)
26     head.next.next.next = Node(8)
27     head.next.next.next.next = Node(10)
28

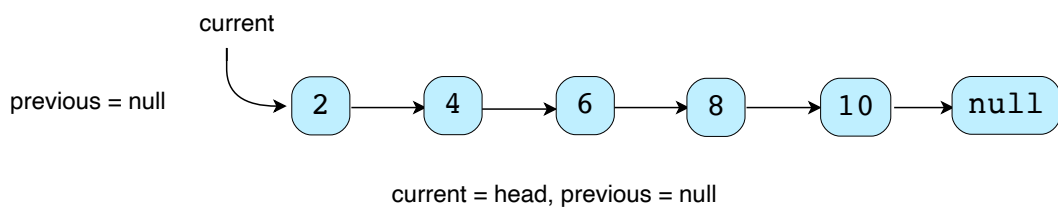
```



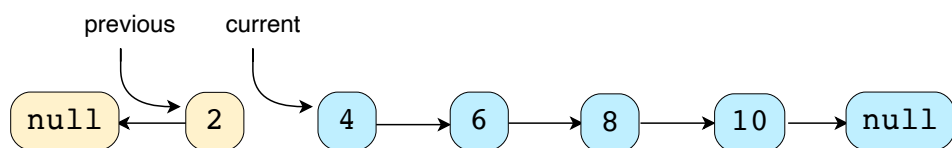
Solution

To reverse a LinkedList, we need to reverse one node at a time. We will start with a variable **current** which will initially point to the head of the LinkedList and a variable **previous** which will point to the previous node that we have processed; initially **previous** will point to **null**.

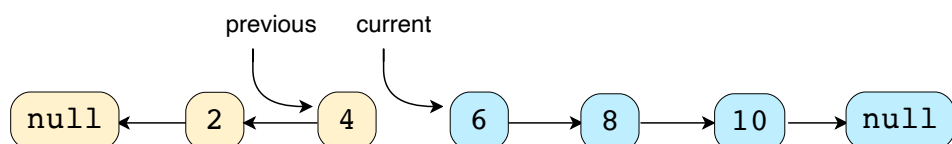
In a stepwise manner, we will reverse the **current** node by pointing it to the **previous** before moving on to the next node. Also, we will update the **previous** to always point to the previous node that we have processed. Here is the visual representation of our algorithm:



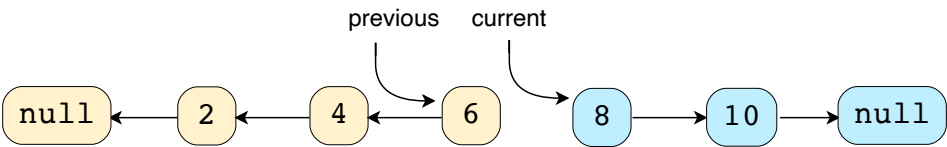
1 of 7



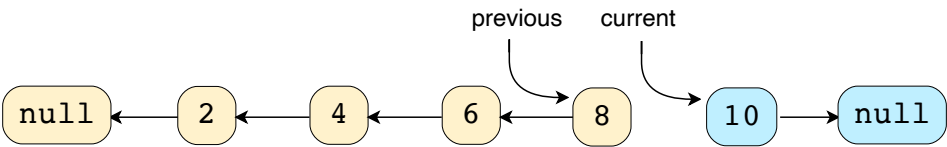
2 of 7



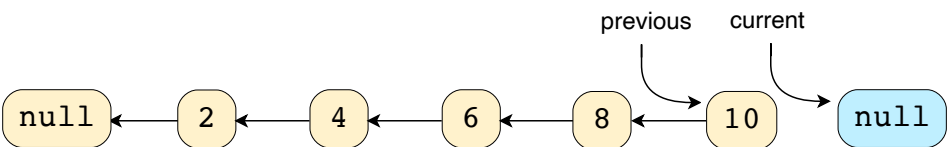
3 of 7



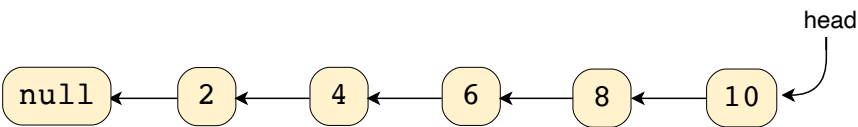
4 of 7



5 of 7



6 of 7



7 of 7

— []

Code #

Here is what our algorithm will look like:

Java

Python3

C++

JS

```

1 from __future__ import print_function
2
3
4 class Node:
5     def __init__(self, value, next=None):
6         self.value = value
7         self.next = next
8

```

⌵

educative

```

~
9   def print_list(self):
10      temp = self
11      while temp is not None:
12          print(temp.value, end=" ")
13          temp = temp.next
14      print()
15
16
17  def reverse(head):
18      previous, current, next = None, head, None
19      while current is not None:
20          next = current.next # temporarily store the next node
21          current.next = previous # reverse the current node
22          previous = current # before we move to the next node, point previous to the current n
23          current = next # move on the next node
24      return previous
25
26
27  def main():
28      head = Node(2)

```



Time complexity

The time complexity of our algorithm will be $O(N)$ where 'N' is the total number of nodes in the LinkedList.

Space complexity

We only used constant space, therefore, the space complexity of our algorithm is $O(1)$.

[← Back](#)

[Next →](#)

[Introduction](#)

[Reverse a Sub-list \(medium\)](#)



Mark as Completed



Report an Issue



Ask a Question

(https://discuss.educative.io/tag/reverse-a-linkedlist-easy__pattern-in-place-reversal-of-a-linkedlist__grokking-the-coding-interview-patterns-for-coding-questions)