Middle of the LinkedList (easy)

We'll cover the following

Problem Statement#

Given the head of a **Singly LinkedList**, write a method to return the **middle node** of the LinkedList.

If the total number of nodes in the LinkedList is even, return the second middle node.

Example 1:

```
Input: 1 -> 2 -> 3 -> 4 -> 5 -> null
Output: 3
```

Example 2:

```
Input: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> null
Output: 4
```

Example 3:

```
Input: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> null
Output: 4
```

Try it yourself#

Try solving this question here:

Java | Python3 | JS | C++

```
1
2  class ListNode {
3    int value = 0;
4    ListNode next;
5
6    ListNode(int value) {
7      this.value = value;
8    }
9  }
10
11 class MiddleOfLinkedList {
12
13   public static ListNode findMiddle(ListNode head) {
14     // TODO: Write your code here
15     return head;
16   }
17
18   public static void main(String[] args) {
```

```
19      ListNode head = new ListNode(1);
20      head.next = new ListNode(2);
21      head.next.next = new ListNode(3);
22      head.next.next.next = new ListNode(4);
23      head.next.next.next.next = new ListNode(5);
24      System.out.println("Middle Node: " + MiddleOfLinkedList.findMiddle(hea
25
26      head.next.next.next.next.next = new ListNode(6);
27      System.out.println("Middle Node: " + MiddleOfLinkedList.findMiddle(hea
28
```

educative(/learn)

Run Save Reset

Solution#

One brute force strategy could be to first count the number of nodes in the LinkedList and then find the middle node in the second iteration. Can we do this in one iteration?

We can use the **Fast & Slow pointers** method such that the fast pointer is always twice the nodes ahead of the slow pointer. This way, when the fast pointer reaches the end of the LinkedList, the slow pointer will be pointing at the middle node.

Code#

Here is what our algorithm will look like:

Java Python3 C++ JS JS

```
1
2  class ListNode {
3    int value = 0;
4    ListNode next;
```

```
5
6     ListNode(int value) {
7       this.value = value;
8     }
9   }
10
11  class MiddleOfLinkedList {
12
13    public static ListNode findMiddle(ListNode head) {
14      ListNode slow = head;
15      ListNode fast = head;
16      while (fast != null && fast.next != null) {
17        slow = slow.next;
18        fast = fast.next.next;
19      }
20
21      return slow;
22    }
23
24    public static void main(String[] args) {
25      ListNode head = new ListNode(1);
26      head.next = new ListNode(2);
27      head.next.next = new ListNode(3);
28      head.next.next.next = new ListNode(4);
```

Run Save Reset

Time complexity#

The above algorithm will have a time complexity of $O(N)$ where 'N' is the number of nodes in the LinkedList.

Space complexity#

The algorithm runs in constant space $O(1)$.

← Back

Next →

Happy Number (medium)

Problem Challenge 1

☑ Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/middle-of-the-linkedlist-easy__pattern-fast-slow-pointers__grokking-the-coding-interview-patterns-for-coding-questions?open=true&ctag=grokking-the-coding-interview-patterns-for-coding-questions__design-gurus&aid=5668639101419520&cid=5671464854355968&pid=6033606055034880)