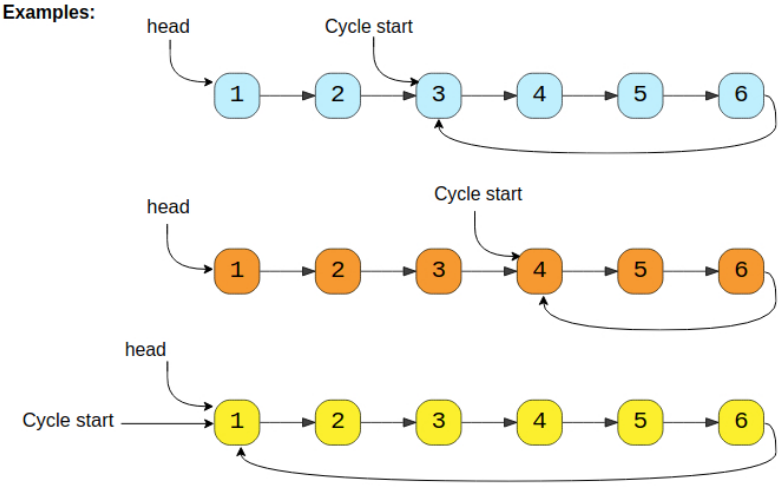
Start of LinkedList Cycle (medium)

We'll cover the following

Problem Statement

Given the head of a **Singly LinkedList** that contains a cycle, write a function to find the **starting node of the cycle**.

Examples:



Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```python
from __future__ import print_function


class Node:
  def __init__(self, value, next=None):
    self.value = value
    self.next = next

  def print_list(self):
    temp = self
    while temp is not None:
      print(temp.value, end='')
      temp = temp.next
    print()


def find_cycle_start(head):
  # TODO: Write your code here
  return head

```

```
def main():
  head = Node(1)
  head.next = Node(2)
  head.next.next = Node(3)
  head.next.next.next = Node(4)
  head.next.next.next.next = Node(5)
  head.next.next.next.next.next = Node(6)

  head.next.next.next.next.next.next = head.next.next
  print("LinkedList cycle start: " + str(find_cycle_start(head).value))

  head.next.next.next.next.next.next = head.next.next.next
  print("LinkedList cycle start: " + str(find_cycle_start(head).value))

  head.next.next.next.next.next.next = head
  print("LinkedList cycle start: " + str(find_cycle_start(head).value))


main()
```

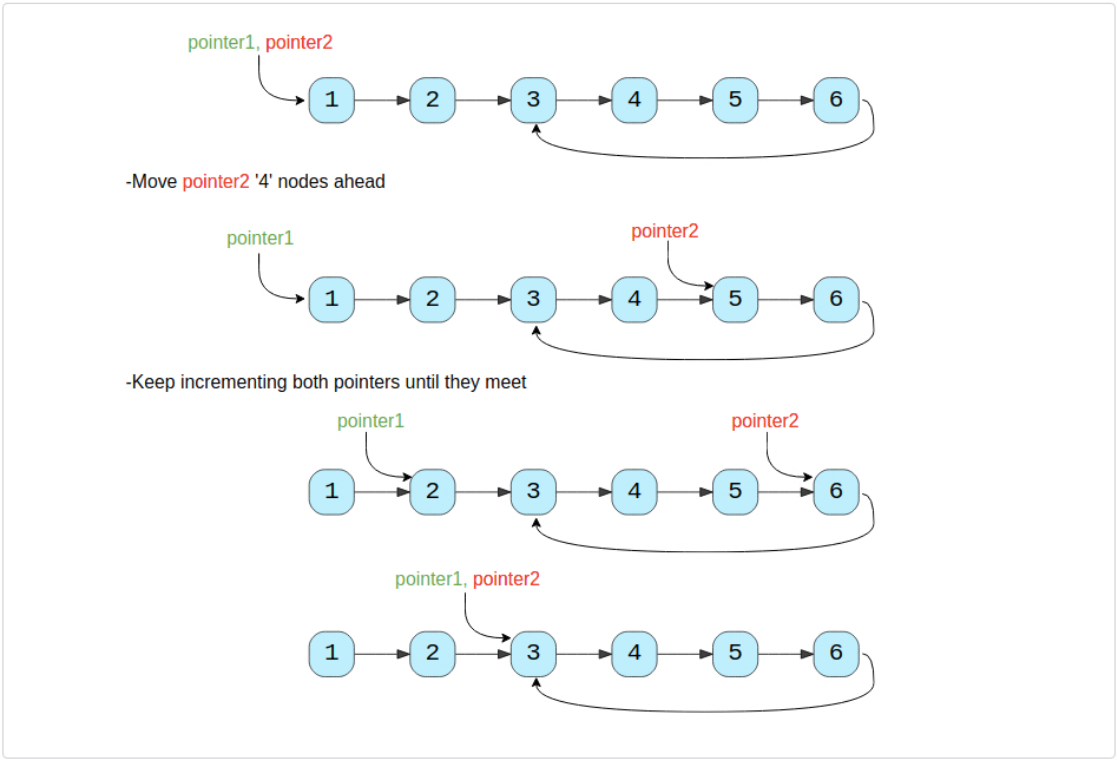
Run Save Reset

Solution

If we know the length of the **LinkedList** cycle, we can find the start of the cycle through the following steps:

1. Take two pointers. Let's call them `pointer1` and `pointer2`.
2. Initialize both pointers to point to the start of the LinkedList.
3. We can find the length of the LinkedList cycle using the approach discussed in LinkedList Cycle. Let's assume that the length of the cycle is 'K' nodes.
4. Move `pointer2` ahead by 'K' nodes.
5. Now, keep incrementing `pointer1` and `pointer2` until they both meet.
6. As `pointer2` is 'K' nodes ahead of `pointer1`, which means, `pointer2` must have completed one loop in the cycle when both pointers meet. Their meeting point will be the start of the cycle.

Let's visually see this with the above-mentioned Example-1:



We can use the algorithm discussed in LinkedList Cycle to find the length of the cycle and then follow the above-mentioned steps to find the start of the cycle.

Code

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```python
from __future__ import print_function


class Node:
  def __init__(self, value, next=None):
    self.value = value
    self.next = next

  def print_list(self):
    temp = self
    while temp is not None:
      print(temp.value, end='')
      temp = temp.next
    print()


def find_cycle_start(head):
  cycle_length = 0
  # find the LinkedList cycle
  slow, fast = head, head
  while (fast is not None and fast.next is not None):
    fast = fast.next.next
    slow = slow.next
    if slow == fast:  # found the cycle
      cycle_length = calculate_cycle_length(slow)
      break
  return find_start(head, cycle_length)


def calculate_cycle_length(slow):
  current = slow
  cycle_length = 0
  while True:
    current = current.next
    cycle_length += 1
    if current == slow:
      break
  return cycle_length


def find_start(head, cycle_length):
  pointer1 = head
  pointer2 = head
  # move pointer2 ahead 'cycle_length' nodes
  while cycle_length > 0:
    pointer2 = pointer2.next
    cycle_length -= 1
  # increment both pointers until they meet at the start of the cycle
  while pointer1 != pointer2:
    pointer1 = pointer1.next
    pointer2 = pointer2.next
  return pointer1


def main():
  head = Node(1)
  head.next = Node(2)
  head.next.next = Node(3)
  head.next.next.next = Node(4)
  head.next.next.next.next = Node(5)
  head.next.next.next.next.next = Node(6)

  head.next.next.next.next.next.next = head.next.next
  print("LinkedList cycle start: " + str(find_cycle_start(head).value))

  head.next.next.next.next.next.next = head.next.next.next
  print("LinkedList cycle start: " + str(find_cycle_start(head).value))

  head.next.next.next.next.next.next = head
  print("LinkedList cycle start: " + str(find_cycle_start(head).value))


main()

```

Run | Save | Reset

Time Complexity

As we know, finding the cycle in a LinkedList with 'N' nodes and also finding the length of the cycle requires $O(N)$. Also, as we saw in the above algorithm, we will need $O(N)$ to find the start of the cycle. Therefore, the overall time complexity of our algorithm will be $O(N)$.

Space Complexity

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

Back
LinkedList Cycle (easy)

Next
Happy Number (medium)

Completed

Report an Issue Ask a Question