educative

Grokking the Coding Interview: Patterns for Coding Questions

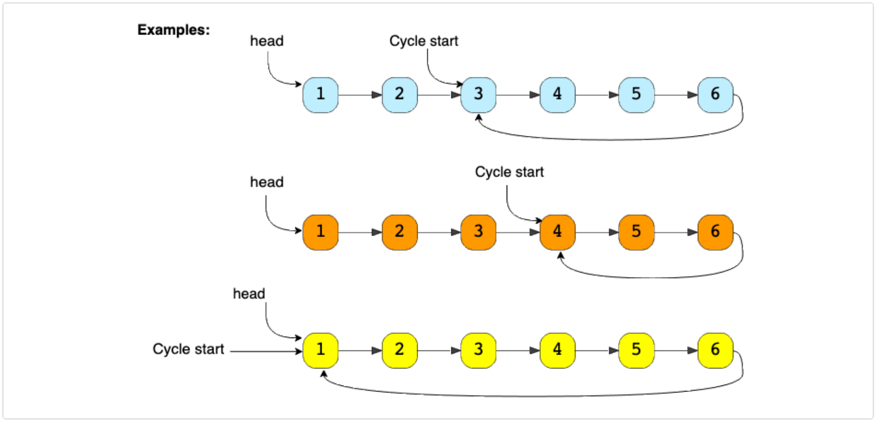
14% completed

Search Course

Start of LinkedList Cycle (medium)

Problem Statement

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



Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```javascript
  constructor(value, next=null){
    this.value = value;
    this.next = next;
  }
}

const find_cycle_start = function(head){
  // TODO: Write your code here
  return head;
};


head = new Node(1)
head.next = new Node(2)
head.next.next = new Node(3)
head.next.next.next = new Node(4)
head.next.next.next.next = new Node(5)
head.next.next.next.next.next = new Node(6)

head.next.next.next.next.next.next = head.next.next
console.log(`LinkedList cycle start: ${find_cycle_start(head).value}`)

head.next.next.next.next.next.next = head.next.next.next
console.log(`LinkedList cycle start: ${find_cycle_start(head).value}`)

head.next.next.next.next.next.next = head
console.log(`LinkedList cycle start: ${find_cycle_start(head).value}`)
```

RUN SAVE RESET

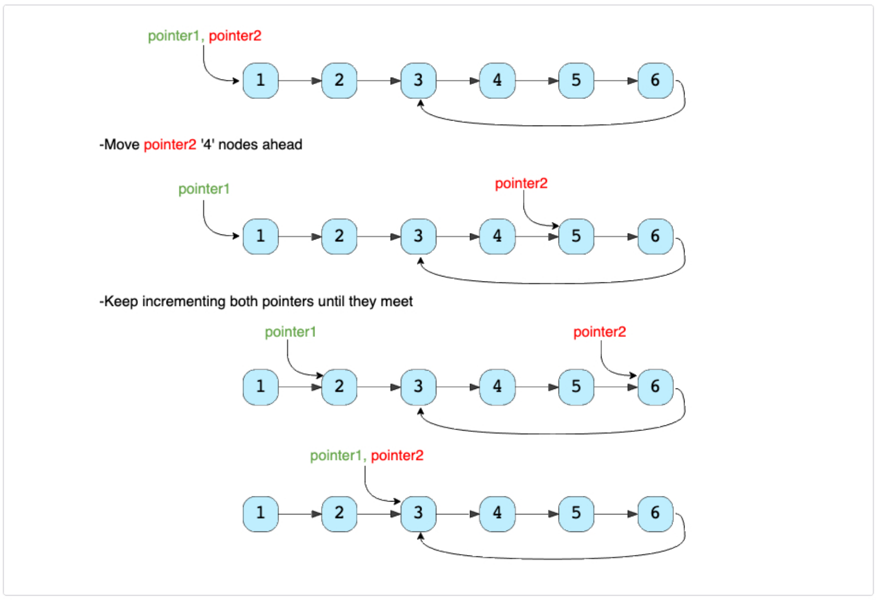
Solution

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.

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

```
class Node {
  constructor(value, next = null) {
    this.value = value;
    this.next = next;
  }
}

function find_cycle_start(head) {
  cycle_length = 0;
  // find the LinkedList cycle
  let slow = head,
    fast = head;
  while ((fast !== null && fast.next !== null)) {
    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);
}


function calculate_cycle_length(slow) {
  let current = slow,
    cycle_length = 0;
  while (true) {
```

RUN SAVE RESET

Close

Output 2.306s

```
LinkedList cycle start: 3
LinkedList cycle start: 4
LinkedList cycle start: 1
```

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 $O(1)$.

Create

MARK AS COMPLETED

← Back
LinkedList Cycle (easy)

Next →
Happy Number (medium)

Report an Issue