Detecting Cycle in Singly Linked List

A slightly silly Microsoft interview question.
But still a nice question.
Cycle in List

Goal: distinguish one situation from the other.
One (Bad) Idea

Somehow mark nodes that have been visited.

Node visit = head;
The Bad Idea

We’ll leave it open how to mark the node.

<mark node>
visit = visit.next;

The Bad Idea

<mark node>
visit = visit.next;

head → node1 → node2 → visit → node3 → node4 → node5
The Bad Idea

\[<\text{mark node}>\]
visit = visit.next;

head -> \[\text{node 1}\] -> \[\text{node 2}\] -> \[\text{node 3}\] -> visit -> visit

\[<\text{node 1}\] -> \[\text{node 2}\] -> \[\text{node 3}\] -> visit

\[<\text{node 1}\] -> \[\text{node 2}\] -> \[\text{node 3}\]
The Bad Idea

<mark node>
visit = visit.next;

head → visit → visit → visit → visit → visit → visit
<mark node>
visit = visit.next;

head

visit
The Bad Idea

<mark node>
visit = visit.next;

\[\text{head} \rightarrow \text{node} \rightarrow \text{node} \rightarrow \text{node} \rightarrow \text{node} \rightarrow \text{node} \rightarrow \text{visited node} \]

\[\text{visit} \rightarrow \text{visited node} \rightarrow \text{visited node} \rightarrow \text{visited node} \]
The Bad Idea

<mark node>
visit = visit.next;

head → node1 → node2 → node3 → node4 → node5

visit → node6 → node7 → node8
The Bad Idea

Stop when we see visited node (or null).
Why a Bad Idea?

Well, it’s not really bad. But we either need to add a field `visited` to each node or maintain an array of the nodes seen so far. Requires extra space and possibly time.
The original question

Use a **constant** amount of extra space (irrespective of the number of nodes in the list).
The Nice Idea

Pointer jumping:

Node slow = head;
Node fast = head;
The Nice Idea

We’ll live dangerously with no error checking.

\[
\text{slow} = \text{slow}.\text{next}; \\
\text{fast} = \text{fast}.\text{next}.\text{next};
\]
The Nice Idea

slow = slow.next;
fast = fast.next.next;
The Nice Idea

slow = slow.next;
fast = fast.next.next;
The Nice Idea

\[
\text{slow} = \text{slow}.\text{next}; \\
\text{fast} = \text{fast}.\text{next}.\text{next};
\]
The Nice Idea

slow = slow.next;
fast = fast.next.next;
The Nice Idea

slow = slow.next;
fast = fast.next.next;
do all this  
while (slow != fast)

Since the pointers collide, there is a cycle.

If either pointer hits null, then no cycle.