**Reverse Linked List**

Let's start with a classic problem:

Reverse a singly linked list.

One solution is to iterate the nodes in original order and move them to the head of the list one by one. It seems hard to understand. We will first use an example to go through our algorithm.

*Algorithm Overview*

Let's look at an example:

A blue line with a square and a point

AI-generated content may be incorrect.

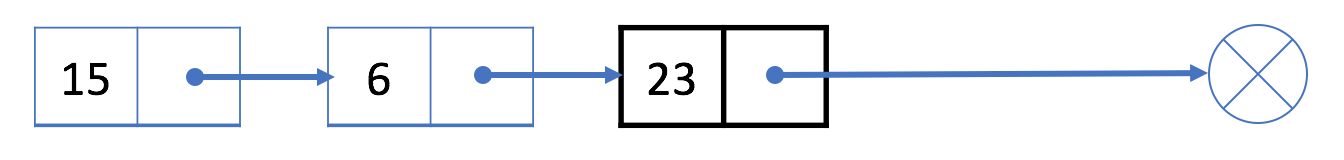
Keep in mind that the black node 23 is our original head node.

1. First, we move the next node of the black node, which is node 6, to the head of the list:

A blue line on a white background

AI-generated content may be incorrect.

2. Then we move the next node of the black node, which is node 15, to the head of the list:



3. The next node of the black node now is null. So we stop and return our new head node 15.

*More*

In this algorithm, each node will be moved exactly once.

Therefore, the time complexity is O(N), where N is the length of the linked list. We only use constant extra space so the space complexity is O(1).

This problem is the foundation of many linked-list problems you might come across in your interview. If you are still stuck, our next article will talk more about the implementation details.

There are also many other solutions. You should be familiar with at least one solution and be able to implement it.