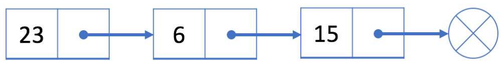
A Introduction - Singly Linked List

Report Issue (https://github.com/LeetCode-Feedback/LeetCode-Feedback/issues)

Each node in a singly-linked list contains not only the value but also `a reference field` to link to the next node. By this way, the singly-linked list organizes all the nodes in a sequence.

Here is an example of a singly-linked list:



The blue arrows show how nodes in a singly linked list are combined together.

Node Structure

Here is the typical definition of a node in a singly-linked list:

C++ Java Copy

```
// Definition for singly-linked list.
struct SinglyListNode {
    int val;
    SinglyListNode *next;
    SinglyListNode(int x) : val(x), next(NULL) {}
};
```

In most cases, we will use the `head` node (the first node) to represent the whole list.

Operations

Unlike the array, we are not able to access a random element in a singly-linked list in constant time. If we want to get the ith element, we have to traverse from the head node one by one. It takes us `O(N)` time on average to `visit an element by index`, where *N* is the length of the linked list.

For instance, in the example above, the head is the node 23. The only way to visit the 3rd node is to use the "next" field of the head node to get to the 2nd node (node 6); Then with the "next" field of node 6, we are able to visit the 3rd node.

You might wonder why the linked list is useful though it has such a bad performance (compared to the array) in accessing data by index. We will introduce the `insert` and `delete` operations in next two articles and you will realize the benefit of the linked list.

After that, we provide an exercise for you to design your own singly linked list.