Description | Solution | Discuss (999+) | Submissions

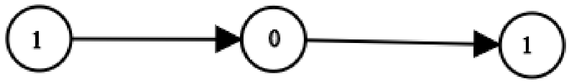
1290. Convert Binary Number in a Linked List to Integer

Easy 2151 100 Add to List Share

Given `head` which is a reference node to a singly-linked list. The value of each node in the linked list is either `0` or `1`. The linked list holds the binary representation of a number.

Return the *decimal value* of the number in the linked list.

Example 1:




```
Input: head = [1,0,1]
Output: 5
Explanation: (101) in base 2 = (5) in base 10
```

Example 2:

```
Input: head = [0]
Output: 0
```

Constraints:

Python3 | Autocomplete

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def getDecimalValue(self, head: ListNode) -> int:
        k = head
        r = []
        while k:
            if k.val != None:
                r.append(k.val)

            k = k.next
        l = len(r)
        s = 0
        for i in range(0, l):
            s = s + r[l-i-1]*(2**i)

        return s
```

Your previous code was restored from your local storage. Reset to default

Description | Solution | Discuss (999+) | Submissions

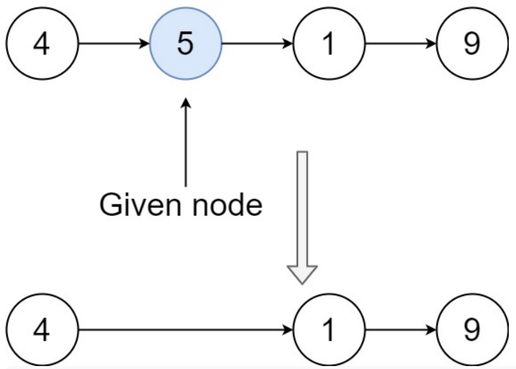
237. Delete Node in a Linked List

Easy 3700 10708 Add to List Share

Write a function to **delete a node** in a singly-linked list. You will **not** be given access to the `head` of the list, instead you will be given access to **the node to be deleted** directly.

It is **guaranteed** that the node to be deleted is **not a tail node** in the list.

Example 1:




```
Input: head = [4,5,1,9], node = 5
Output: [4,1,9]
Explanation: You are given the second node with value 5, the linked list should become 4
-> 1 -> 9 after calling your function.
```

Example 2:

Python3 Autocomplete

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def deleteNode(self, node):
        node.val=node.next.val
        node.next=node.next.next


```

Your previous code was restored from your local storage. Reset to default

Description | Solution | Discuss (999+) | Submissions

876. Middle of the Linked List

Easy 4256 110 Add to List Share

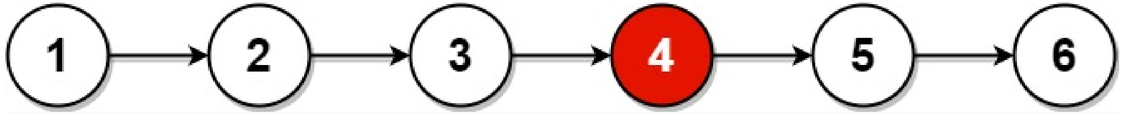
Given the `head` of a singly linked list, return *the middle node of the linked list*.

If there are two middle nodes, return **the second middle** node.

Example 1:


```
Input: head = [1,2,3,4,5]
Output: [3,4,5]
Explanation: The middle node of the list is node 3.
```

Example 2:




```
Input: head = [1,2,3,4,5,6]
Output: [4,5,6]
Explanation: Since the list has two middle nodes with values 3 and 4, we return the
second one.
```

Python3 | Autocomplete

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def middleNode(self, head: Optional[ListNode]) -> Optional[ListNode]:

        slow = fast = head
        while fast and fast.next:

            slow = slow.next
            fast = fast.next.next
        return slow
```

Your previous code was restored from your local storage. Reset to default

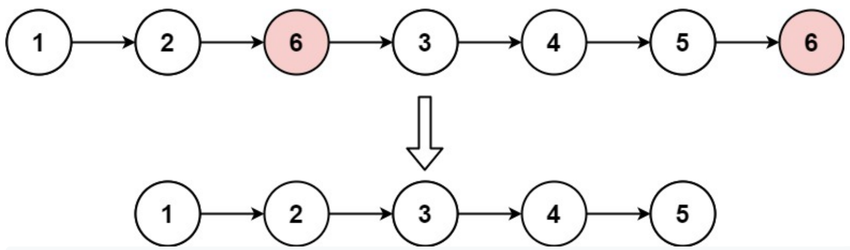
Description | Solution | Discuss (999+) | Submissions

203. Remove Linked List Elements

Easy 4162 151 Add to List Share

Given the `head` of a linked list and an integer `val`, remove all the nodes of the linked list that has `Node.val == val`, and return *the new head*.

Example 1:




```
Input: head = [1,2,6,3,4,5,6], val = 6
Output: [1,2,3,4,5]
```

Example 2:

```
Input: head = [], val = 1
Output: []
```

Example 3:

```
Input: head = [7,7,7,7], val = 7
```

Python3 | Autocomplete

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def removeElements(self, head: Optional[ListNode], val: int) -> Optional[ListNode]:
        k = head
        prev = None

        while k:
            if k.val == val:
                if prev:
                    prev.next = k.next
                else:
                    head = k.next
                k = k.next

            else:
                prev = k
                k = k.next

        return head
```

Your previous code was restored from your local storage. Reset to default

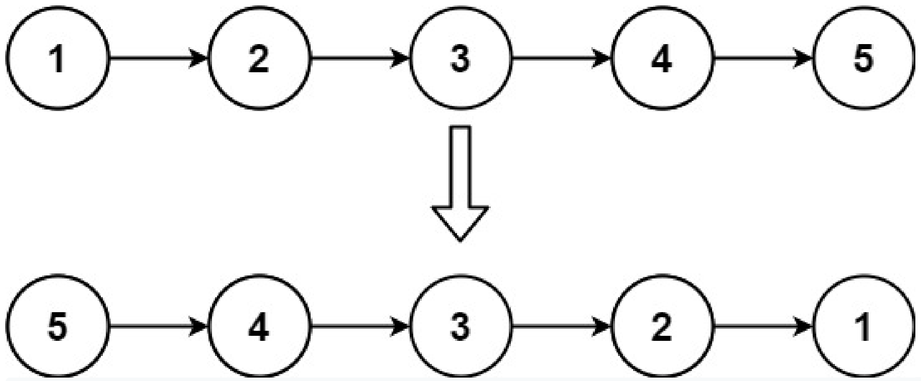
Description | Solution | Discuss (999+) | Submissions

206. Reverse Linked List

Easy 9819 170 Add to List Share

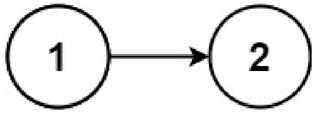
Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

Example 1:




```
Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]
```

Example 2:



Python3 | Autocomplete

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:

        if not head:
            return None

        else:
            k = head
            prev = None
            fallow = k.next


            while k != None:
                k.next = prev
                prev = k
                k = fallow

                if fallow:
                    fallow = fallow.next

            return prev
```

Your previous code was restored from your local storage. Reset to default

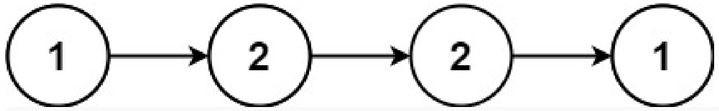
Description | Solution | Discuss (999+) | Submissions

234. Palindrome Linked List

Easy 7405 492 Add to List Share

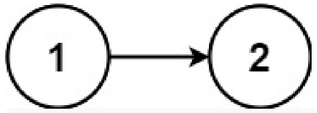
Given the `head` of a singly linked list, return `true` if it is a palindrome.

Example 1:




```
Input: head = [1,2,2,1]
Output: true
```

Example 2:




```
Input: head = [1,2]
Output: false
```

Constraints:

- The number of nodes in the list is in the range `[1, 10^5]`.
- `0 <= Node.val <= 9`

Python3 | Autocomplete

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def isPalindrome(self, head: Optional[ListNode]) -> bool:
        k = head
        r = []
        while k:
            if k.val != None:
                r.append(k.val)

            k = k.next

        p = len(r)
        for i in range(0,p):
            if (r[i]==r[p-i-1]):
                pass
            else:
                return False


        return True
```

Your previous code was restored from your local storage. Reset to default

Description | Solution | Discuss (999+) | Submissions

141. Linked List Cycle

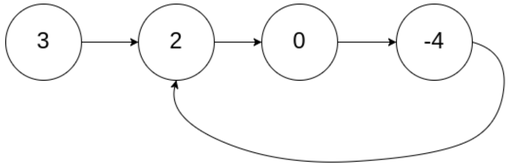
Easy 6413 707 Add to List Share

Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to. **Note that `pos` is not passed as a parameter**.

Return `true` *if there is a cycle in the linked list*. Otherwise, return `false`.

Example 1:




```
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st
node (0-indexed).
```

Example 2:

Python3 Autocomplete

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def hasCycle(self, head: Optional[ListNode]) -> bool:
        dic = set([])
        while head is not None:
            if head not in dic:
                dic.add(head)
            head = head.next
            if head in dic:
                return True
        return False
```

Your previous code was restored from your local storage. Reset to default

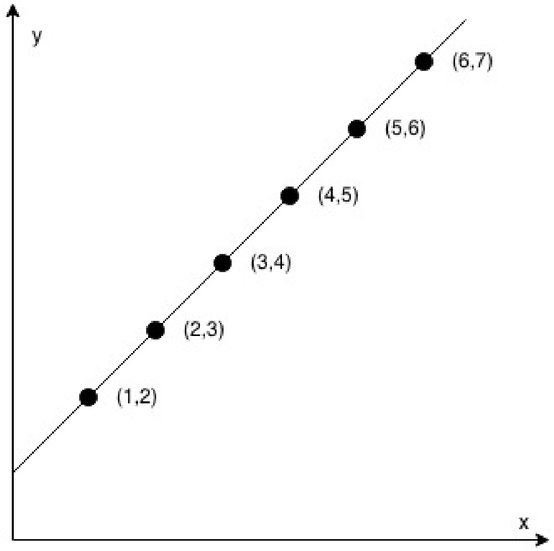
Description | Solution | Discuss (941) | Submissions

1232. Check If It Is a Straight Line

Easy 716 107 Add to List Share

You are given an array `coordinates`, `coordinates[i] = [x, y]`, where `[x, y]` represents the coordinate of a point. Check if these points make a straight line in the XY plane.

Example 1:



Python3 | Autocomplete

```python
class Solution:
    def checkStraightLine(self, coordinates: List[List[int]]) -> bool:
        x1, y1 = coordinates[0]
        x2, y2 = coordinates[1]

        for x, y in coordinates:
            if (y2 - y1) * (x - x1) != (x2 - x1) * (y - y1):
                return False

        return True
```

Your previous code was restored from your local storage. Reset to default

Testcase | Run Code Result | Debugger

Accepted Runtime: 45 ms

Your input `[[1,2],[2,3],[3,4],[4,5],[5,6],[6,7]]`

Output `true` Diff

Expected `true`