

## 876. Middle of the Linked List

Easy Topics Companies

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:

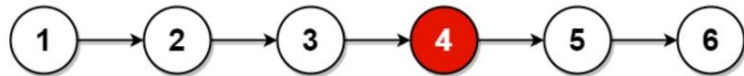


Input: `head = [1,2,3,4,5]`

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.

Constraints:

- The number of nodes in the list is in the range `[1, 100]`.
- `1 <= Node.val <= 100`

</> Code

Python3 Auto

```
1 # Definition for singly-linked list.
2 # class ListNode:
3 #     def __init__(self, val=0, next=None):
4 #         self.val = val
5 #         self.next = next
6 class Solution:
7     def middleNode(self, head: Optional[ListNode]) -> Optional[ListNode]:
8         |
```

Saved

Ln 8, Col 9

Testcase Test Result

Case 1 Case 2 +

head =

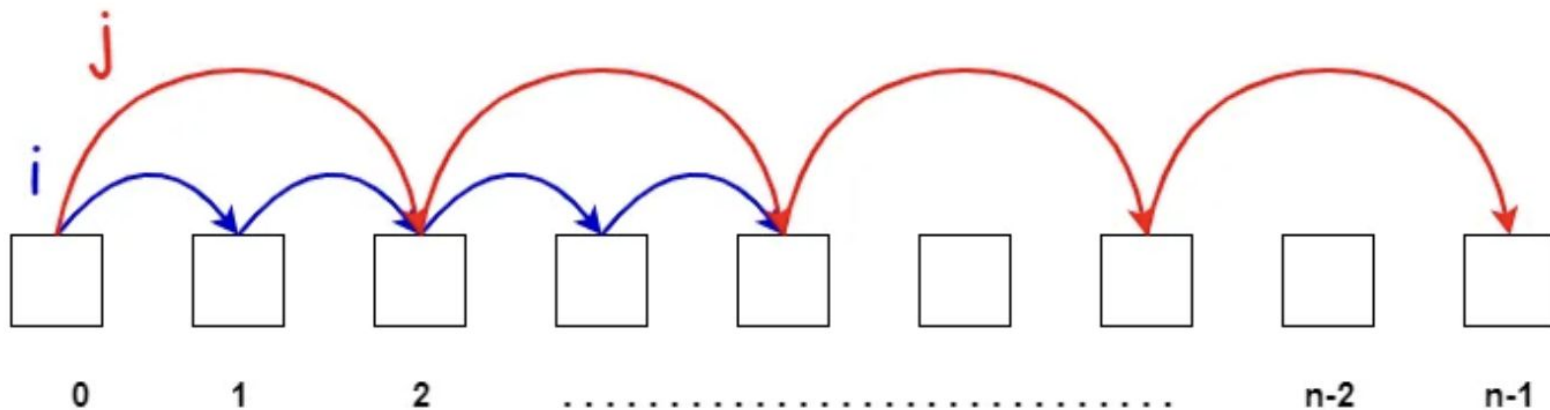
[1,2,3,4,5]

</> Source ?



## 876. Middle of the Linked List

Easy

[Topics](#)[Companies](#)

### Constraints:

- The number of nodes in the list is in the range  $[1, 100]$ .
- $1 \leq \text{Node.val} \leq 100$

## 876. Middle of the Linked List

Easy Topics Companies

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:

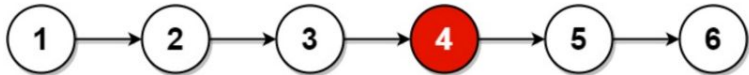


Input: `head = [1,2,3,4,5]`

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.

Constraints:

- The number of nodes in the list is in the range `[1, 100]`.
- `1 <= Node.val <= 100`

Solved

Code

Python3 Auto

```

1 # Definition for singly-linked list.
2 # class ListNode:
3 #     def __init__(self, val=0, next=None):
4 #         self.val = val
5 #         self.next = next
6 class Solution:
7     def middleNode(self, head):
8         fast = head
9         slow = head
10        while fast and fast.next is not None:
11            slow = slow.next
12            fast = fast.next.next
13        return slow
  
```

Saved

Ln 7, Col 32

Testcase Test Result

Accepted Runtime: 62 ms

Case 1 Case 2

Input

head =  
[1,2,3,4,5]



## 206. Reverse Linked List

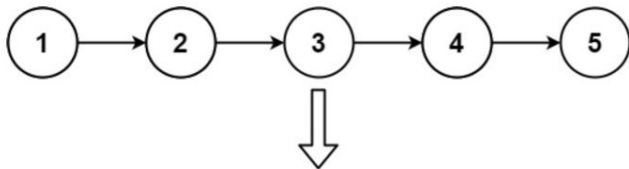
Solved ✓

Easy

Topics

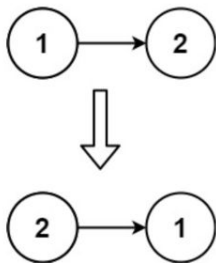
Companies

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:**

&lt;/&gt; Code

Python3 🔒 Auto

≡ 📖 {} ↻

```
1 # Definition for singly-linked list.
2 # class ListNode:
3 #     def __init__(self, val=0, next=None):
4 #         self.val = val
5 #         self.next = next
6 class Solution:
7     def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:
8
```

Saved

Ln 1,

✓ Testcase | &gt;\_ Test Result

Case 1

Case 2

Case 3

+

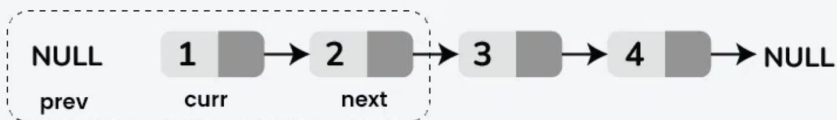
head =

[1,2,3,4,5]

Head  
↓



**02** Step | Initialize prev pointer = NULL  
Store next = curr.next  
Update curr.next = prev



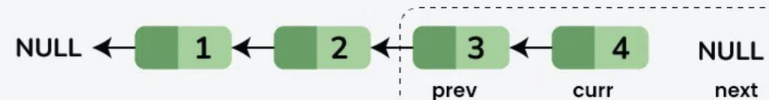
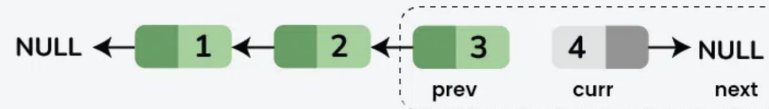
**03** Step | Update prev = curr and curr = next  
Store next node 3 as next  
Update next pointer of current node 2 to previous node 1.



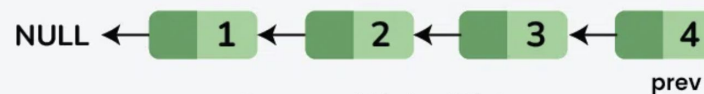
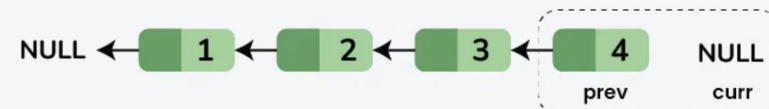
**04** Step | Update prev = curr and curr = next  
Store next node 4 as next  
Update next pointer of current node 3 to previous node 2.



**05** Step | Update prev = curr and curr = next  
next pointer points to NULL  
Update next pointer of current node 4 to previous node 3.



**06** Step | Update prev = curr and curr = next  
Finally, curr becomes NULL and prev stores the head of the reversed linked list

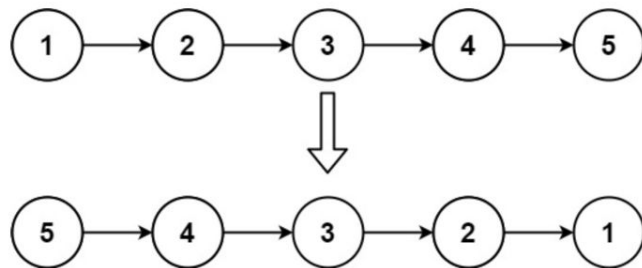


## 206. Reverse Linked List

Easy Topics Companies

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

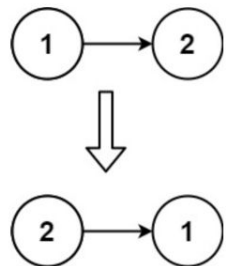
Example 1:



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

Output: [5,4,3,2,1]

Example 2:



Solved

</> Code

Python3 Auto

```
6 class Solution:
7     def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:
8         # Initialize three pointers: curr, prev and next
9         curr = head
10        prev = None
11
12        # Traverse all the nodes of Linked List
13        while curr is not None:
14            # Store next
15            next_node = curr.next
16
17            # Reverse current node's next pointer
18            curr.next = prev
19
20            # Move pointers one position ahead
21            prev = curr
22            curr = next_node
23
24        # Return the head of reversed linked list
25        return prev
```

Saved

Testcase Test Result

Accepted Runtime: 43 ms

Case 1 Case 2 Case 3

Input

head =  
[1,2,3,4,5]

## 23. Merge k Sorted Lists

Hard Topics Companies

You are given an array of  $k$  linked-lists `lists`, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

### Example 1:

Input: `lists = [[1,4,5],[1,3,4],[2,6]]`

Output: `[1,1,2,3,4,4,5,6]`

Explanation: The linked-lists are:

```
[
  1->4->5,
  1->3->4,
  2->6
]
```

merging them into one sorted list:

`1->1->2->3->4->4->5->6`

### Example 2:

Input: `lists = []`

Output: `[]`

### Example 3:

Input: `lists = [[]]`

Output: `[]`

### Constraints:

- $k == \text{lists.length}$

Solved ✓

### </> Code

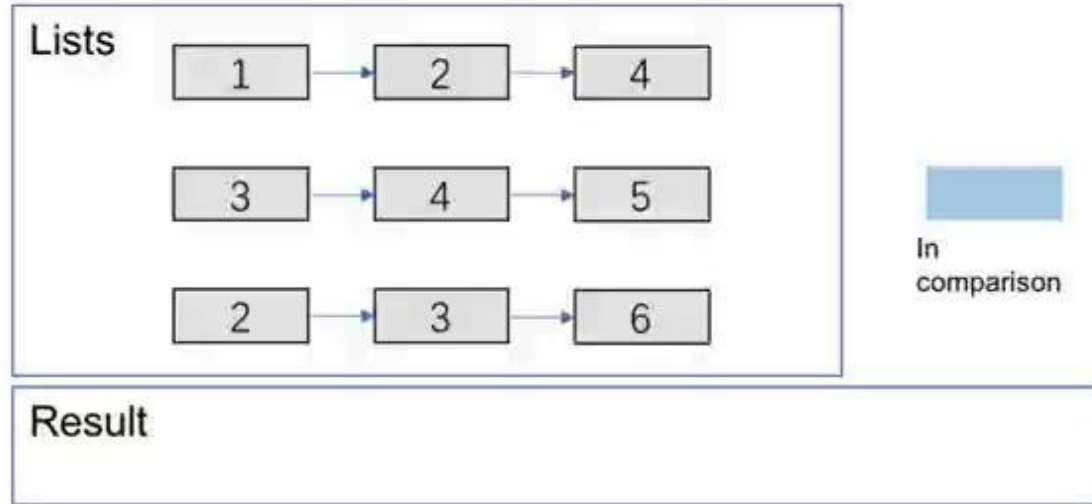
Python3 🔒 Auto

```
1 # Definition for singly-linked list.
2 # class ListNode:
3 #     def __init__(self, val=0, next=None):
4 #         self.val = val
5 #         self.next = next
6 class Solution:
7     def mergeKLists(self, lists: List[Optional[ListNode]]) ->
8         Optional[ListNode]:
```

Saved

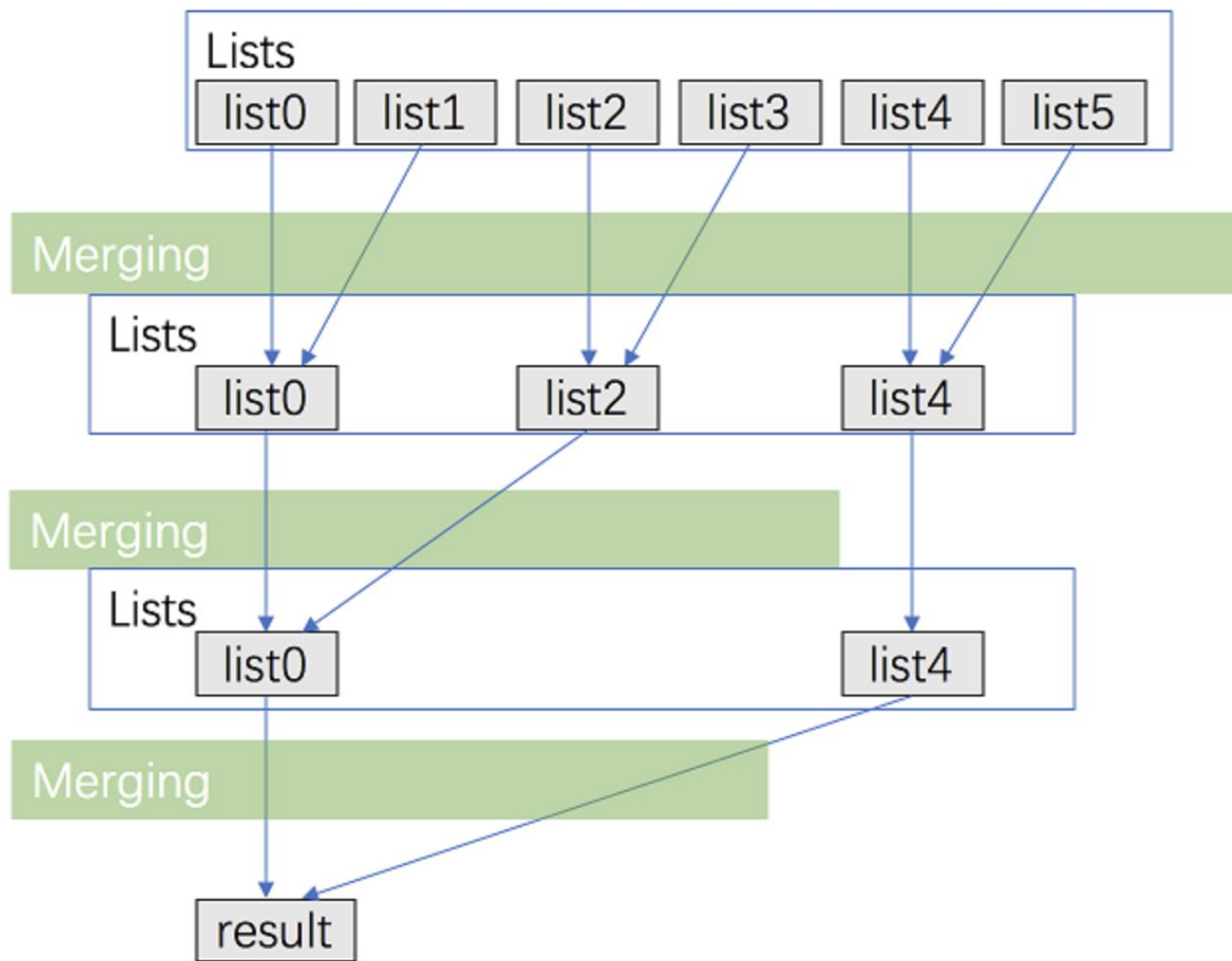
Ln 1, Col 1

✓ Testcase ➤ Test Result



- Time complexity :  $O(kN)$  where  $k$  is the number of linked lists.
  - Almost every selection of node in final linked costs  $O(k)$  ( $k-1$  times comparison).
  - There are  $N$  nodes in the final linked list.





## Input:

- List 1: 1 -> 4 -> 5
- List 2: 1 -> 3 -> 4
- List 3: 2 -> 6
- List 4: 0 -> 7
- List 5: 3 -> 8

### 1. First iteration ( interval = 1):

- Merge List 1 and List 2: 1 -> 1 -> 3 -> 4 -> 4 -> 5
- Merge List 3 and List 4: 0 -> 2 -> 6 -> 7
- List 5 remains unchanged.

After iteration:

- lists[0] = 1 -> 1 -> 3 -> 4 -> 4 -> 5
- lists[1] = 1 -> 3 -> 4
- lists[2] = 0 -> 2 -> 6 -> 7
- lists[3] = 3 -> 8
- lists[4] = 3 -> 8

### 2. Second iteration ( interval = 2):

- Merge List 0 and List 2: 0 -> 1 -> 1 -> 2 -> 3 -> 4 -> 4 -> 5 -> 6 -> 7
- List 5 remains unchanged.

After iteration:

- lists[0] = 0 -> 1 -> 1 -> 2 -> 3 -> 4 -> 4 -> 5 -> 6 -> 7
- lists[1] = 1 -> 3 -> 4
- lists[2] = 0 -> 2 -> 6 -> 7
- lists[3] = 3 -> 8
- lists[4] = 3 -> 8

### 3. Third iteration ( interval = 4):

- Merge List 0 and List 4: 0 -> 1 -> 1 -> 2 -> 3 -> 3 -> 4 -> 4 -> 5 -> 6 -> 7 -> 8

Final result:

- lists[0] = 0 -> 1 -> 1 -> 2 -> 3 -> 3 -> 4 -> 4 -> 5 -> 6 -> 7 -> 8

## 23. Merge k Sorted Lists

Solved

Hard Topics Companies

You are given an array of  $k$  linked-lists `lists`, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

- Time complexity :  $O(N \log k)$  where  $k$  is the number of linked lists.
  - We can merge two sorted linked list in  $O(n)$  time where  $n$  is the total number of nodes in two lists.

</> Code

Python3 Auto

```

6 class Solution:
7     def mergeKLists(
8         self, lists: List[Optional[ListNode]]
9     ) -> Optional[ListNode]:
10         amount = len(lists)
11         interval = 1
12         # Repeatedly merge pairs of lists in intervals of powers of 2
13         while interval < amount:
14             for i in range(0, amount - interval, interval * 2):
15                 lists[i] = self.merge2Lists(lists[i], lists[i + interval])
16                 interval *= 2 # Double the interval in each iteration
17
18         return lists[0] if amount > 0 else None # Return merged list or None if empty
19
20     def merge2Lists(self, l1, l2):
21         head = point = ListNode(0) # Dummy node to simplify list merging
22         while l1 and l2:
23             if l1.val <= l2.val:
24                 point.next = l1
25                 l1 = l1.next
26             else:
27                 point.next = l2
28                 l2 = l1
29                 l1 = point.next.next
30             point = point.next # Move the pointer forward
31
32         # Append any remaining elements from the non-empty list
33         if not l1:
34             point.next = l2
35         else:
36             point.next = l1
37
38         return head.next # Return merged list, skipping the dummy node
    
```

## 143. Reorder List

Medium Topics Companies

You are given the head of a singly linked-list. The list can be represented as:

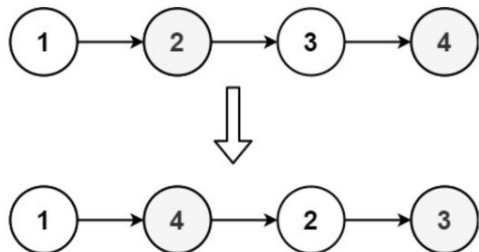
$L_0 \rightarrow L_1 \rightarrow \dots \rightarrow L_{n-1} \rightarrow L_n$

Reorder the list to be on the following form:

$L_0 \rightarrow L_n \rightarrow L_1 \rightarrow L_{n-1} \rightarrow L_2 \rightarrow L_{n-2} \rightarrow \dots$

You may not modify the values in the list's nodes. Only nodes themselves may be changed.

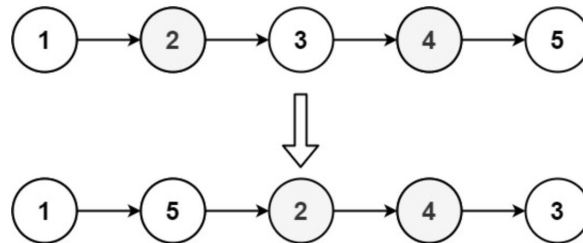
Example 1:



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

Solved

Example 2:



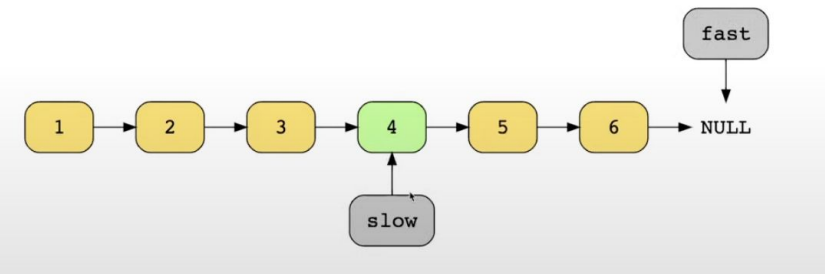
Input: head = [1,2,3,4,5]  
Output: [1,5,2,4,3]

Saved

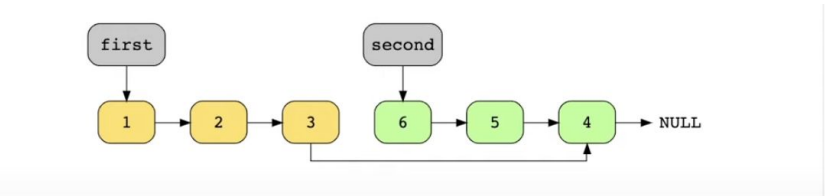
Ln 11, Col 12

Testcase Test Result

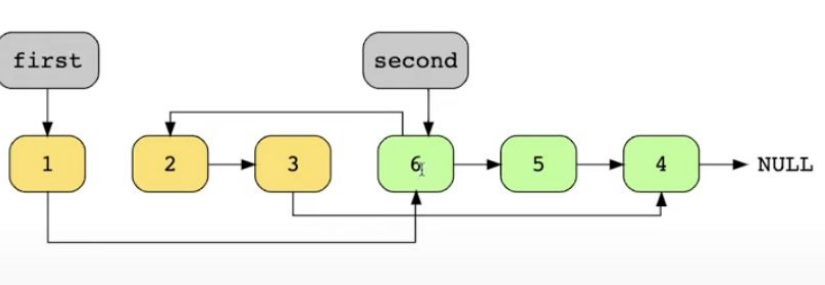
1. Find the middle of the linked list using the fast and slow pointers



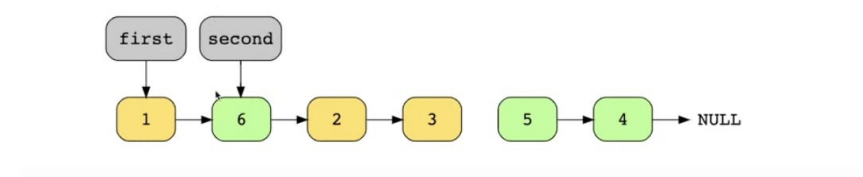
2. Reverse the second half of the linked list using in-place reversal



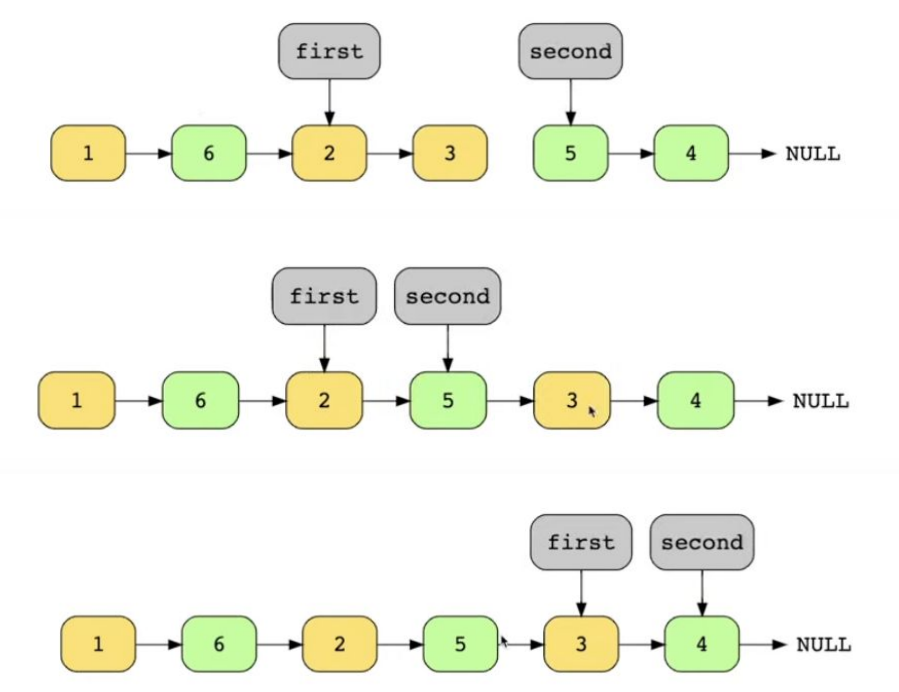
3. Point first.next at second and second.next at first.next



4. After first iteration



5. Move first and second forward



## 143. Reorder List

Medium Topics Companies

You are given the head of a singly linked-list. The list can be represented as:

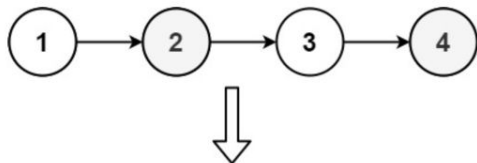
$L_0 \rightarrow L_1 \rightarrow \dots \rightarrow L_{n-1} \rightarrow L_n$

Reorder the list to be on the following form:

$L_0 \rightarrow L_n \rightarrow L_1 \rightarrow L_{n-1} \rightarrow L_2 \rightarrow L_{n-2} \rightarrow \dots$

You may not modify the values in the list's nodes. Only nodes themselves may be changed.

Example 1:



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

Example 2:



Solved

Code

Python3 Auto

```
6 class Solution:
7     def reorderList(self, head: Optional[ListNode]) -> None:
8         """
9         Do not return anything, modify head in-place instead.
10        """
11        if not head:
12            return
13        #find the middle node
14        slow = fast = head
15
16        while fast and fast.next:
17            slow = slow.next
18            fast = fast.next.next
19
20        #reverse the second part of the list
21        prev, curr = None, slow
22        while curr:
23            curr.next, prev, curr = prev, curr, curr.next
24        #merge the two sorted lists
25        first, second = head, prev
26        while second.next:
27            first.next, first = second, first.next
28            second.next, second = first, second.next
```

Saved

Ln 28, Col 53

Testcase Test Result

Accepted Runtime: 39 ms

Case 1 Case 2

## 36. Valid Sudoku

Medium Topics Companies

Determine if a  $9 \times 9$  Sudoku board is valid. Only the filled cells need to be validated **according to the following rules**:

- Each row must contain the digits 1–9 without repetition.
- Each column must contain the digits 1–9 without repetition.
- Each of the nine  $3 \times 3$  sub-boxes of the grid must contain the digits 1–9 without repetition.

### Note:

- A Sudoku board (partially filled) could be valid but is not necessarily solvable.
- Only the filled cells need to be validated according to the mentioned rules.

### Example 1:

5	3		7					
6			1	9	5			
	9	8					6	
8			6					3
4		8		3				1
7			2					6
	6				2	8		
			4	1	9			5
			8			7	9	

Input: board =  
[[["5","3",".", ".", ".", "7",".", ".", ".", ".", "."],  
["6",".", ".", ".", ".", "1","9","5",".", ".", ".", "."],  
[".", ".", "9","8",".", ".", ".", ".", ".", "6",".", "."],  
["8",".", ".", ".", ".", "6",".", ".", ".", ".", "3"],  
["4",".", ".", ".", "8",".", ".", "3",".", ".", ".", "1"],  
["7",".", ".", ".", ".", "2",".", ".", ".", ".", "6"],  
[".", ".", "6",".", ".", ".", ".", ".", "2","8",".", "."],  
[".", ".", ".", ".", "4","1","9",".", ".", ".", "5"],  
[".", ".", ".", ".", "8",".", ".", ".", "7","9","."]]  
Output: true

Solved 🟢

### </> Code

Python 🔒 Auto

```
1 class Solution(object):  
2     def isValidSudoku(self, board):  
3         """"  
4         :type board: List[List[str]]  
5         :rtype: bool  
6         """"  
7
```

### Example 2:

Input: board =

```
[["8","3",".", ".", ".", "7",".", ".", ".", ".", "."]  
 , ["6",".", ".", ".", "1","9","5",".", ".", ".", "."]  
 , [".","9","8",".", ".", ".", ".", ".", "6",".", "."]  
 , ["8",".", ".", ".", ".", "6",".", ".", ".", ".", "3"]  
 , ["4",".", ".", ".", "8",".", ".", "3",".", ".", ".", "1"]  
 , ["7",".", ".", ".", ".", "2",".", ".", ".", ".", "6"]  
 , [".","6",".", ".", ".", ".", ".", "2","8",".", "."]  
 , [".",".", ".", ".", "4","1","9",".", ".", ".", "5"]  
 , [".",".", ".", ".", ".", "8",".", ".", ".", "7","9"]]
```

Output: false

Explanation: Same as Example 1, except with the 5 in the top left corner being modified to 8. Since there are two 8's in the top left 3x3 sub-box, it is invalid.



## 36. Valid Sudoku

Medium

Topics

Companies

Determine if a  $9 \times 9$  Sudoku board is valid. Only the filled cells need to be validated **according to the following rules**:

- Each row must contain the digits 1–9 without repetition.
- Each column must contain the digits 1–9 without repetition.
- Each of the nine  $3 \times 3$  sub-boxes of the grid must contain the digits 1–9 without repetition.

### Note:

- A Sudoku board (partially filled) could be valid but is not necessarily solvable.
- Only the filled cells need to be validated according to the mentioned rules.

### Example 1:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Input: board =

```
[["5","3",".",".","7",".",".",".","."],  
 ["6",".",".","1","9","5",".",".","."],  
 ["9","8",".",".",".","6",".","."],  
 ["8",".",".","6",".",".","3","."],  
 ["4",".","8","3",".","1","."],  
 ["7",".","2",".","6","."],  
 ["6",".","2","8","."],  
 ["4","1","9","5","."],  
 ["8","7","9"]]
```

$r // 3$ : This part divides the row index  $r$  by 3 using integer division. It determines which of the three rows of  $3 \times 3$  boxes the cell is in.

$c // 3$ : determines which of the three columns of  $3 \times 3$  boxes the cell is in



## 36. Valid Sudoku

Solved

Medium Topics Companies

Determine if a  $9 \times 9$  Sudoku board is valid. Only the filled cells need to be validated **according to the following rules**:

- Each row must contain the digits 1–9 without repetition.
- Each column must contain the digits 1–9 without repetition.
- Each of the nine  $3 \times 3$  sub-boxes of the grid must contain the digits 1–9 without repetition.

### Note:

- A Sudoku board (partially filled) could be valid but is not necessarily solvable.
- Only the filled cells need to be validated according to the mentioned rules.

### Example 1:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Input: board =

## Code

Python Auto

```
1 class Solution(object):
2     def isValidSudoku(self, board):
3         N = 9
4         # Manually initialize lists of sets using a loop
5         rows = []
6         cols = []
7         boxes = []
8         for _ in range(N):
9             rows.append(set()) # Tracks numbers in each row
10            cols.append(set()) # Tracks numbers in each column
11            boxes.append(set()) # Tracks numbers in each 3x3 box
12
13        for r in range(N):
14            for c in range(N):
15                val = board[r][c]
16                if val == ".": # Skip empty cells
17                    continue
18
19                # (r // 3) * 3 + c // 3 maps this row-column combination to a single index in the range [0, 8]
20                box_index = (r // 3) * 3 + (c // 3)
21                # Check if value already exists in row, column, or box
22                if val in rows[r]:
23                    return False # Duplicate in the same row
24                if val in cols[c]:
25                    return False # Duplicate in the same column
26                if val in boxes[box_index]:
27                    return False # Duplicate in the same 3x3 box
28
29                # Add value to respective row, column, and box
30                rows[r].add(val)
31                cols[c].add(val)
32                boxes[box_index].add(val)
33
34        return True
```

Saved

Ln 1, Col 1



## 48. Rotate Image

Solved ✓

Medium

Topics


Companies

You are given an  $n \times n$  2D matrix representing an image, rotate the image by 90 degrees (clockwise).

You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

### Example 1:

1	2	3
4	5	6
7	8	9




7	4	1
8	5	2
9	6	3

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [[7,4,1],[8,5,2],[9,6,3]]

### Example 2:

5	1	9	11
2	4	8	10
13	3	6	7
15	14	12	16



15	13	2	5
14	3	4	1
12	6	8	9
16	7	10	11

### Code

Python Auto



```
1 class Solution(object):
2     def rotate(self, matrix):
3
```

Saved

Ln 3, Col 1

☑️ Testcase ➤ Test Result

**Accepted** Runtime: 12 ms

• Case 1 • Case 2

Input

matrix =

Original

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

Transpose



Transposed

	0	1	2	3	4
0	1	6	11	16	21
1	2	7	12	17	22
2	3	8	13	18	23
3	4	9	14	19	24
4	5	10	15	20	25

Reverse



Transposed+Reversed

	0	1	2	3	4
0	21	16	11	6	1
1	22	17	12	7	2
2	23	18	13	8	3
3	24	19	14	9	4
4	25	20	15	10	5

Rotated

	0	1	2	3	4
0	21	16	11	6	1
1	22	17	12	7	2
2	23	18	13	8	3
3	24	19	14	9	4
4	25	20	15	10	5

=

Equal

Transposed+Reversed

	0	1	2	3	4
0	21	16	11	6	1
1	22	17	12	7	2
2	23	18	13	8	3
3	24	19	14	9	4
4	25	20	15	10	5



## 48. Rotate Image

Solved 🟢

Medium Topics Companies

You are given an  $n \times n$  2D matrix representing an image, rotate the image by 90 degrees (clockwise).

You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

Example 1:

1	2	3
4	5	6
7	8	9

 → 

7	4	1
8	5	2
9	6	3

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [[7,4,1],[8,5,2],[9,6,3]]

Example 2:

5	1	9	11
2	4	8	10
13	3	6	7
15	14	12	16

 → 

15	13	2	5
14	3	4	1
12	6	8	9
16	7	10	11

## &lt;/&gt; Code

Python Auto

🔖 {} ↺ ↻

```
1 class Solution(object):
2     def rotate(self, matrix):
3         self.transpose(matrix)
4         self.reflect(matrix)
5
6     def transpose(self, matrix):
7         n = len(matrix)
8         for i in range(n):
9             for j in range(i + 1, n):
10                matrix[j][i], matrix[i][j] = matrix[i][j], matrix[j][i]
11
12    def reflect(self, matrix):
13        n = len(matrix)
14        for i in range(n):
15            for j in range(n // 2):
16                matrix[i][j], matrix[i][n - j - 1] = (
17                    matrix[i][n - j - 1],
18                    matrix[i][j],
19                )
```

Saved

Ln 19, Col 18

☑ Testcase &gt; Test Result

Accepted Runtime: 12 ms

• Case 1

• Case 2

Input