Sort List

# Index	148
CreatedAt	@September 28, 2022
Person	A Ally Hyeseong Kim
Status	In Progress
Tags	Python Sorting
UpdatedAt	@September 28, 2022

References

LeetCode - The World's Leading Online Programming Learning Platform

Level up your coding skills and quickly land a job. This is the best place to expand your knowledge and get prepared for your next interview.

https://leetcode.com/problems/sort-list/

파이썬 알고리즘 인터뷰

2021 세종도서 학술부문 선정작. 현업과 실무에 유용한 주요 알고리즘 이론을 깊숙이 이해하고, 파이썬의 핵심 기능과 문법까지 상세하게 이해할 수 있는 취업용 코딩 테스트를 위한 완벽 가이드다. 200여 개가 넘는...

https://www.aladin.co.kr/shop/wproduct.aspx?ItemId=245495826

1. Merge Sort Algorithm

```
class Solution:
    def mergeList(self, l1: ListNode, l2: ListNode) -> ListNode:
        if l1 and l2:
            if l1.val > l2.val:
                l1, l2 = l2, l1
            l1.next = self.mergeList(l1.next, l2)

        return l1 or l2

    def sortList(self, head: Optional[ListNode]) -> Optional[ListNode]:
        if not (head and head.next):
            return head

        half, slow, fast = None, head, head
        while fast and fast.next:
            half, slow, fast = slow, slow.next, fast.next.next
        half.next = None

        l1 = self.sortList(head)
        l2 = self.sortList(slow)

        return self.mergeList(l1, l2)
```

- `Runner` Method: Using 2 pointers at the same time when traversing Linked List. One pointer goes ahead of the other.

```
half, slow, fast = None, head, head
while fast and fast.next:
    half, slow, fast = slow, slow.next, fast.next.next
half.next = None
```

  - `Fast runner` step = 2 * `Slow runner` step
  - When `Fast runner` reaches the end of the list, `Slow runner` reaches exactly the middle.

2. Quick Sort Algorithm

- It is difficult to set the `pivot` to the desired one.
- If the list is already sorted, it goes on in an unbalanced list.

3. Python Sort: Time Sort Algorithm

```
class Solution:
    def sortList(self, head: Optional[ListNode]) -> Optional[ListNode]:
        p = head
        lst: List = []
        while p:
            lst.append(p.val)
            p = p.next

        lst.sort

        p = head
        for i in range(len(lst)):
            p.val = lst[i]
            p = p.next

        return head
```