Python algorithm

임채빈

8장. 연결 리스트

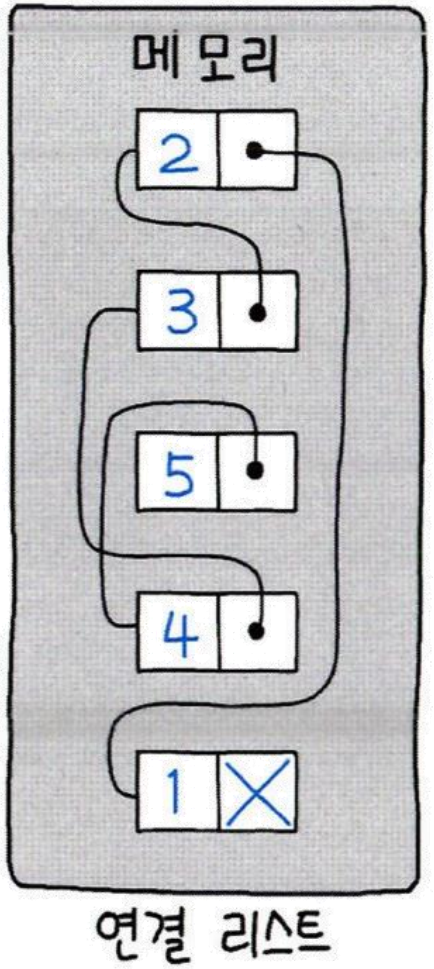
Contents

연결 리스트 (Linked List)

- 동적으로 새로운 노드를 삽입하거나 삭제하기가 간편
- 물리 메모리의 연속적 할당 불필요

- 특정 인덱스에 접근하기 위해 순서대로 읽어야함 $\rightarrow O(n)$
- 시작 또는 끝 지점에 노드를 추가, 삭제, 추출 -> $O(1)$

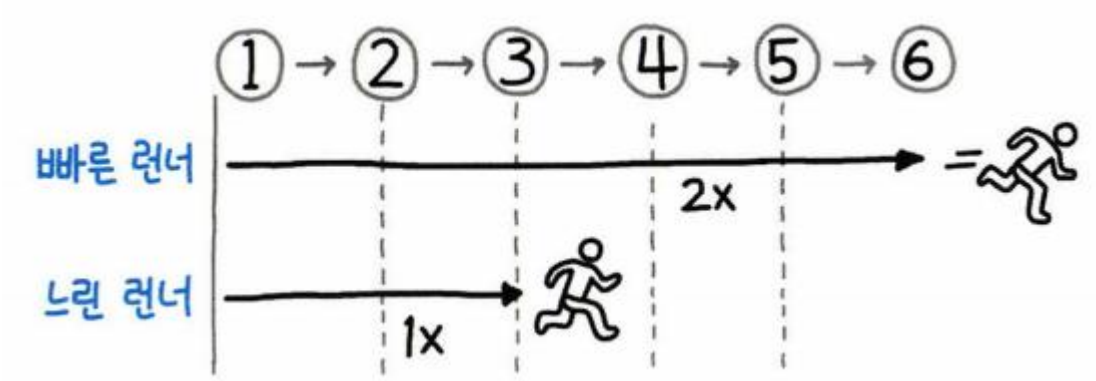
```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
```



런너 기법 (from 13. 팰린드롬 연결 리스트 풀이4 & p210 참고)

- 연결 리스트를 순회할 때 2개의 포인터를 동시에 사용 하는 기법
- 병합 지점, 중간 위치, 길이 등을 판별할 때 사용

- 빠른 런너가 끝에 도달하면 중간 런너는 중간 지점에 위치
- 중간에서부터 값을 뒤집거나 비교



다중 할당 (p211 문법)

- 2개 이상의 변수에 동시에 할당
- 파이써닉한 표현

```
rev, rev.next, slow = slow, rev, slow.next
```

⟶

```
rev, rev.next = slow, rev
slow = slow.next
```

- 가독성을 위해 우측으로 바꿀경우 slow와 rev가 동일한 참조가 되버림

08. 두 정렬 리스트의 병합(Merge Two Sorted Lists)

- 정렬되어 있는 두 연결 리스트를 합쳐라.

• 입력

```
1->2->4, 1->3->4
```

• 출력

```
1->1->2->3->4->4
```

https://leetcode.com/problems/merge-two-sorted-lists/

Unit 02 | 두 정렬 리스트의 병합

02. 두 정렬 리스트의 병합(Merge Two Sorted Lists)

- 1. 리스트와 pop() 사용 (재귀x)

```
class Solution:
    def mergeTwoLists(self, l1: ListNode, l2: ListNode) -> ListNode:
        new = []
        while l1 and l2:
            if l1.val < l2.val:
                new.append(l1.val)
                l1 = l1.next
            else:
                new.append(l2.val)
                l2 = l2.next
        if l1:
            while l1:
                new.append(l1.val)
                l1 = l1.next
        else:
            while l2:
                new.append(l2.val)
                l2 = l2.next
        temp = None
        for i in range(len(new)):
            temp = ListNode(val=new.pop(),next=temp)
        return temp
```

Unit 02 | 두 정렬 리스트의 병합

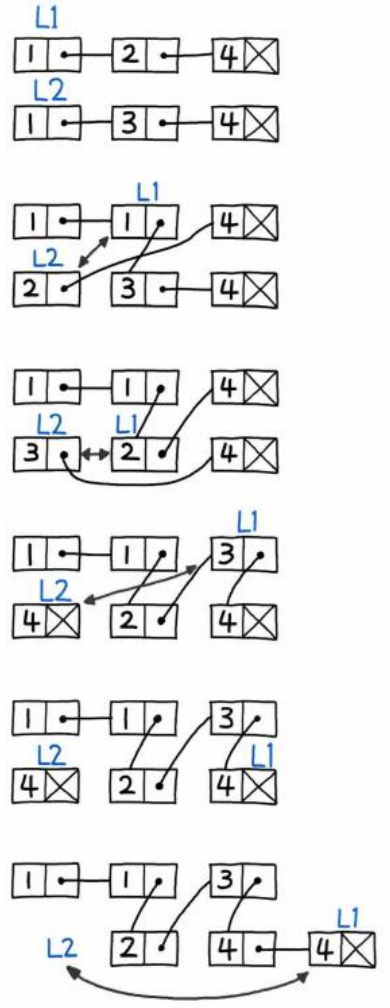
02. 두 정렬 리스트의 병합(Merge Two Sorted Lists)

- 2. 재귀함수와 다중 할당 이용

```
class Solution:
    def mergeTwoLists(self, l1: ListNode, l2: ListNode) -> ListNode:
        if (not l1) or (l2 and l1.val > l2.val):
            l1, l2 = l2, l1
        if l1:
            l1.next = self.mergeTwoLists(l1.next, l2)
        return l1
```

Runtime: 24 ms, faster than 99.78% of Python3 online submissions for Merge Two Sorted Lists.

Memory Usage: 13.8 MB, less than 65.63% of Python3 online submissions for Merge Two Sorted Lists.



Unit 03 | 역순 연결 리스트 2

19. 역순 연결 리스트 2

- 인덱스 m에서 n까지를 역순으로 만들어라. 인덱스 m은 1부터 시작한다.
  - 입력

```
1->2->3->4->5->NULL, m = 2, n = 4
```

  - 출력

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

- 전에 확인해보면 좋을 풀이
  - 15. 역순 연결 리스트
    - 1) 재귀구조로 뒤집기
    - 2) 반복구조로 뒤집기

https://leetcode.com/problems/longest-palindromic-substring/

Unit 03 | 역순 연결 리스트 2

19. 역순 연결 리스트 2

- 1. 리스트와 pop() 사용

```
class Solution:
    def reverseBetween(self, head: ListNode, m: int, n: int) -> ListNode:
        new = []
        while head:
            new.append(head.val)
            head = head.next
        temp = None
        for i in range(len(new),0,-1):
            if i > n:
                temp = ListNode(val=new.pop(),next=temp)
            elif m<=i<=n:
                temp = ListNode(val=new.pop(m-1),next=temp)
            else:
                temp = ListNode(val=new.pop(),next=temp)
        return temp
```

Runtime: 28 ms, faster than 89.61% of Python3 online submissions for Reverse Linked List II.

Memory Usage: 14.1 MB, less than 30.59% of Python3 online submissions for Reverse Linked List II.

Unit 03 | 역순 연결 리스트 2

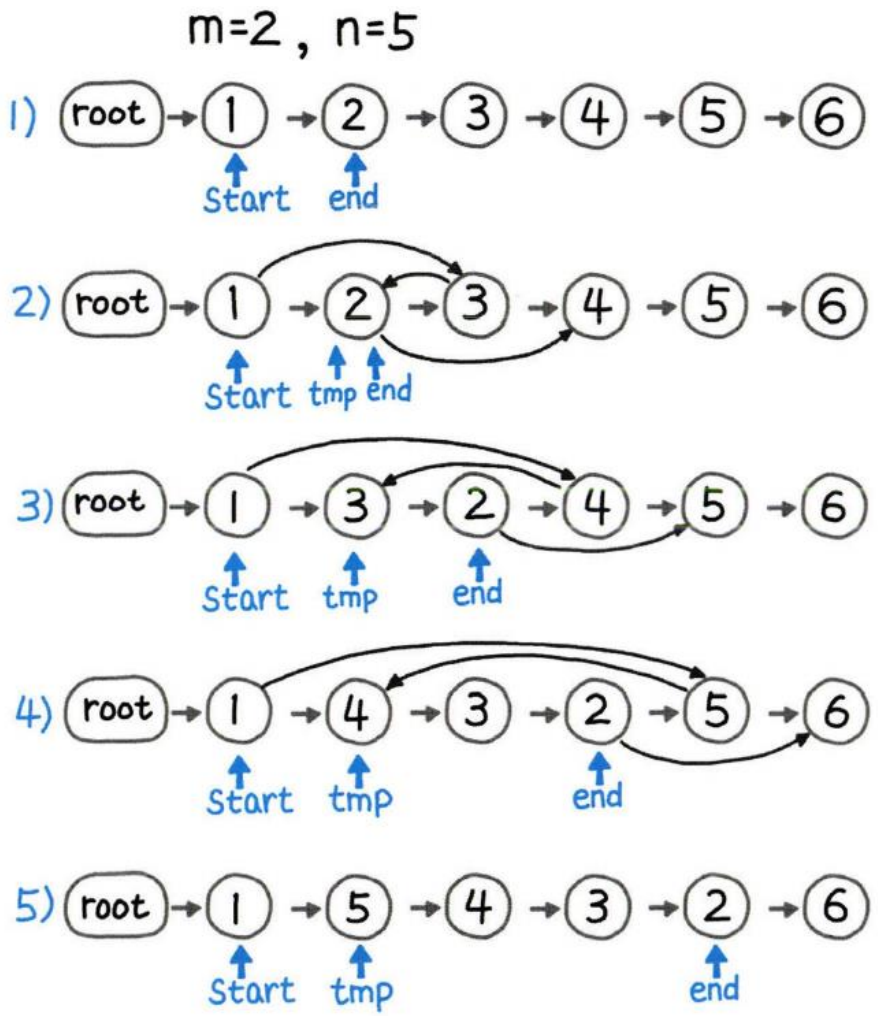
19. 역순 연결 리스트 2

- 2. 반복 구조로 노드 뒤집기

```
class Solution:
    def reverseBetween(self, head: ListNode, m: int, n: int) -> ListNode:
        if not head or m==n:
            return head

        root = start = ListNode(None)
        root.next = head

        for _ in range(m-1):
            start = start.next
        end = start.next

        for _ in range(n-m):
            tmp, start.next, end.next = start.next, end.next, end.next.next
            start.next.next = tmp

        return root.next
```

Runtime: 32 ms, faster than 72.63% of Python3 online submissions for Reverse Linked List II.

Memory Usage: 14 MB, less than 39.42% of Python3 online submissions for Reverse Linked List II.



들어주셔서 감사합니다.