1. AddTwoNumbers
2. Merge k Sorted Lists
3. Reverse LinkedLists

문제 1) AddTwoNumbers

Problem

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

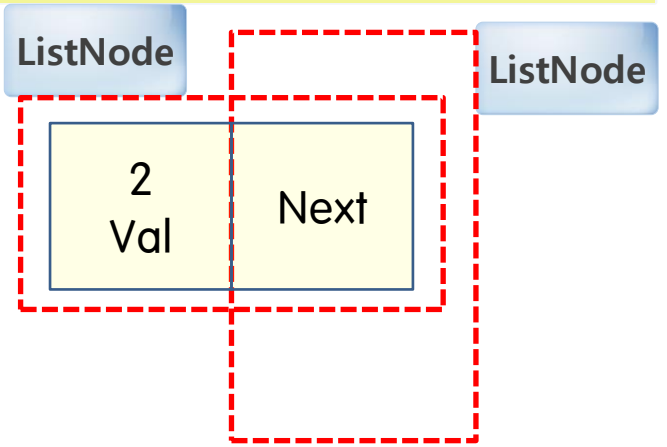
Example:

Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)
Output: 7 -> 0 -> 8
Explanation: 342 + 465 = 807.

AddTwoNumbers

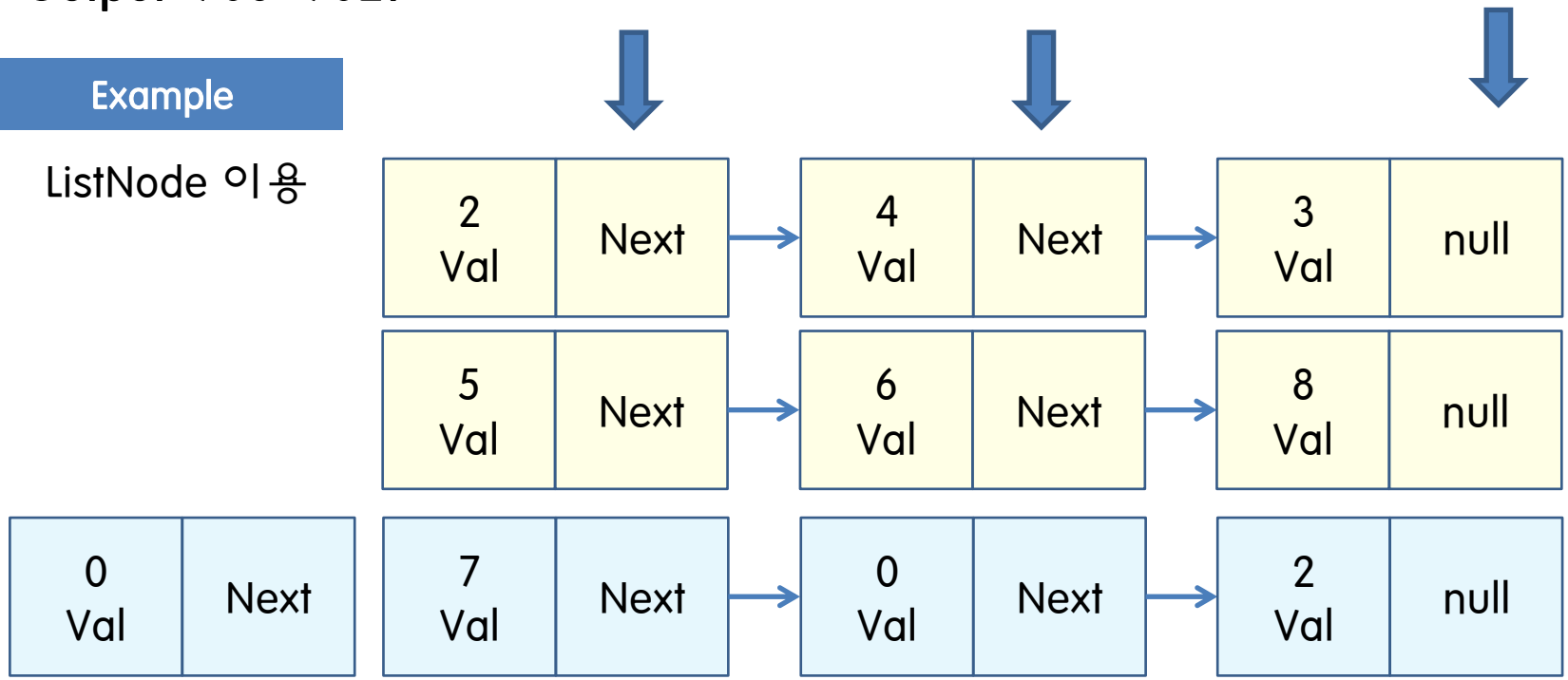
Problem

Input:
2->4->3,
5->6->8
7 0 6
Output: 706 7021



Example

ListNode 이용



Solution

LinkedList

1. 문제를 정확히 이해
2. 알고리즘 정하고 담을 그릇 정한다
3. for 문 돌리기
4. 생각->프로그램(한국말로 생각하고->Java)
   결과를 해석하여 이미지화시킨다

I Can Image

Input:
2->4->3,
5->6->2
7 0 6
Output: 706

1. ListNode 정확히 이해

문제 2)Merge k Sorted Lists

Problem

Merge k sorted linked lists and return it as one sorted list. Analyze and describe its complexity.

Example:

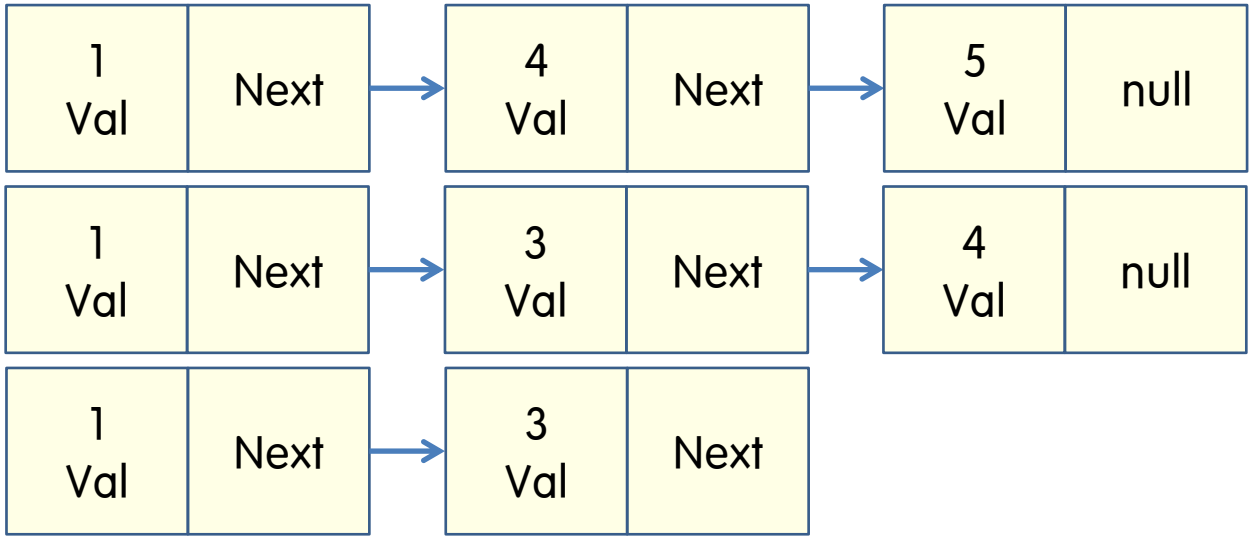
Input:
[
  1->4->5,
  1->3->4,
  2->6
]
Output: 1->1->2->3->4->4->5->6

Merge k Sorted Lists

Problem

Input:
1->4->5,
1->3->4,
2->6
Output: 1->1->2->3->4->4->5->6

Example



Merge k Sorted Lists

Problem

Input:
ListNode list1 = 1->4->5,
ListNode list2 = 1->3->4,
ListNode list3 = 2->6

PriorityQueue<ListNode> queue = **new PriorityQueue<ListNode>(Comp);**

소팅된 queue에 list1 , list2 , list3을 offer하면
[list1 , list2 , list3] 이런 형태로 담겨지겠죠?

List1(1->4->5)를 poll을 해서
Queue.offer

Output: 1->1->2->3->4->4->5->6

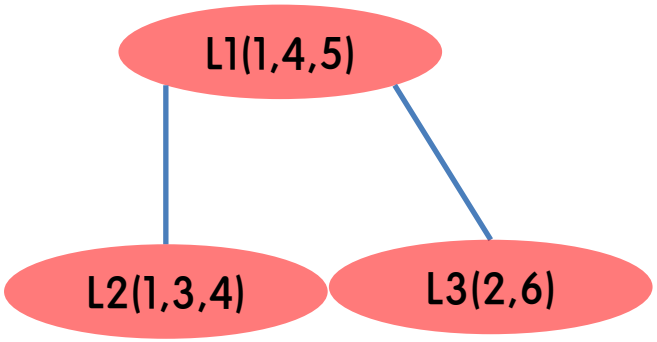
Priority Queue

Queue

MIN HEAP

1. minHeap 구조로 pq를 만듭니다. list1, list2, list3 가 아래 모양처럼 되어 있게됩니다.

1



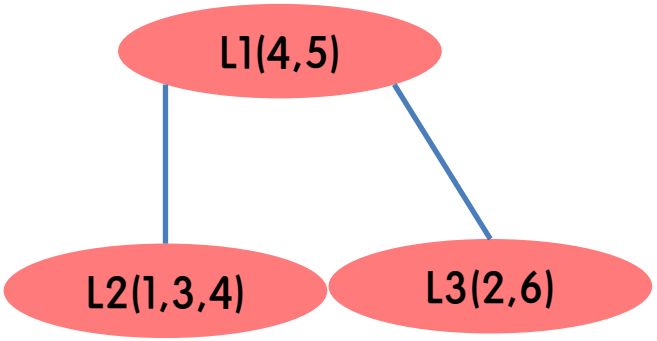
ListNode list1 = 1->4->5,

ListNode list2 = 1->3->4,

ListNode list3 = 2->6

2. 질문 주신 2번째 pq는 아래 처럼 작동합니다.

2



ListNode list1 = 4->5,

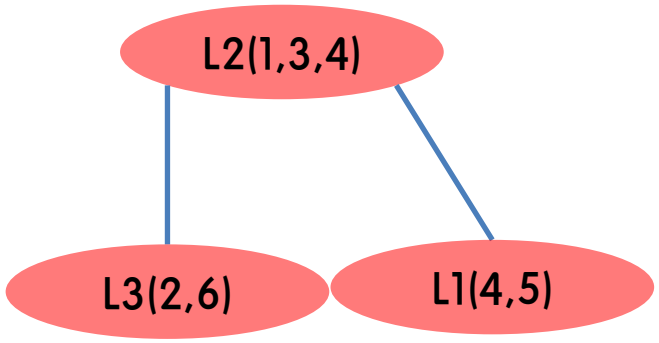
ListNode list2 = 1->3->4,

ListNode list3 = 2->6

1. list1에서 queue.poll()을 했으니까 1을 빼서 담기 시작합니다.
2. list1의 1은 결과를 담을 ListNode p 에 넣습니다.(list p = 1)

3. Pq에 계속해서 offer하면 자체적으로 비교하고 교체해줍니다.(heapy 과정입니다)

3

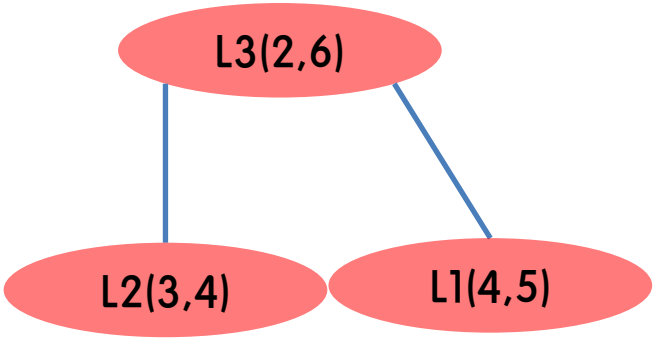


ListNode list1 =4->5,
ListNode list2 = ->3->4,
ListNode list3 = 2->6

list2의 1이 제일 작기때문에 기존 ListNode p 에 담습니다.(listp = 1->1)

4. list3의 2가 제일 작기때문에 heapy과정을 또 거칩니다.

4



ListNode list1 =4->5,
ListNode list2 = 3->4,
ListNode list3 = 2->6

1. list3의 2가 제일 작기때문에 기존 ListNode p 에 담습니다.(listp = 1->1->2)

이런식으로 계속 pq가 작동합니다. 맨위에 제일 작은값을 담아주면 계속 빼넵니다.

더 이상 그림을 안 그려도 안 시겠죠??

Solution

BFS

1. 문제를 정확히 이해
2. 알고리즘 정하고 담을 그릇 정한다
3. for 문 돌리기
4. 생각->프로그램(한국말로 생각하고->Java)
   결과를 해석하여 이미지화시킨다

I Can Image

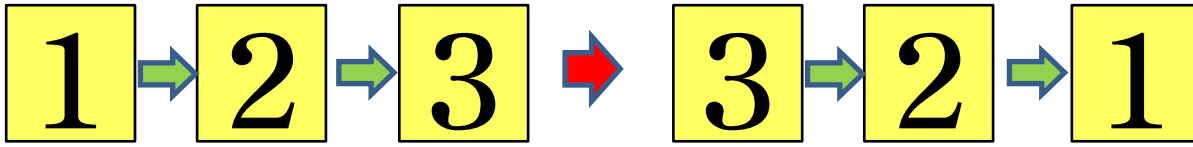
Reverse LinkedLists

설명

단일 연결 리스트가 주어집니다. 리스트를 reverse해서 reversed lists를 리턴하세요
(**Reverse a singly linked list.**)

입출력

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



Input : head = []
Output: []

문제 Format

```
class ListNode{
        int val;
        ListNode next;
        ListNode(int x){
        this.val = x;
}}
class Solution {
    public ListNode solve(ListNode s) {
    }
}
```

제한사항

The number of nodes in the list is the range [0, 5000].
-5000 <= Node.val <= 5000

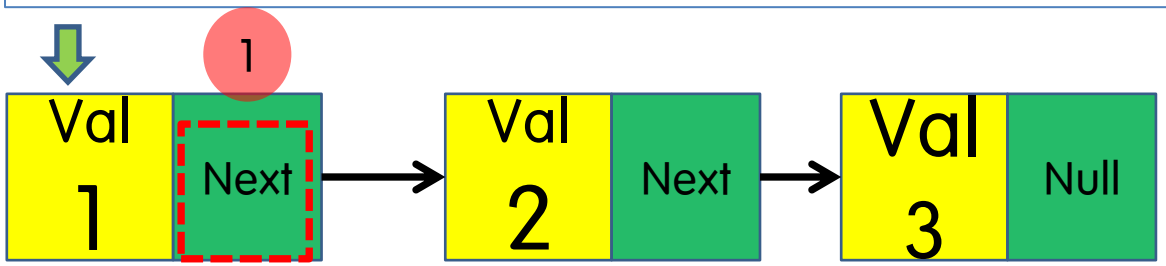
문제분석

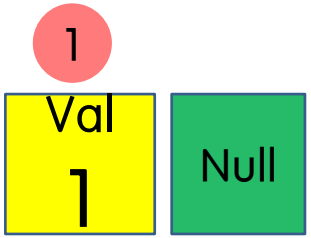
1. singly linked list.

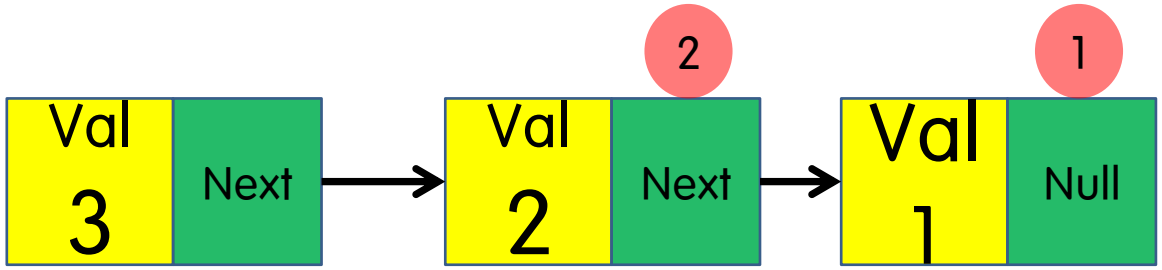
2. doubly linked list.

문제분석

1. Val(1)의 next를 null로 만든다
2. Val(1)을 앞에 next가 가리켜야 한다

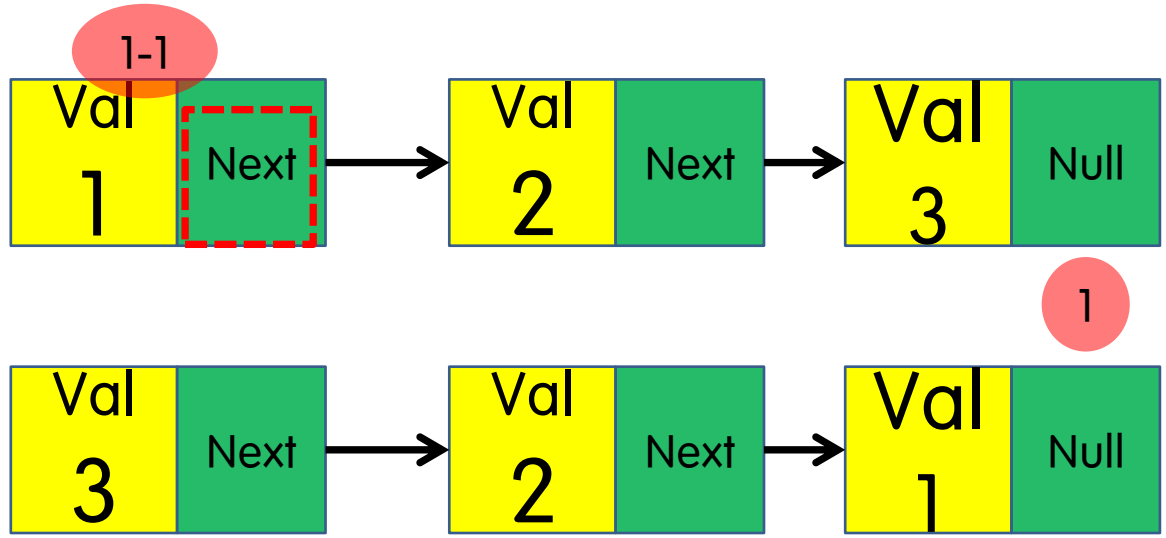






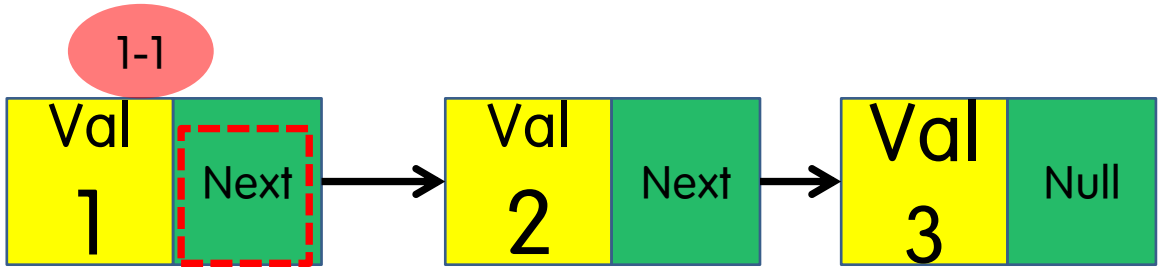
문제분석

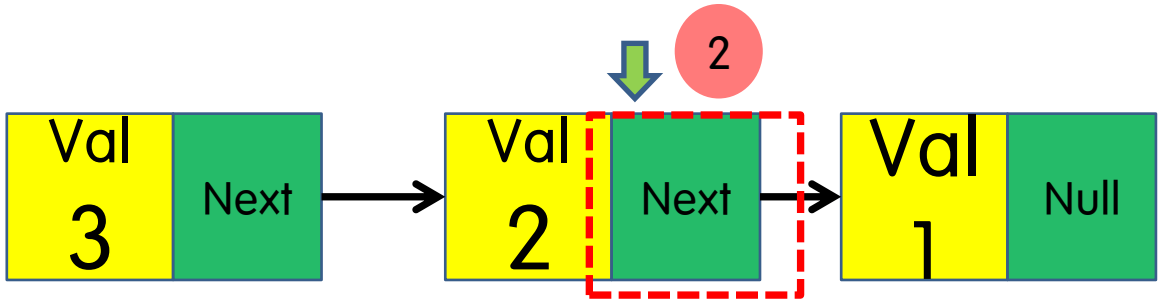
1. Val(1)의 next를 null로 만든다



문제분석

2. Val(1)을 앞에 next가 가리켜야 한다

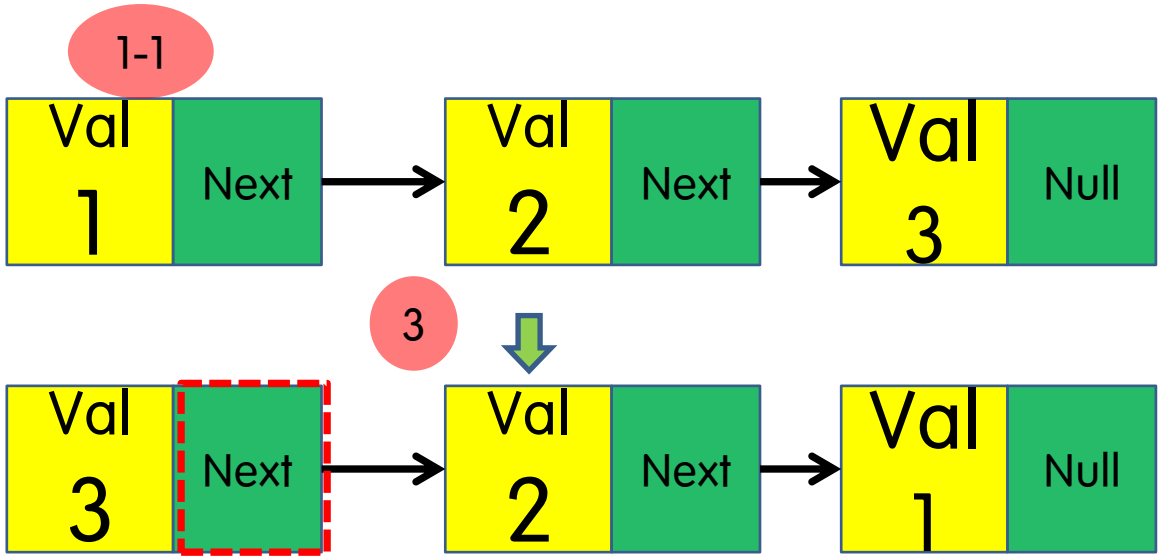




2 cur.next = 1-1 Val 1 Null

문제분석

3. 앞에 next가 뒤에 값들을 가리켜야 한다.

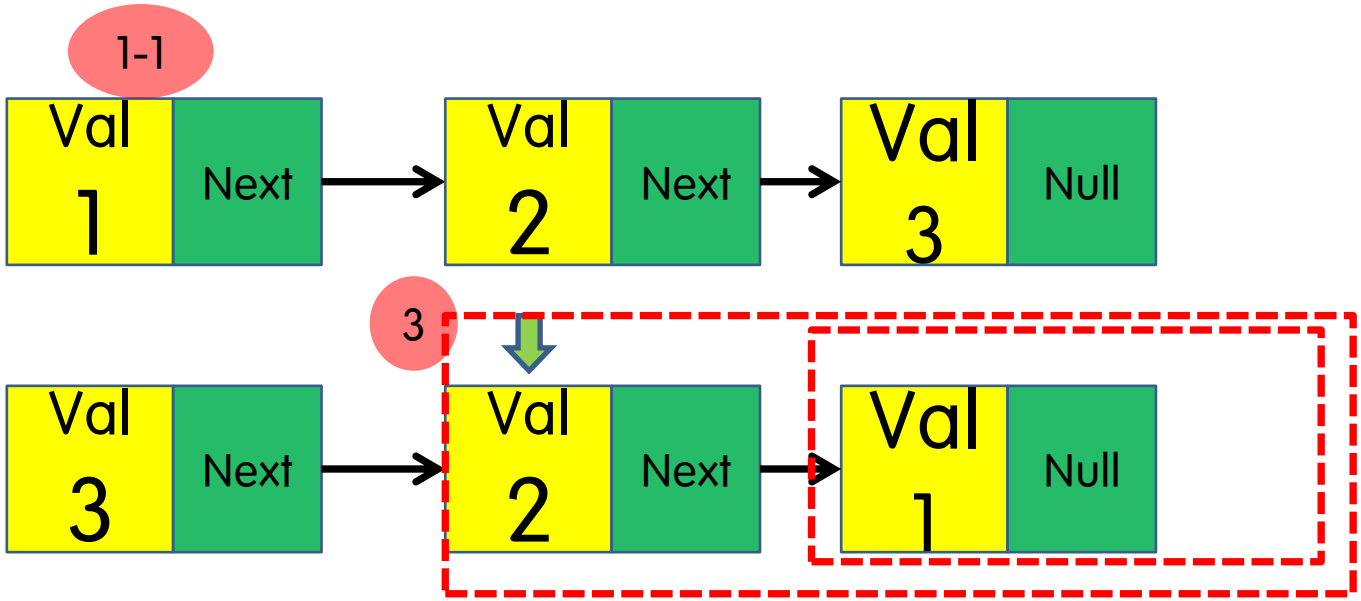


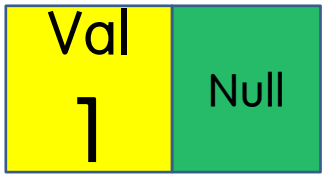
2 cur.next = 1-1 prev

3 prev = cur

문제분석

3. 앞에 next가 뒤에 값들을 가리켜야 한다.



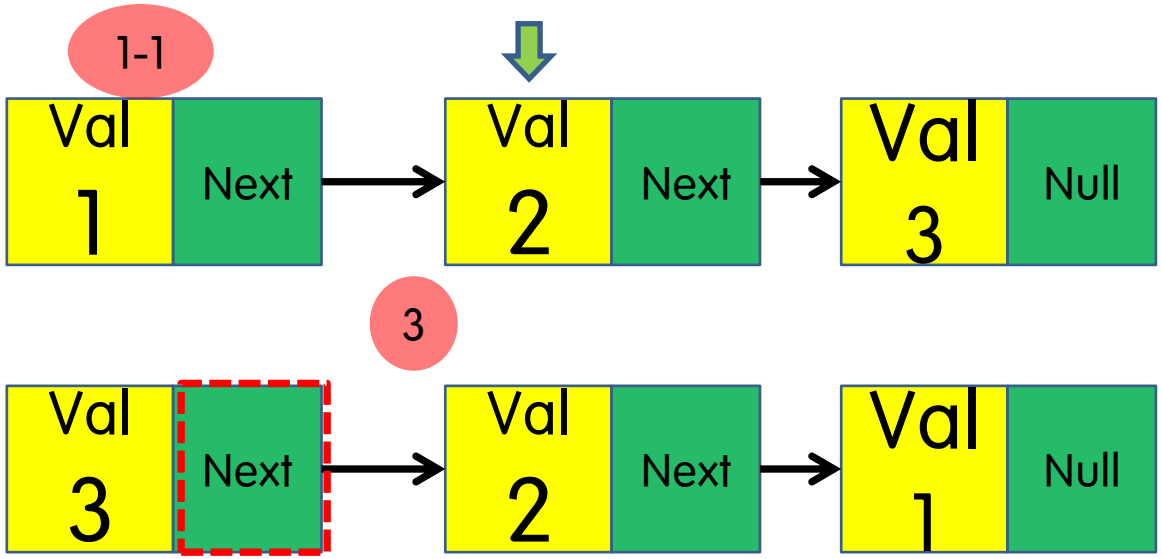
1-1 

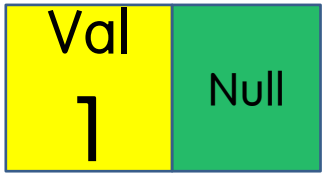
2 cur.next = 1-1 prev

3 prev = cur

문제분석

4. cur 포인터를 한 칸 뒤로 간다



1-1 

next = cur.next

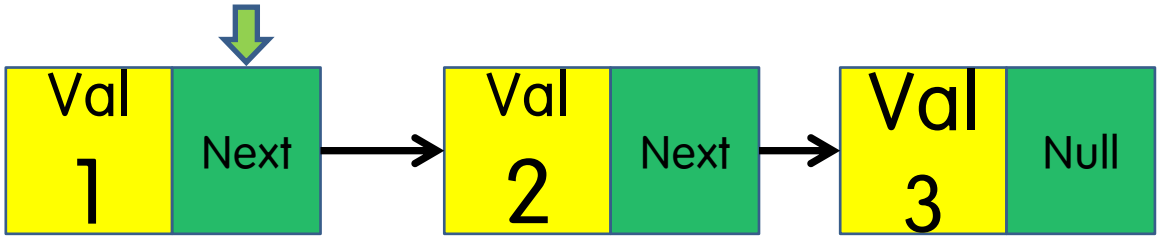
2 cur.next = 1-1 prev

3 prev = cur

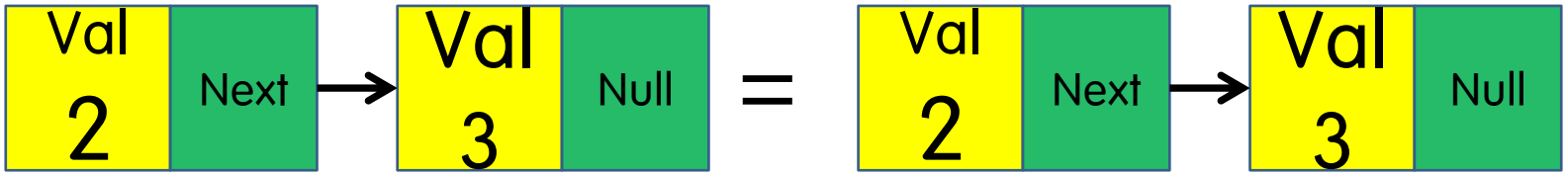
4 cur = next

문제분석

1. Cur의 next



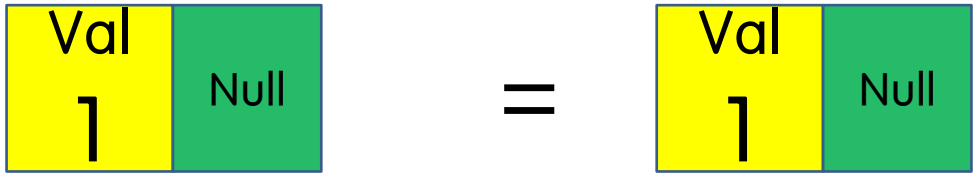
1



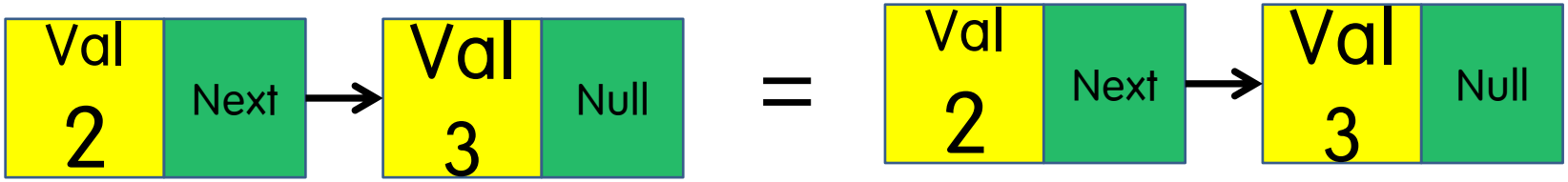
2

null = prev(null)

3

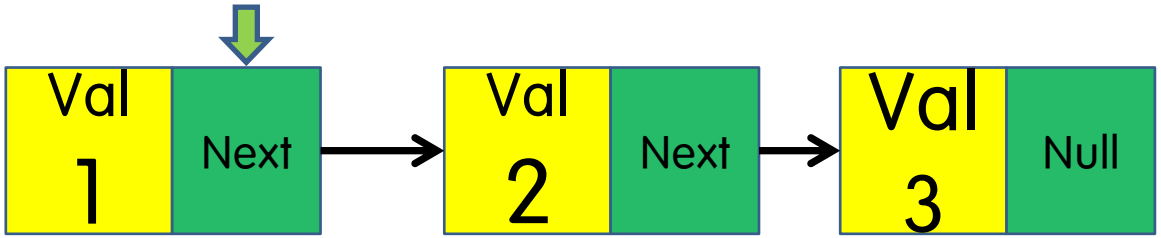


4

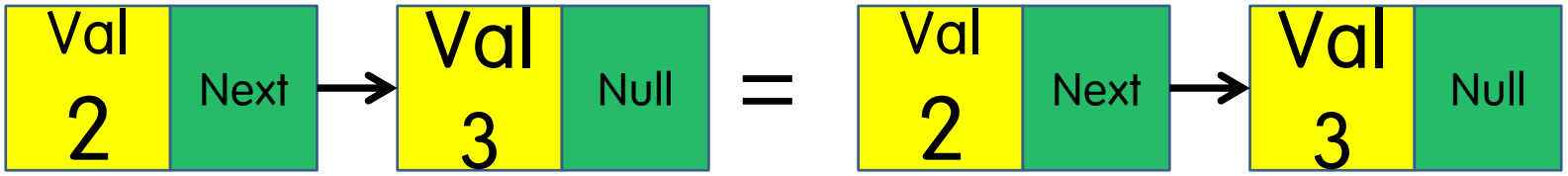


문제분석

1. Cur의 next



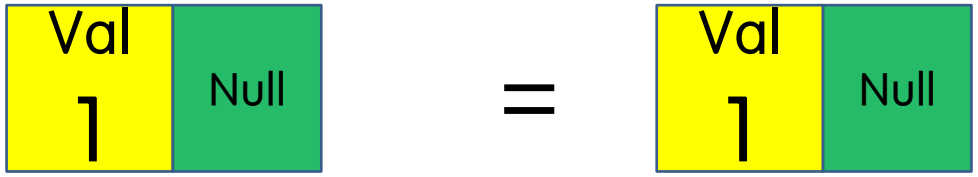
1



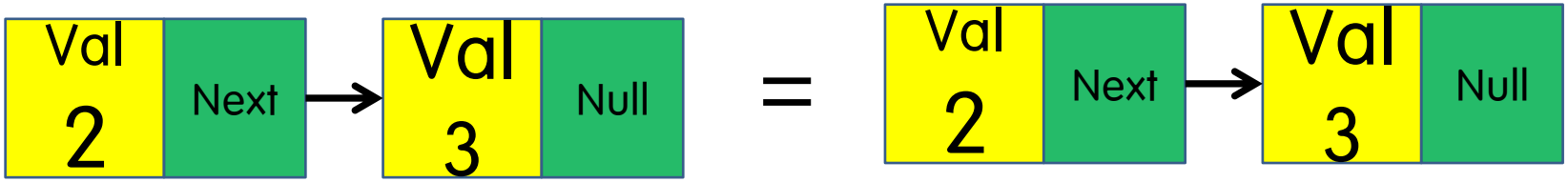
2

null = prev(null)

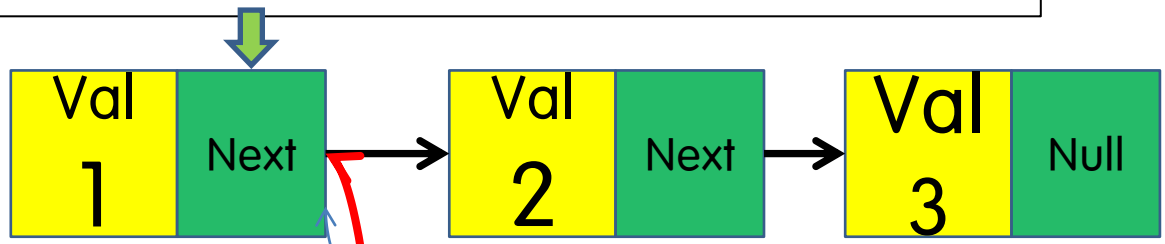
3

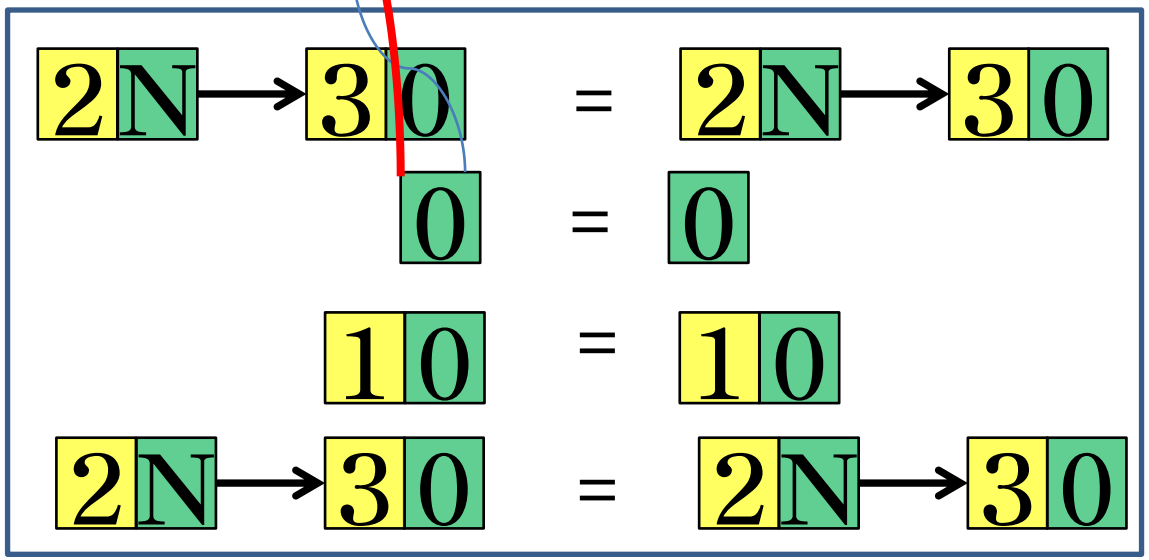


4

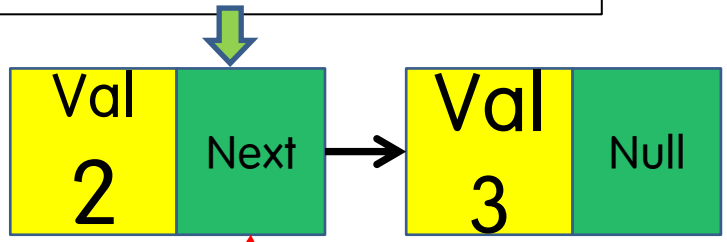


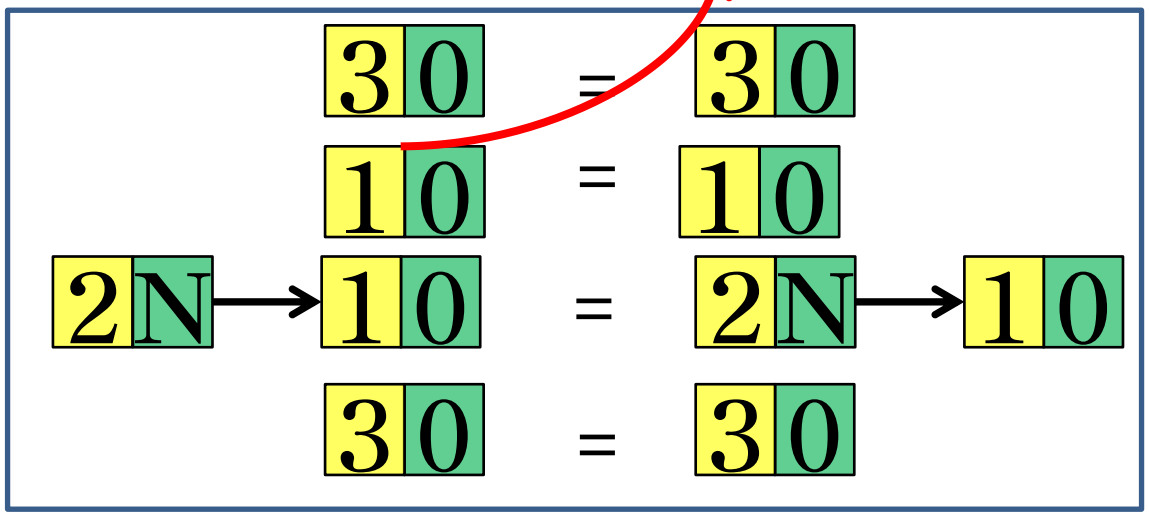
요약



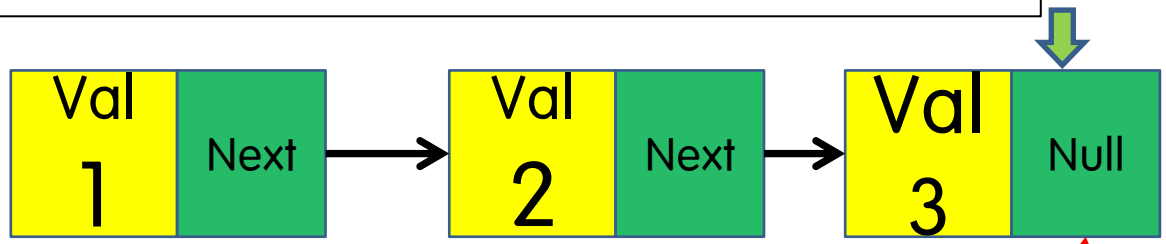


요약





요약



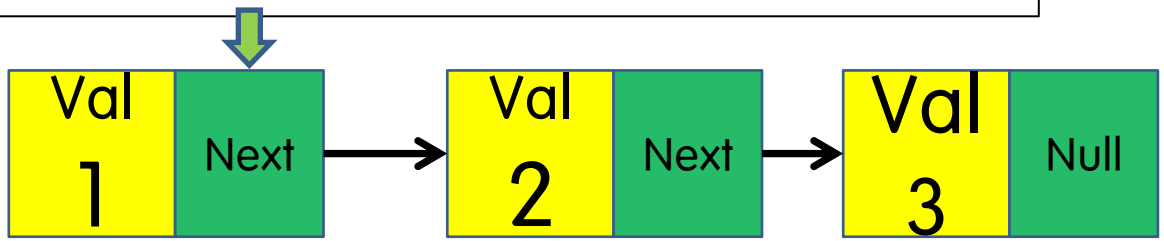
0 = 0

2N → 10 = 2N → 10

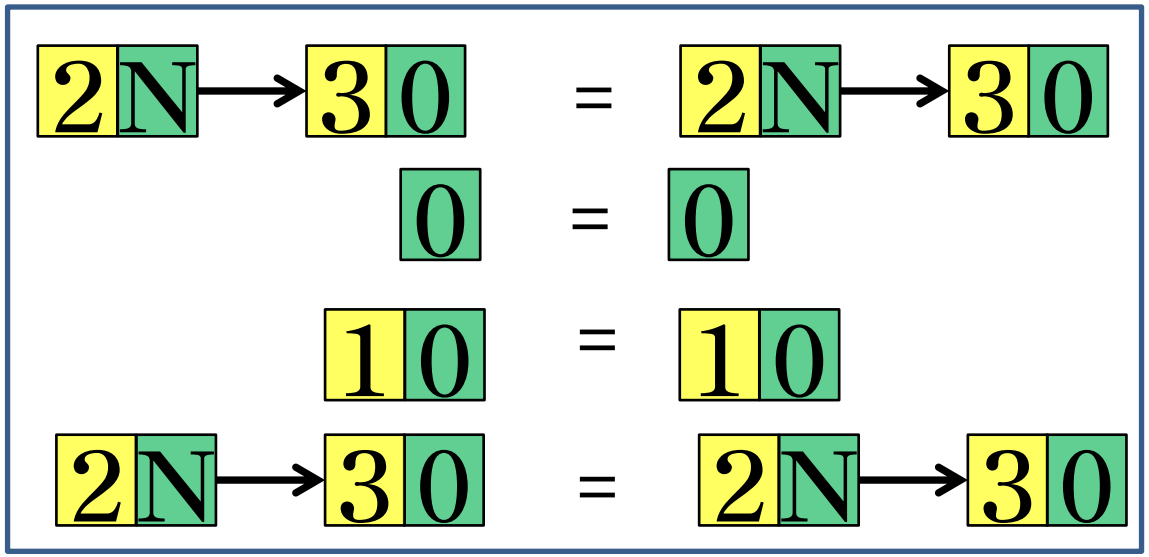
3n → 2N → 10 = 3n → 2N → 10

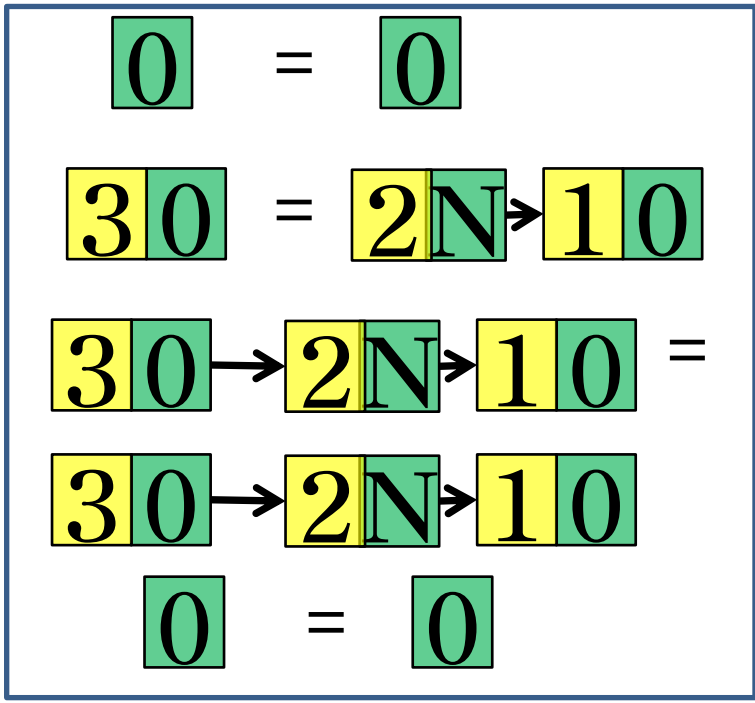
0 = 0

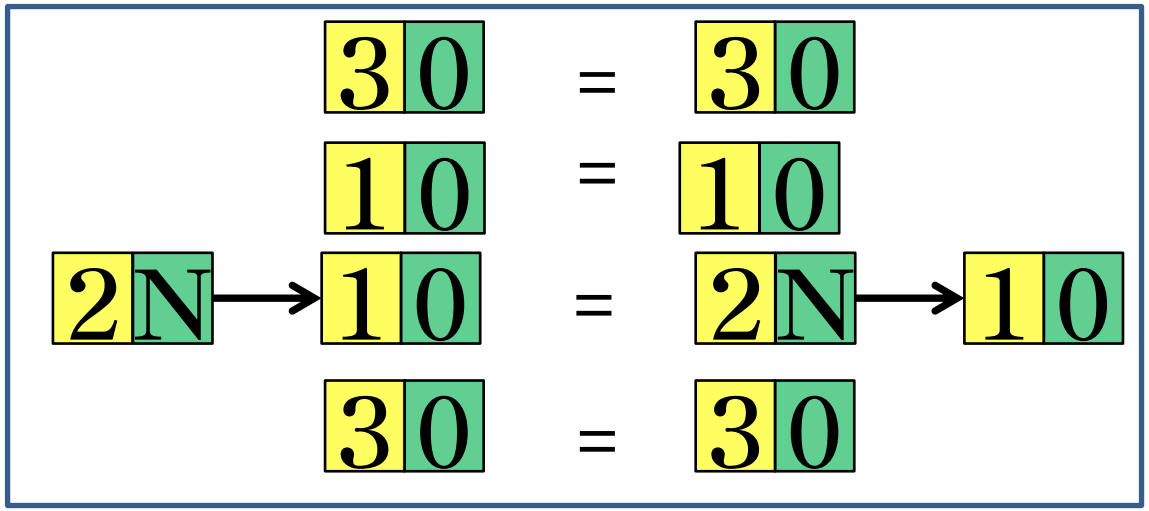
요약




```
next = cur.next;
cur.next = prev;
prev = cur;
cur = next;
```







시간복잡도/공간복잡도 계산

시간복잡도

1. 대상(Source) : 문제에서 입력받은 파라미터(array 등) (속도)
   Time Complexity : O(n)
   대상 : ListNode cur
   이유 : cur의 길이만큼 수행하기 때문입니다.

공간복잡도

2. 대상(Source) : 실제 사용되는 저장 공간을 계산(메모리 사용량)
   예) 프로그램을 실행 및 완료하는데 필요한 저장공간
   Space Complexity : O(1)
   대상 : ListNode prev = null;
   이유 : prev에 넣은값만 이용하기 때문입니다.

참고

O(1) : 스택, 큐,Map
O(n) : for문 => 데이터를 한번씩 다 호출하니까 (제일 많음)
O(log N) : sort, prirotiyQueue, binary Search Tree, Tree
O(Klog N) : k번만큼 소팅하는 경우
O(n^2) : 이중for문
O(m*n) : 이중for문인데, n이 다른경우bfs,dfs 류 (예 n=100 인데 m=5인경우)