reverse Stack and Queue 4번

크래프톤 정글 8기

5조 안채호

문제 설명

스택을 이용해서 큐의 요소를 역순으로 뒤집는 문제.

큐는 선입선출로 dequeue함수를 이용하면 먼저
입력한 순서대로 제거 되며

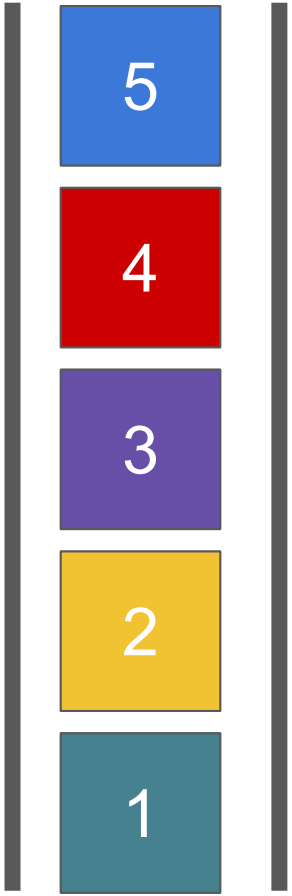
스택은 후입선출로 pop을 하면 가장 마지막에 push 된
값부터 제거 됨

큐의 모든 요소를 스택으로 이동 시키고 스택의 모든
요소를 다시 큐에 이동.

그러므로 순서는

dequeue → push → pop → enqueue

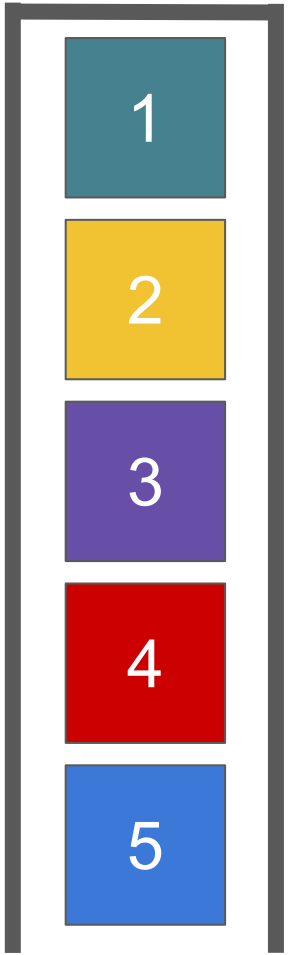
입력 예제: 1 2 3 4 5
출력 예제: 5 4 3 2 1



Queue

```
void reverse(Queue *q)
{

    Stack s;
    s.ll.head = NULL;
    s.ll.size = 0;

    while(!isEmptyQueue(q)){
        push(&s, dequeue(q));
    }

    while(!isEmptyStack(&s)){
        enqueue(q, pop(&s));
    }

}
```

Stack

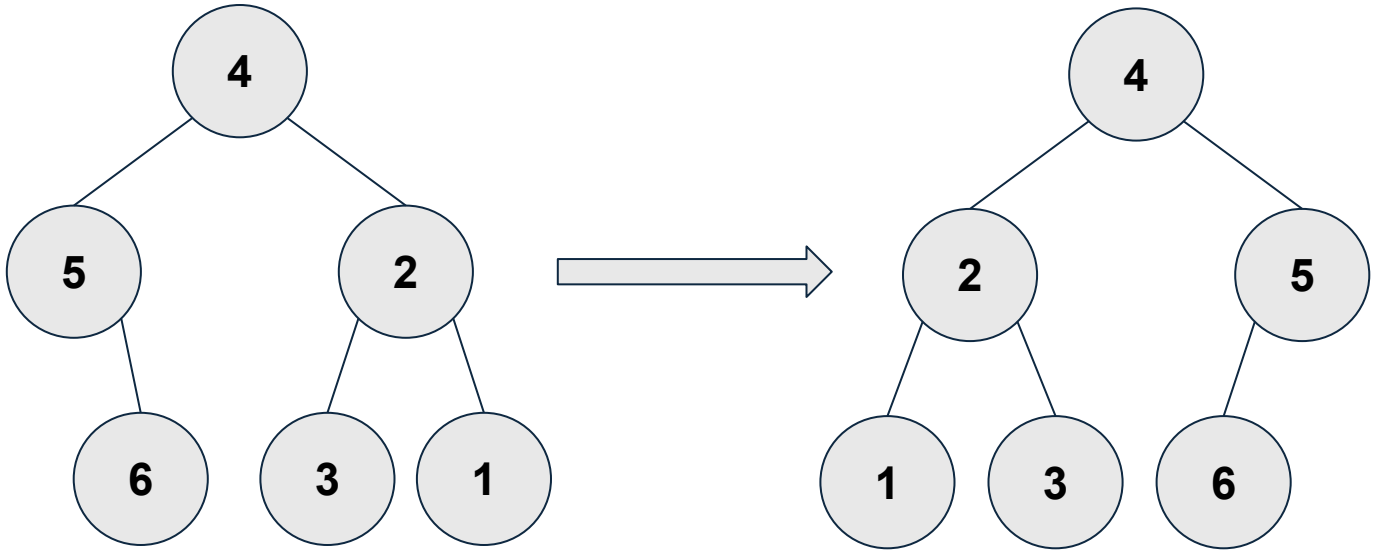


MirrorTree BinaryTree 5번

크래프톤 정글 8기

5조 김윤호

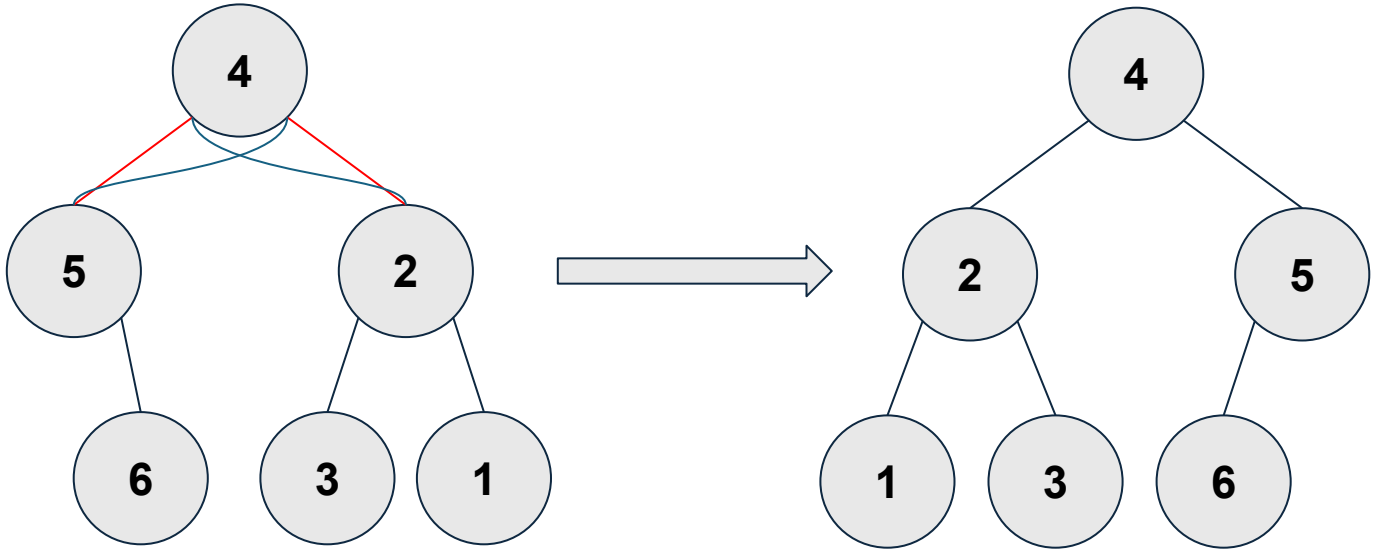
문제 설명



Question
1. 중간 매개체 사용 x
2. 새로운 트리 사용 x
3. MirrorTree 함수의 매개변수만 사용

Answer
Recursion을 사용해야겠다.

문제 설명



변경 전

변경 후

Idea

- 중간 노드를 저장할 temp가 필요

문제 접근 : Recursive

```
106  void mirrorTree(BTNode *node)
107  {
108      if(node == NULL){
109          return;
110      }
111      BTNode* RTEMP = node->right;
112      BTNode* LTEMP = node->left;
113
114      node->right = LTEMP;
115      node->left = RTEMP;
116
117      mirrorTree(node->left);
118      mirrorTree(node->right);
119
120      /* add your code here */
121  }
```

base case

input node is NULL{
    return;
}

recursive case

1. temp_left = node.left
   temp_right = node.right

1. node.right = temp_left
   node.left = temp_right

3. recur_func(node->left)
   recur_func(node->right)

recursiveRevers LinkedList 7번

크래프톤 정글 8기

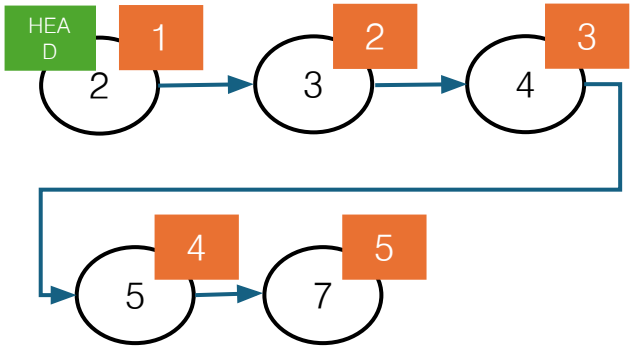
5조 김성광

문제 설명

[Fig. 1]은 5개의 리스트 노드를 보여준다.

원 안의 수는 노드의 **item** 을
사각형 은 **입력 순서** 를 나타낸다

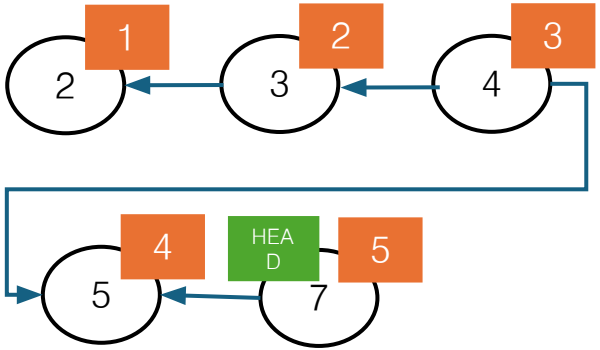
파란 선 은 해당 노드가 가리키는 **next** 노드를 의미한다



[Fig.

[Fig. 2]는 각 노드가 거꾸로 연결된 것을 보여준다.

아래의 경우 재조정된 리스트 **head** 는 7이다.



[Fig.

문제 설명

void recursiveReverse(ListNode **ptrHead)
• 리스트의 입력이 모두 끝난 상태에서 진행 • head의 주소를 바꿔야 하기 때문에 이중 포인터 사용

문제 접근: Recursive

recursiveReverse(ListNode **ptrHead)

```
if (*ptrHead == NULL || (*ptrHead)->next ==
NULL) {
        return;
    }
    // 다음 노드를 뒤집은 후, 그 리스트의 새 head
    ListNode *prev = (*ptrHead)->next;

    // 나머지를 먼저 뒤집는다
    recursiveReverse(&prev);

    // 현재 노드를 뒤집힌 나머지에 연결
    (*ptrHead)->next->next = *ptrHead;
    (*ptrHead)->next = NULL;

    // head 갱신
    *ptrHead = prev;
```