LINKED LIST

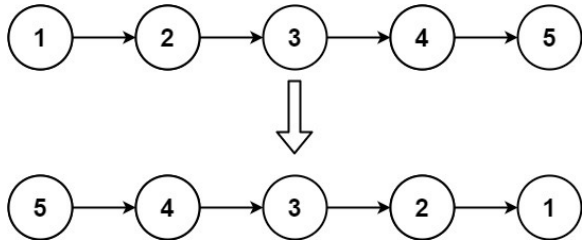
Problem – 206. Reverse Linked List

Easy

LeetCode leetcode.com/problems/reverse-linked-list

Problem

- This is a classic problem
- Given a singly linked list, reverse its order



Solution – 206. Reverse Linked List

Easy

LeetCode leetcode.com/problems/reverse-linked-list

Solution

- Use recursive approach
- Looking at the pseudo-code, this recursion will return the last node:

```
reverseList(head) {
    if (!head->next) return head
    node = reverseList(head->next);
    return node
}
```

- From end to beginning, each head will be a node in the list
- Therefore, you can change this node by setting a new head:

```
head->next->next = head;
head->next = nullptr;
```

Code – 206. Reverse Linked List

Easy

LeetCode leetcode.com/problems/reverse-linked-list

Code

Time: **O(n)** Space: **O(1)**

```
ListNode* reverseList(ListNode* head) {
    if (!head->next) return head;
    ListNode* node = reverseList(head->next);
    head->next->next = head;
    head->next = nullptr;
    return node;
}
```

Problem – 141. Linked List Cycle

Easy

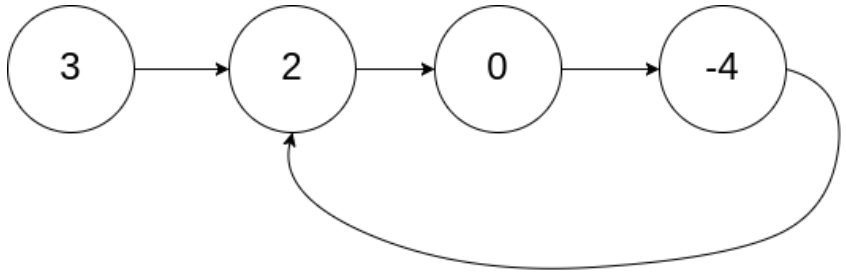
LeetCode leetcode.com/problems/linked-list-cycle

Problem

- You are given the head of a linked list
- Return **true** if there is a cycle, false otherwise
- **Example:**

  In the image below, there is a cycle (-4 to 2)

  Output: true



Solution – 141. Linked List Cycle

Easy

LeetCode leetcode.com/problems/linked-list-cycle

Solution

- Have two pointers: fast and slow
- Slow will go over each item in the linked list
- Fast will go twice as fast as slow (`fast = fast->next->next`)
- If fast reach at the end, there is no cycle
- If fast encounter slow, there is a cycle, return true

Code – 141. Linked List Cycle

Easy

LeetCode leetcode.com/problems/linked-list-cycle

Code

Time: **O(n)** Space: **O(1)**

```
bool hasCycle(ListNode *head) {
    if (!head || !head->next) return false;
    ListNode* slow = head;
    ListNode* fast = head;
    while (fast && fast->next) {
        slow = slow->next;
        fast = fast->next->next;
        if (slow == fast) return true;
    }
    return false;
}
```

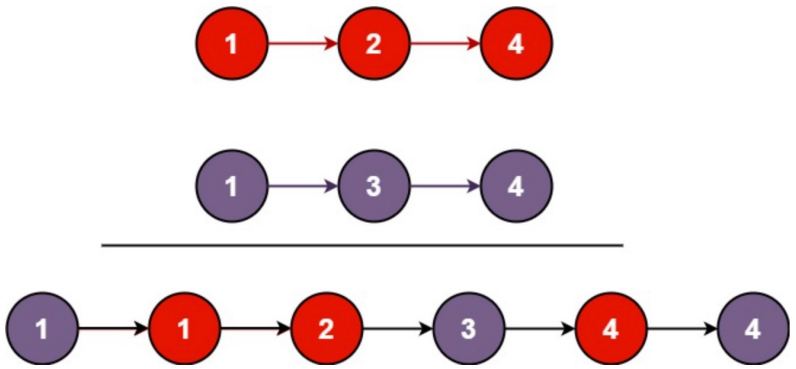
Problem – 21. Merge Two Sorted Lists

Easy

LeetCode leetcode.com/problems/merge-two-sorted-lists

Problem

- You are given the head of two linked lists (list1 and list2)
- Merge the two lists into one **sorted** list



Solution – 21. Merge Two Sorted Lists

Easy

LeetCode leetcode.com/problems/merge-two-sorted-lists

Solution

- Recursively explore the two lists. Base case:

  ```
  if (!list1) return list2;
  ```

  ```
  if (!list2) return list1;
  ```

- Compare the value of the current node of list 1 and list 2

  ```
  if (list1->val > list2->val) { ...
  ```

- Set the next node of the node with the minimum value:

  assume the previous condition is true, so

  ```
  list2->next = mergeTwoLists(list1, list2->next);
  ```

  ```
  return list2;
  ```

  meaning, we want list2->next to come before list1. But we do this recursively since we need the next result

Code – 21. Merge Two Sorted Lists

Easy

LeetCode leetcode.com/problems/merge-two-sorted-lists

Code

Time: **O(n + m)** Space: **O(n + m)** where n is the length of list1 and m is the length of list2

```
ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
    if (!list1) return list2;
    if (!list2) return list1;

    if (list1->val < list2->val) {
        list1->next = mergeTwoLists(list1->next, list2);
        return list1;
    } else {
        list2->next = mergeTwoLists(list2->next, list1);
        return list2;
    }
}
```

Problem – 23. Merge k Sorted Lists

Hard

LeetCode leetcode.com/problems/merge-k-sorted-lists

Problem

- You are given an **array of k linked lists**
- Each linked list is **sorted** in ascending order
- Merge all linked lists into one **sorted** linked-lists

Solution – 23. Merge k Sorted Lists

Hard

LeetCode leetcode.com/problems/merge-k-sorted-lists

Solution

- Create a function to merge two lists
- Go over the lists and merge with each over; **or**
- Use divide and conquer to merge (more optimal)
- Divide and conquer is more efficient because it avoids merging a big list with a small one multiple times

Code – 23. Merge k Sorted Lists

Hard

LeetCode leetcode.com/problems/merge-k-sorted-lists

Code

Time: **O(N log k)** Space: **O(log k)** where **N** is the total number of nodes across all lists and **k** is the number of lists

```
ListNode* mergeKLists(vector<ListNode*>& lists) {
    if (lists.empty()) return nullptr;
    return divideAndConquer(lists, 0 /* left */, lists.size() - 1 /* right */);
}

ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
    if (!l1) return l2;
    if (!l2) return l1;

    if (l1->val < l2->val) {
        l1->next = mergeTwoLists(l1->next, l2);
        return l1;
    } else {
        l2->next = mergeTwoLists(l2->next, l1);
        return l2;
    }
}

ListNode* divideAndConquer(vector<ListNode*> lists, int left, int right) {
    if (left == right) return lists[right];

    int mid = left + (right - left) / 2;
    ListNode* l1 = divideAndConquer(lists, left, mid);
    ListNode* l2 = divideAndConquer(lists, mid + 1, right);
    return mergeTwoLists(l1, l2);
}
```