75 Days of Code Day 29 2095. Delete the Middle Node of a Linked List
(leetcode)
Type : Linked List

You are given the head of a linked list. Delete the middle node, and return the head of the modified linked list.

The middle node of a linked list of size n is the ⌊n / 2⌋th node from the start using 0-based indexing, where ⌊x⌋ denotes the largest integer less than or equal to x.

For n = 1, 2, 3, 4, and 5, the middle nodes are 0, 1, 1, 2, and 2, respectively.

Example 1:

Input: head = [1,3,4,7,1,2,6]
Output: [1,3,4,1,2,6]
Explanation:
The above figure represents the given linked list. The indices of the nodes are written below.
Since n = 7, node 3 with value 7 is the middle node, which is marked in red.
We return the new list after removing this node.
Example 2:

Input: head = [1,2,3,4]
Output: [1,2,4]
Explanation:

The above figure represents the given linked list.
For n = 4, node 2 with value 3 is the middle node, which is marked in red.
Example 3:

Input: head = [2,1]
Output: [2]
Explanation:
The above figure represents the given linked list.
For n = 2, node 1 with value 1 is the middle node, which is marked in red.
Node 0 with value 2 is the only node remaining after removing node 1.

Solution using array

```
function deleteMiddle(head: ListNode | null): ListNode | null {
  let listElement: number[] = [];
  let node = head;
  let totalLength = 0;
  while (node) {
    totalLength++;
    listElement.push(node.val);
    node = node.next;
  }

  const middleIndex: number =
    totalLength % 2 === 0 ? totalLength / 2 : Math.floor(totalLength / 2);
  listElement.splice(middleIndex, 1);
  if (listElement.length === 0) return null;
  const newList = new ListNode(listElement[0]);
  node = newList;
  for (let index = 1; index < listElement.length; index++) {
    node.next = new ListNode(listElement[index]);
    node = node.next;
  }

  return newList;
}
```

Solution using fast and slow algorithm

1. We are using fast and slow algorithm , so we create fast , slow and prev as three pointer to traverse
2. The fast will move 2 steps , slow will move one step , when fast reach end of the list slow will reach the middle , prev will point the previous slow
3. we will just change the prev pointer next to slow's next element so the middle element will be deleted

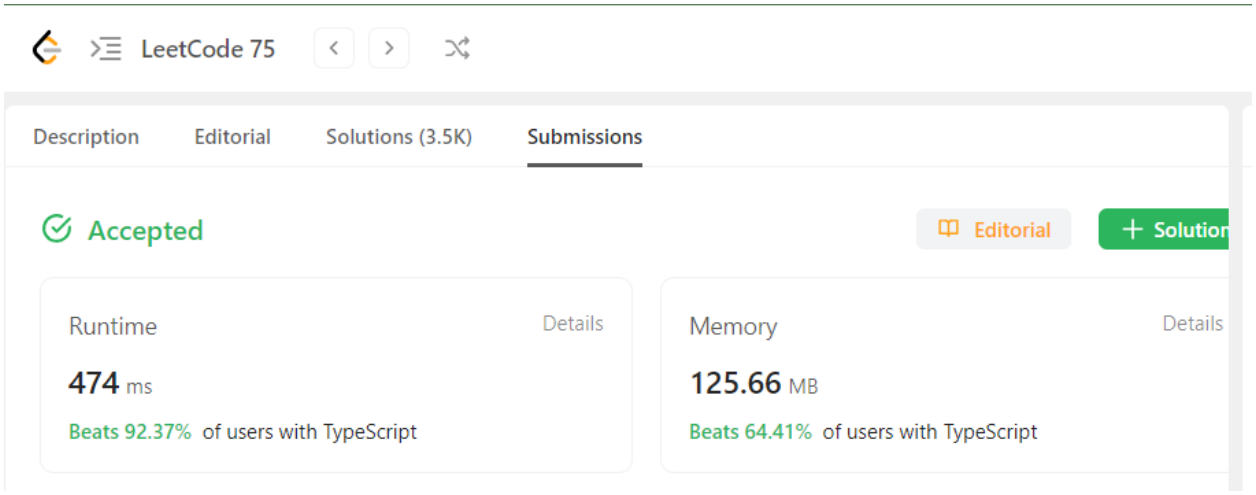
day27.ts | day28.ts | day28.js | README.md | day29.ts 2, U

day29.ts > deleteMiddleOptimize

```typescript
function deleteMiddleOptimize(head: ListNode | null): ListNode | null {
    if( !head || !head.next){
        return null;
    }
        let fast :ListNode = head;
        let slow :ListNode = head;
        let prev :ListNode | null =null;

        while(fast && fast.next){
            fast = fast.next.next;
            prev = slow;
            slow = slow.next
        }
        if(prev){
            prev.next = slow?.next ||  null;
        }
        return head;

};
```



Shubham Agrahari