75 Days of Code Day 31   Problem 2130. Maximum Twin Sum of a Linked List

Type : Linked List

In a linked list of size n, where n is even, the ith node (0-indexed) of the linked list is known as the twin of the (n-1-i)th node, if 0 <= i <= (n / 2) - 1.

For example, if n = 4, then node 0 is the twin of node 3, and node 1 is the twin of node 2. These are the only nodes with twins for n = 4.
The twin sum is defined as the sum of a node and its twin.

Given the head of a linked list with even length, return the maximum twin sum of the linked list.

Example 1:

Input: head = [5,4,2,1]
Output: 6
Explanation:
Nodes 0 and 1 are the twins of nodes 3 and 2, respectively. All have twin sum = 6.
There are no other nodes with twins in the linked list.
Thus, the maximum twin sum of the linked list is 6.
Example 2:

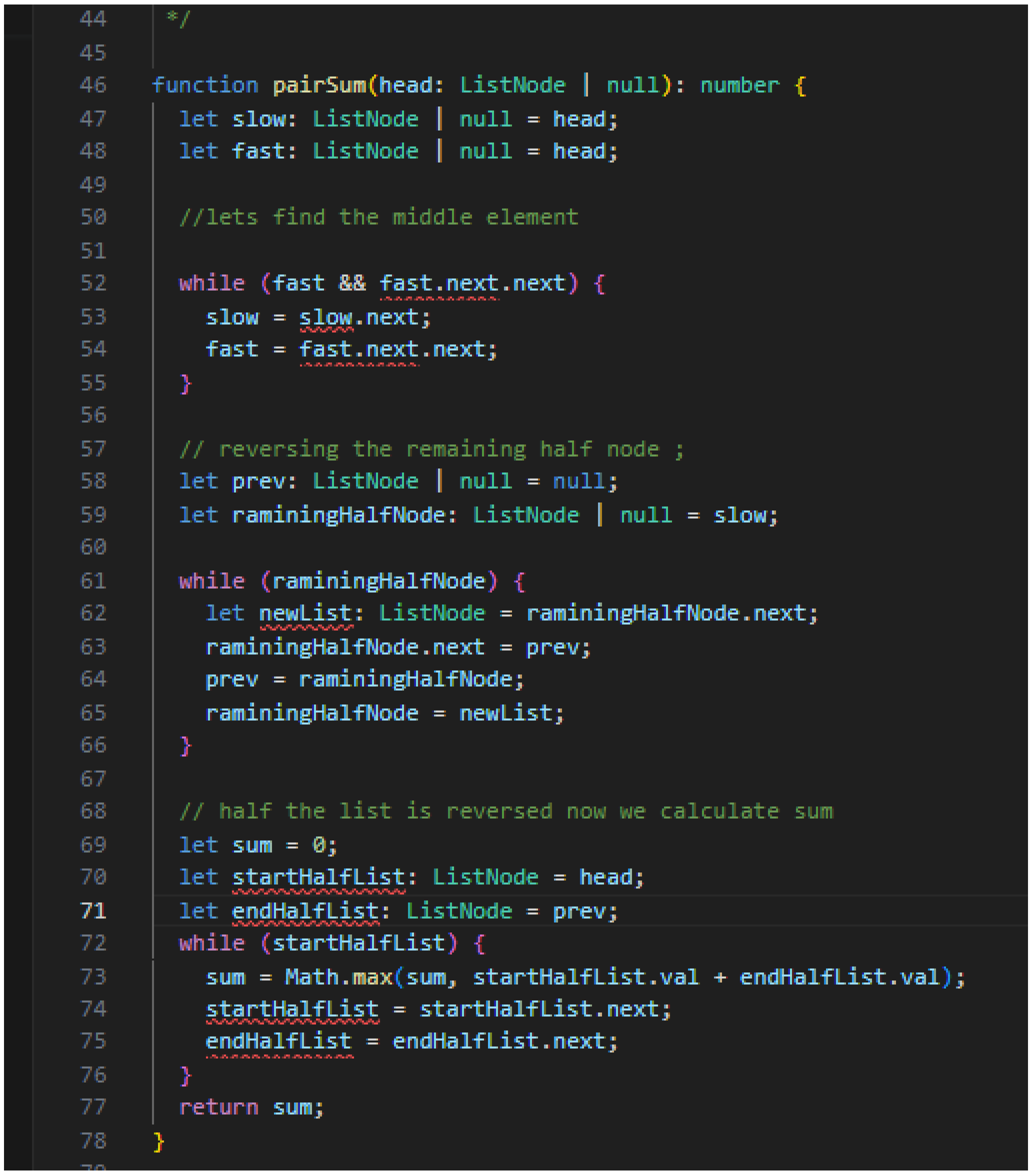
Input: head = [4,2,2,3]
Output: 7
Explanation:
The nodes with twins present in this linked list are:

- Node 0 is the twin of node 3 having a twin sum of 4 + 3 = 7.
- Node 1 is the twin of node 2 having a twin sum of 2 + 2 = 4.
Thus, the maximum twin sum of the linked list is max(7, 4) = 7.
Example 3:

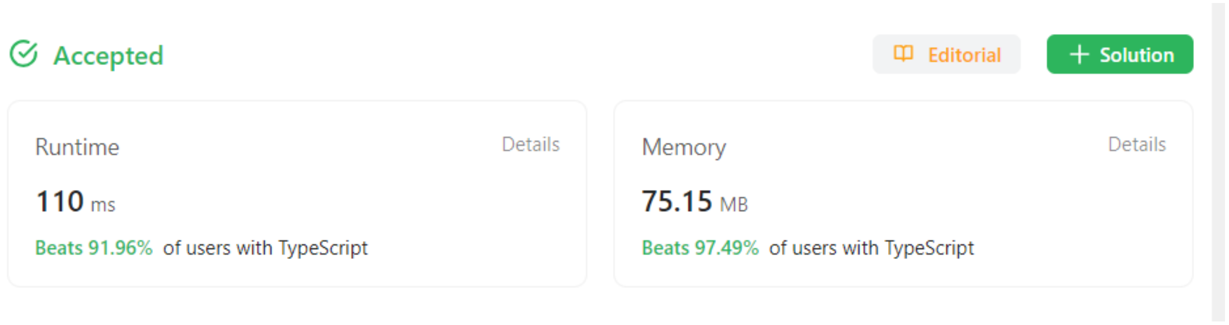
Input: head = [1,100000]
Output: 100001
Explanation:
There is only one node with a twin in the linked list having twin sum of 1 + 100000 = 100001.

Solution using linked list

1. Divide the list in two two half list
2. Reverse the second half
3. Calculate the max sum of using two pointer starting from start of both list



```
 */

function pairSum(head: ListNode | null): number {
  let slow: ListNode | null = head;
  let fast: ListNode | null = head;

  //lets find the middle element

  while (fast && fast.next.next) {
    slow = slow.next;
    fast = fast.next.next;
  }

  // reversing the remaining half node ;
  let prev: ListNode | null = null;
  let raminingHalfNode: ListNode | null = slow;

  while (raminingHalfNode) {
    let newList: ListNode = raminingHalfNode.next;
    raminingHalfNode.next = prev;
    prev = raminingHalfNode;
    raminingHalfNode = newList;
  }

  // half the list is reversed now we calculate sum
  let sum = 0;
  let startHalfList: ListNode = head;
  let endHalfList: ListNode = prev;
  while (startHalfList) {
    sum = Math.max(sum, startHalfList.val + endHalfList.val);
    startHalfList = startHalfList.next;
    endHalfList = endHalfList.next;
  }
  return sum;
}
```



Runtime Details

110 ms

Beats 91.96% of users with TypeScript

Memory Details

75.15 MB

Beats 97.49% of users with TypeScript

Shubham Agrahari

Shubham Agrahari