2807. Insert Greatest Common Divisors in Linked List

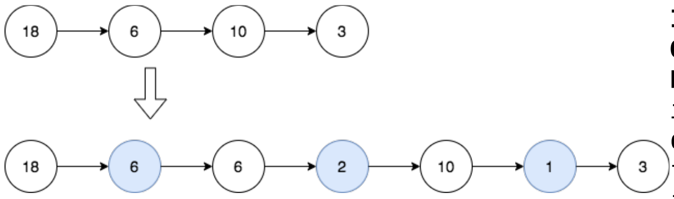
Given the head of a linked list `head`, in which each node contains an integer value.

Between every pair of adjacent nodes, insert a new node with a value equal to the **greatest common divisor** of them.

Return *the linked list after insertion*.

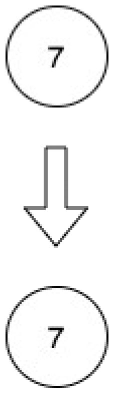
The **greatest common divisor** of two numbers is the largest positive integer that evenly divides both numbers.

Example 1:



```
Input: head = [18,6,10,3]
Output: [18,6,6,2,10,1,3]
Explanation: The 1st diagram denotes the
initial linked list and the 2nd diagram
denotes the linked list after inserting
the new nodes (nodes in blue are the
inserted nodes).
- We insert the greatest common divisor
of 18 and 6 = 6 between the 1st and the
2nd nodes.
- We insert the greatest common divisor
of 6 and 10 = 2 between the 2nd and the
3rd nodes.
- We insert the greatest common divisor
of 10 and 3 = 1 between the 3rd and the
4th nodes.
There are no more adjacent nodes, so we
return the linked list
```

Example 2:



```
Input: head = [7]
Output: [7]
Explanation: The 1st diagram denotes the
initial linked list and the 2nd diagram
denotes the linked list after inserting
the new nodes.
There are no pairs of adjacent nodes, so
we return the initial linked list.
```

Constraints:

- The number of nodes in the list is in the range `[1, 5000]`.
- `1 <= Node.val <= 1000`

2807. Insert Greatest Common Divisors in Linked List

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */

/*
    Two pointers
    Time complexity: O(n)
    Space complexity: O(1)
*/
class Solution {
    public:
        ListNode* insertGreatestCommonDivisors(ListNode* head) {
            ListNode* slow=head;
            ListNode* fast=head->next;
            while(fast){
                int gcd_val=std::gcd(slow->val,fast->val);
                ListNode* gcd_node=new ListNode(gcd_val,fast);
                slow->next=gcd_node;
                slow=fast;
                fast=fast->next;
            }
            return head;
        }
};
```