# TALLER PROBLEMAS DE BUSQUEDA Y ORDENAMIENTO

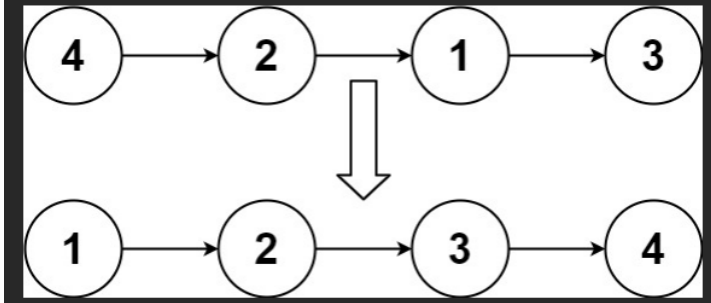
**JAIRO ALBERTO DURAN RIVERO**
**1152160**

**EJERCICIO DE LEETCODE**
**148. LISTA ORDENADA**

# 148. LISTA ORDENADA

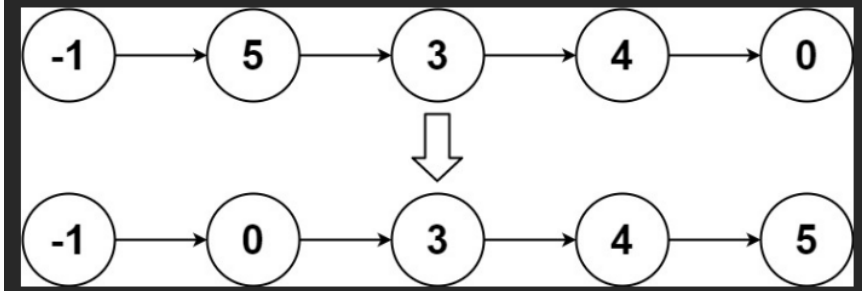
1) El ejercicio numero 148 de LeetCode sortList nos decía. Dada la cabeza de una lista enlazada, devolver la lista después de ordenarla en **orden ascendente.**

**Ejemplo 1:**




**Entrada:** cabeza = [4,2,1,3]
**Salida:** [1,2,3,4]

**Ejemplo 2:**



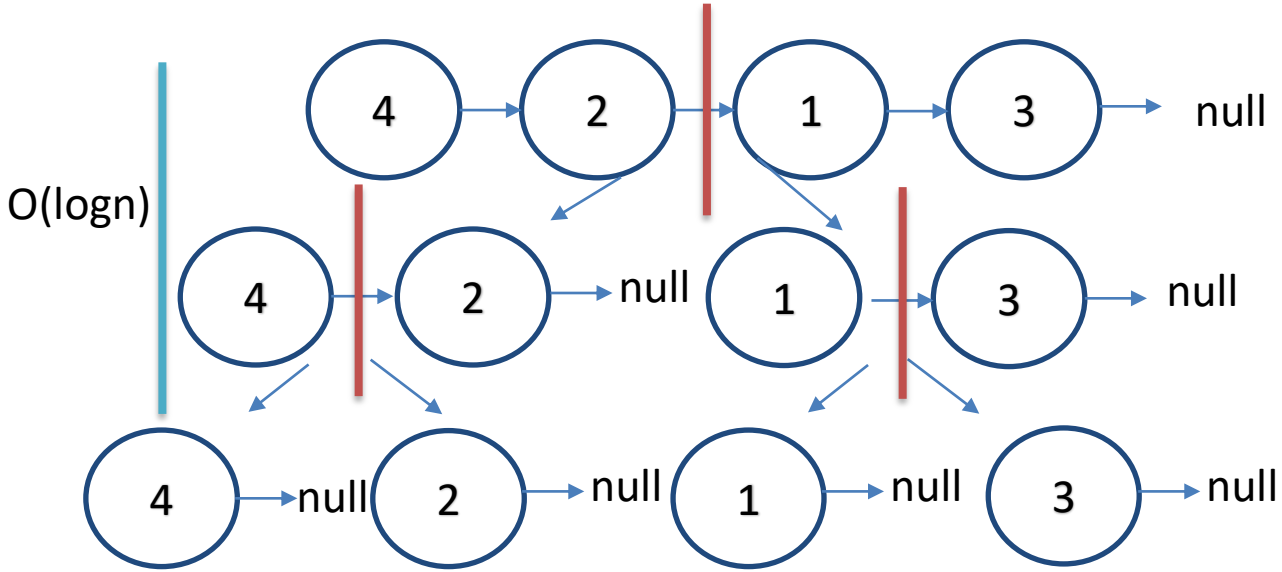

```
Entrada: cabeza = [-1,5,3,4,0]
 Salida: [-1,0,3,4,5]
```

**Ejemplo 3:**

```
Entrada: cabeza = []
 Salida: []
```

Seguimiento: ¿Puedes ordenar la lista enlazada en O(n logn) tiempo y O(1) memoria (es decir, espacio contante)?

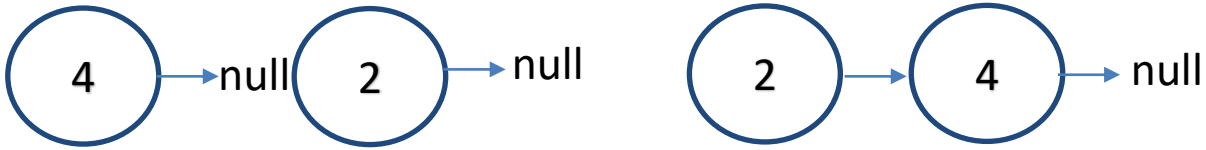
Se decidió usar el método de ordenamiento Merge sort para el ordenamiento de la lista.



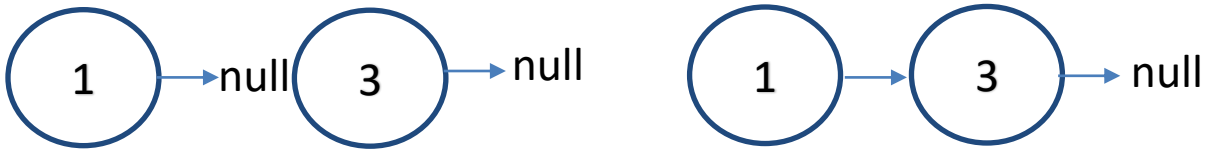

Cortar y dividir en dos listas.

Se repite recursivamente cortar y dividir las listas tanto izquierda como derecha.

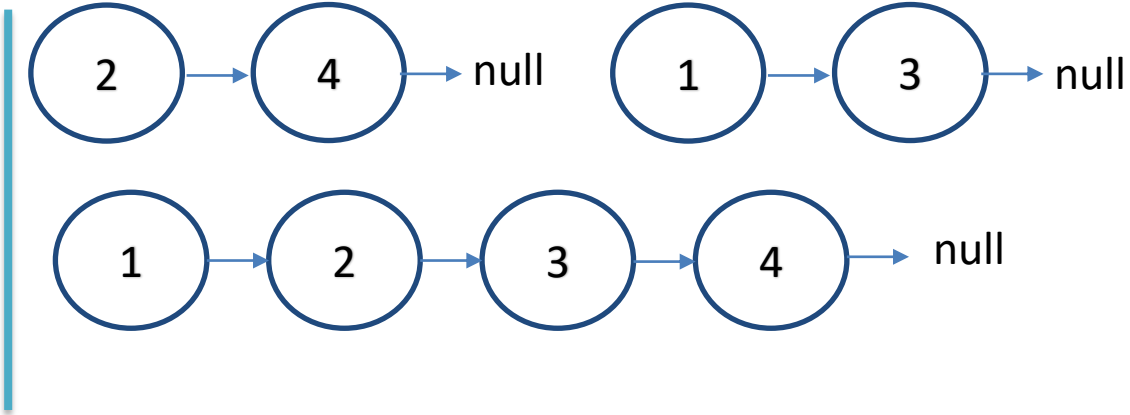
En este momento se llega al caso base de nuestra recursión , Ahora haremos la parte de la fusión tomaremos esos nodos y los fusionaremos asegurándonos de que estén en el orden correcto.




Se unieron anteriormente el dos apuntaba a null ahora su siguiente es el cuatro y el cuatro seguirá apuntado a null. Hacemos exactamente lo mismo con los otros nodos.



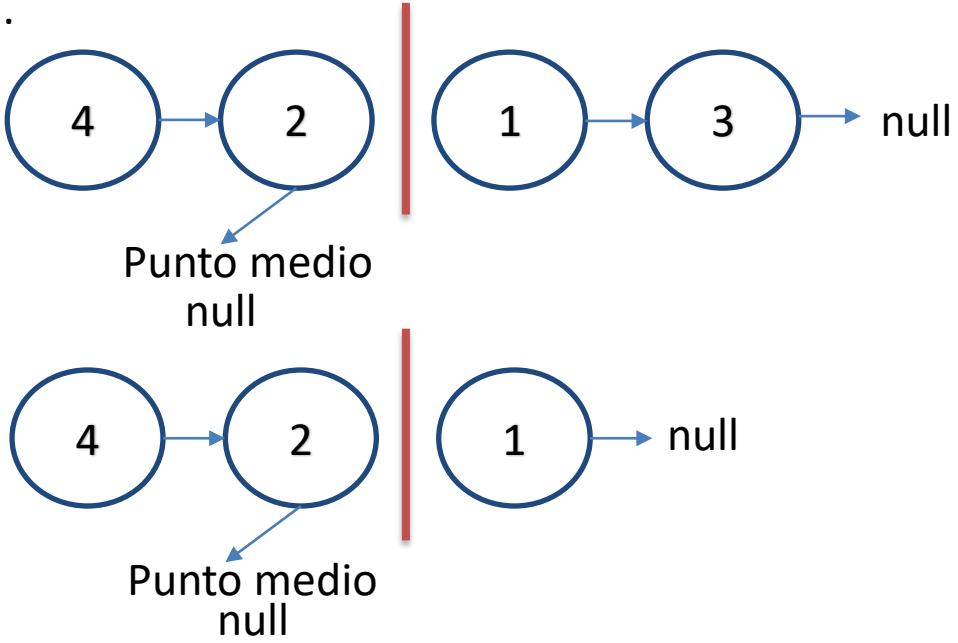
Ahora tomaremos estas dos listas de tamaño 2 y luego las fusionaremos, pero esto obviamente será un poco diferente porque cada lista tiene mas de un nodo, así que lo primero que veremos será el valor de cada lista porque sabemos que las listas están en orden.




O(n)

O(n logn)
complejidad

Algo muy importante y que debemos resaltar es dividir la mitad de manera correcta.

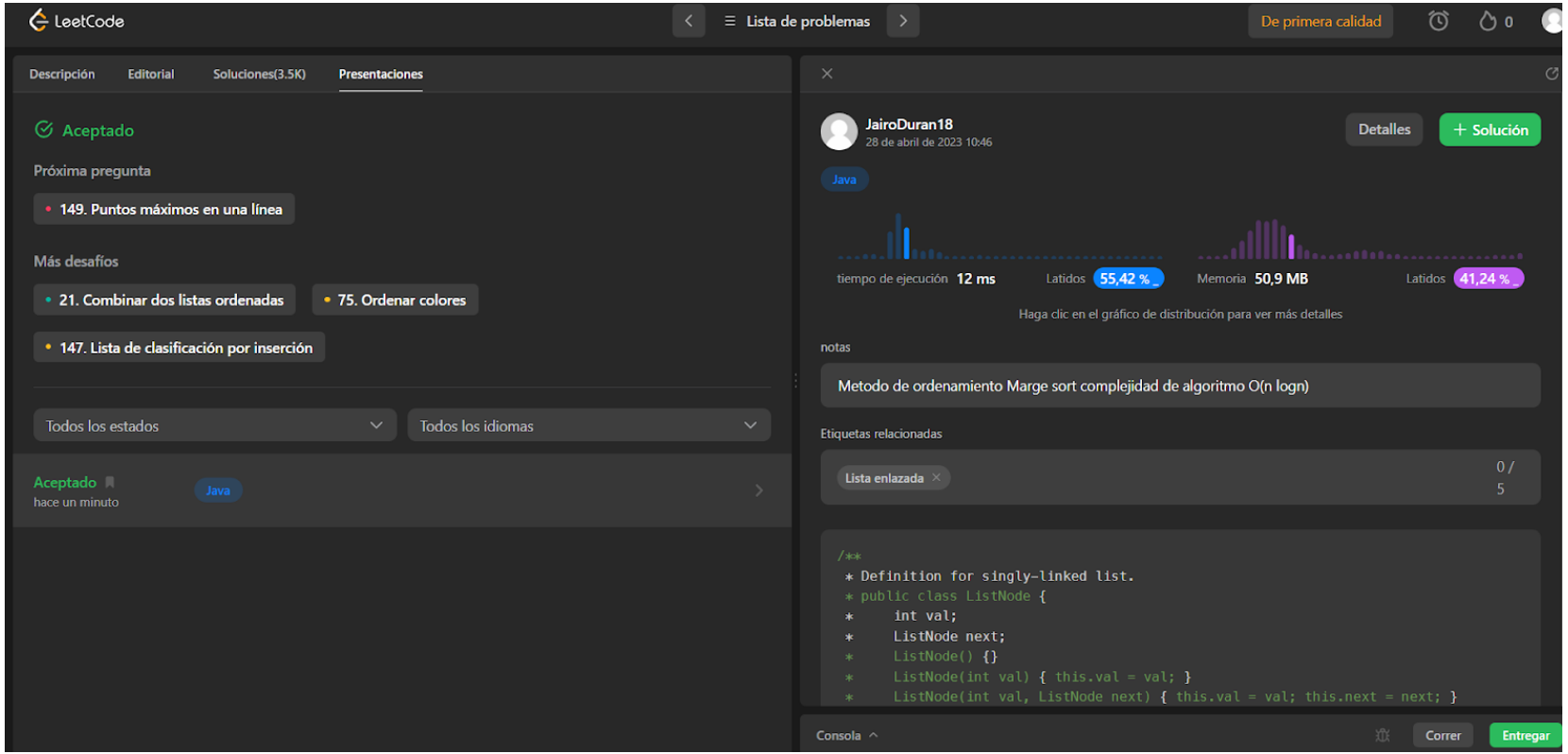


# Código del problema

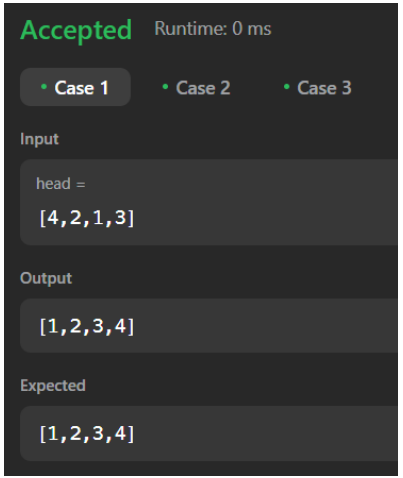
```java
class Solution {
    public ListNode sortList(ListNode head) {
        if (head == null || head.next == null) {
            return head;
        }

        ListNode left = head;
        ListNode right = getMid(head);
        ListNode tmp = right.next;
        right.next = null;
        right = tmp;

        left = sortList(left);
        right = sortList(right);
        return merge(left, right);
    }

    private ListNode getMid(ListNode head) {
        ListNode slow = head, fast = head.next;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        return slow;
    }
```

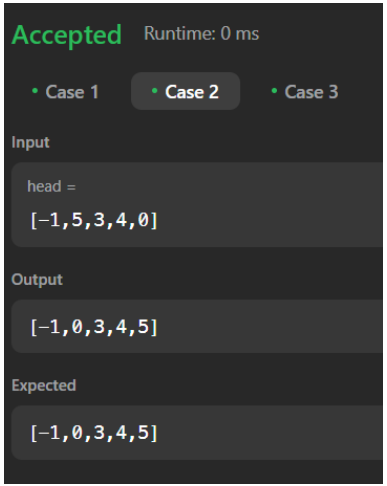
```java
    private ListNode merge(ListNode list1, ListNode list2) {
        ListNode tail = new ListNode();
        ListNode dummy = tail;

        while (list1 != null && list2 != null) {
            if (list1.val < list2.val) {
                tail.next = list1;
                list1 = list1.next;
            } else {
                tail.next = list2;
                list2 = list2.next;
            }
            tail = tail.next;
        }

        if (list1 != null) {
            tail.next = list1;
        }
        if (list2 != null) {
            tail.next = list2;
        }

        return dummy.next;
    }
}
```

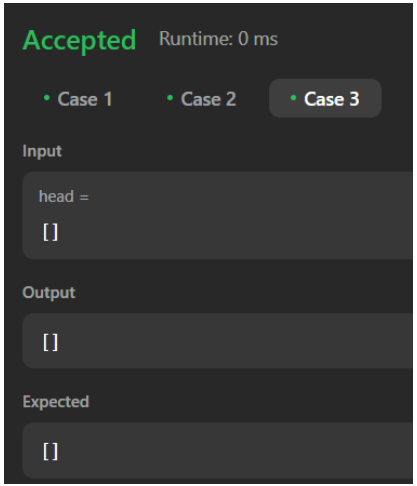
# Evidencia


LeetCode
Lista de problemas
De primera calidad
0
Descripción
Editorial
Soluciones(3.5K)
Presentaciones
Aceptado
Próxima pregunta
149. Puntos máximos en una línea
Más desafíos
21. Combinar dos listas ordenadas
75. Ordenar colores
147. Lista de clasificación por inserción
Todos los estados
Todos los idiomas
Aceptado
hace un minuto
Java
JairoDuran18
28 de abril de 2023 10:46
Detalles
+ Solución
Java
tiempo de ejecución 12 ms
Latidos 55,42 %
Memoria 50,9 MB
Latidos 41,24 %
Haga clic en el gráfico de distribución para ver más detalles
notas
Metodo de ordenamiento Marge sort complejidad de algoritmo O(n logn)
Etiquetas relacionadas
Lista enlazada
0 / 5
/**
* Definition for singly-linked list.
* public class ListNode {
* int val;
* ListNode next;
* ListNode() {}
* ListNode(int val) { this.val = val; }
* ListNode(int val, ListNode next) { this.val = val; this.next = next; }
Consola
Correr
Entregar

2) Lo siguiente que se hizo fue desarrollar una aplicación completa, con main y lectura de datos, de manera que se pruebe la solución independientemente de la plataforma https://leetcode.com/. Esta solución debe ser consistente con la solución realizada en el numeral 1.



Resultados generados por la aplicación

```
run:
Ingrese la cantidad de casos:
3
4 2 1 3
-1 5 3 4 0

1 2 3 4
-1 0 3 4 5
```

Se comprueba que los resultados obtenidos son los mismos

3) Por ultimo se desarrollo una aplicación que genera al menos 100 casos de prueba para el problema. Los casos deben cubrir todas las posibilidades de casos del problema, de manera equilibrada, este programa se desarrollo en java.
Casos evaluados :

- Caso 1 = Lista Vacía
- Caso 2 = Lista Ordenada
- Caso 3 = Lista de un solo elemento
- Caso 4 = Lista Totalmente desordenada
- Caso 5 = Lista mitad izquierda ordenada, mitad derecha desordenada
- Caso 6 = Lista mitad Izquierda desordenada, mitad derecha ordenada

Links del GitHub:
https://github.com/JairoD18/TallerBusquedaOrdenamiento

Video YouTube:
https://www.youtube.com/watch?v=awAWI7XS3E0


# GRACIAS

ACREDITADA EN ALTA CALIDAD
UFPS
6 años
Res. MEN 015176 del 2 de Agosto de 2022

Universidad Francisco de Paula Santander
Vigilada Mineducación

60 AÑOS 1962-2022