SOLUCIÓN EVALUACIÓN CONTINUA 3

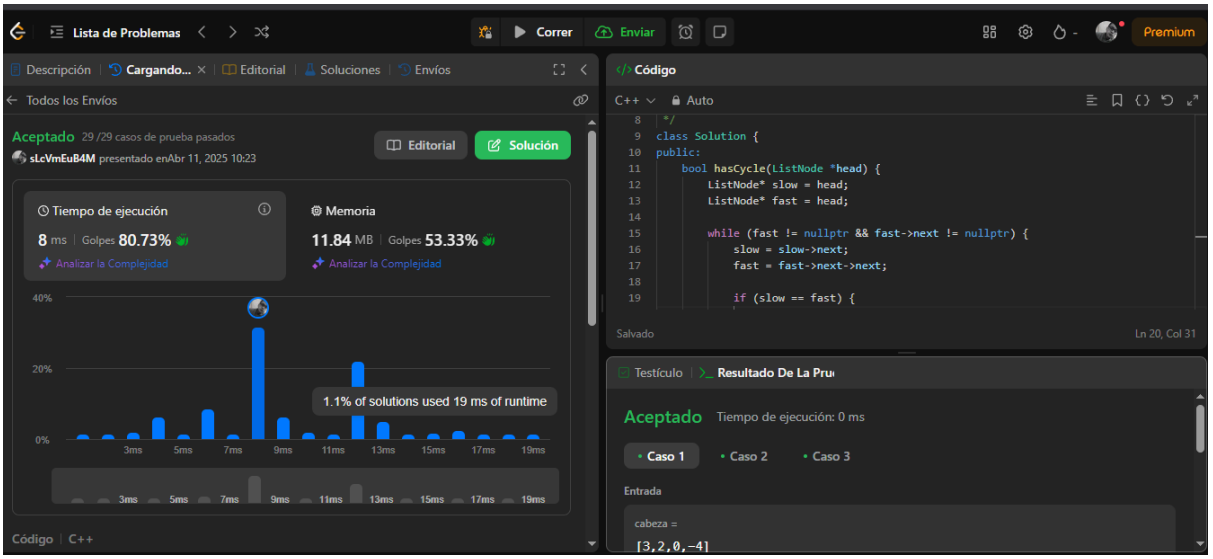
Autores: Yulinio Zavala Mariño , Gabriel Vaccaro, Jose Espinoza Verano, Favian Huarca Mendoza, Leonel Leodolfo Campuzano Diestra , Sergio Ricce

Linked List Cycle

Solución alto nivel:

Se crean dos punteros:
"slow" que avanza de uno en uno. "fast" que avanza de dos en dos.
Si hay un ciclo, los punteros se encontrarán. Si no hay ciclo, fast llegará a nullptr.
Condiciones de parada:
Mientras "fast" y "fast->next" no sean nullptr, seguimos avanzando.
Si "slow" == "fast" en algún momento, retornamos true.
Si llegamos al final de la lista sin que se crucen, retornamos false.

Captura de LeedCode:

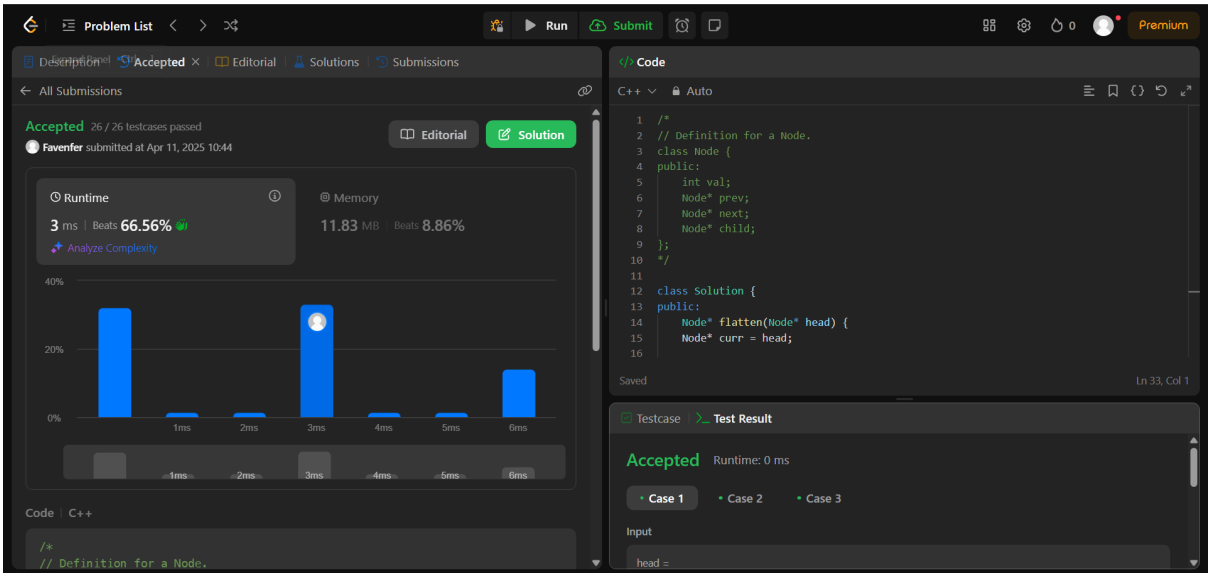


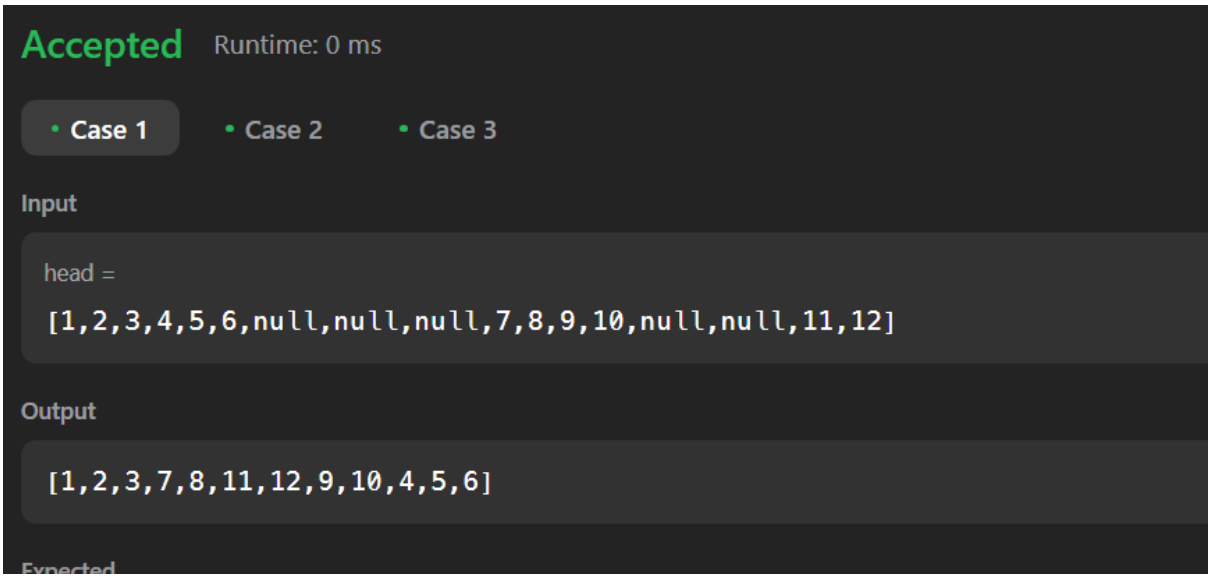
Flatten a Multilevel Doubly Linked List

Solución alto nivel:

Se usa el puntero "curr" para recorrer la double linked list, mientras esta no tenga valor de nullptr. Cada que recorre un nodo se pregunta si este nodo tiene un "child". Si no lo tiene solo pasa al siguiente, pero si lo tiene procede a usar la variable contenedor "child" y "next", las cual toman los valores del "child" y "next" de "curr" respectivamente. Luego se procede a concatenar los nodos "curr" y "child" asignando el "child->prev" en "curr" y el "curr->next" en "child". Posteriormente el valor de "child" de "curr" se apunta hacia nullptr, ya que el "child" ya no esta encapsulado en "curr". Una vez hecho esto, se procede a validar si "curr->next" no apunta a nullptr, si lo hace se acaba el recorrido. Si no, "curr" toma el valor de "curr->next" y sigue recorriendo.

Captura LeedCode:





Design Linked List

Solución alto nivel:

Para resolver el problema, se usó una lista doblemente enlazada circular con un nodo centinela. Esta estructura simplifica mucho las operaciones al evitar casos especiales al principio o al final de la lista. El nodo centinela no guarda un valor real, pero actúa como punto de partida y de referencia para recorrer la lista. Por esto, insertar y eliminar nodos se vuelve más ordenado y sin necesidad de controlar punteros nulos.

Inicializamos el objeto con nuestro centinela apuntando hacia sí mismo tanto en head como en tail con un valor arbitrario y un tamaño 0. Posteriormente tenemos funciones de encontrar index, agregar un valor en un índice, agregar los valores en el head, la cola y borrar el valor en un índice.

Para la función de encontrar el index, nos guiamos de que no sea un índice invalido, o sea menor a 0 y mayor o igual al tamaño de la lista. Luego inicializamos un puntero curr que apuntará al siguiente nodo de centinela. Luego hacemos un bucle para recorrer según el índice dado y mover el puntero curr.

Para la función agregar en un índice, de forma similar al anterior, debemos encontrar el índice que deseamos, solo que en este caso queremos el previo, no el curr. Por eso, inicializamos el puntero prevNode. Una vez tenemos el puntero en el lugar correspondiente, agregamos un nuevo nodo con el valor de val y luego asignamos los punteros del nodo siguiente y prev a ese nuevo nodo e incrementamos el valor de size ya que aumentamos un nuevo nodo.

Para la función borrar, similar a buscar por el índice, solo que en esta función, se encarga de borrar el nodo del valor que deseamos. Ajustando los punteros next y prev para desvincular al nodo que queremos borrar y vincular a los otros nodos siguientes.

Captura LeedCode: