Taller IA: Ordenamiento

Problema 1

Trixie Toys es una empresa que fabrica software especializado para organizar juguetes en estanterías virtuales. Su última aplicación, *ToyShelf*, permite a los usuarios colocar y reorganizar juguetes en estantes, sobreponiendo unos sobre otros. Describe una base de datos que mantenga un registro de los juguetes en una estantería y que soporte las siguientes operaciones:

1. **create_shelf()**: construye una estantería vacía que no contiene juguetes.
2. **add_toy(x)**: añade un juguete con un ID único entero x en la parte superior de la estantería.
3. **view_shelf()**: devuelve un arreglo de los IDs de los juguetes en la estantería en orden de abajo hacia arriba.
4. **move_below(x, y)**: mueve el juguete con ID x directamente debajo del juguete con ID y.

La operación (1) debe ejecutarse en $O(1)$ en el peor de los casos, la operación (2) debe ejecutarse en $O(n)$ y (3) en $O(n\log(n))$deben ejecutarse cada una en $O(n)$ en el peor de los casos, y la operación (4) debe ejecutarse en $O(n)$ en el peor de los casos, donde n es el número de juguetes en la estantería en el momento de la operación.

Problema 2

En una fila de torres de Lego construidas por niños, cada torre está compuesta por un cierto número de ladrillos de Lego apilados uno sobre otro. Las torres están alineadas de izquierda a derecha a lo largo de una mesa larga y recta. Un día, un niño travieso decide derribar las torres. Si el niño empuja una torre que contiene b ladrillos, esa torre caerá, y en su caída derribará cualquier torre adyacente a su derecha que contenga estrictamente menos de b ladrillos, provocando un efecto dominó.

Para cada torre, queremos determinar su impacto, es decir, el número de torres que caerían si el niño empuja esa torre.

(a) Supón que el número de ladrillos en cada torre de izquierda a derecha es dado por un arreglo de enteros. Calcula cuántas torres se derrumbarán, incluyendo la que se empuja, si el derrumbe comienza en una torre específica.

(b) Una torre es "especial" si (1) no tiene una torre adyacente a su derecha, o (2) su vecino inmediato a la derecha tiene al menos tantos ladrillos como ella. Dado un arreglo que contiene el número de ladrillos en cada torre, describe un algoritmo de tiempo $O(n)$ que devuelva el impacto para cada torre cuando todas las torres menos una son especiales.

(c) Dado un arreglo que contiene el número de ladrillos en cada torre, describe un algoritmo de tiempo $O(n \log n)$ que devuelva el impacto para cada torre.

(d) Escribe una función en C++ `get_impacts` que implemente tu algoritmo para calcular el impacto de cada torre.

Evaluación

La evaluación de este taller se realizará en base la siguiente escala:

1. **Muy superior (4.7-5.0):** El programa compila y se ejecuta correctamente, implementando todas las operaciones con precisión y cumpliendo con los tiempos de complejidad indicados. Además, se han respondido correctamente todas las preguntas teóricas con explicaciones claras y detalladas. El código está bien estructurado, es eficiente y está completamente documentado.

2. **Superior (4.2-4.6):** El programa compila y se ejecuta correctamente, pero presenta ligeras desviaciones en la eficiencia de algunas operaciones o hay áreas menores de mejora en la estructura del código. Las respuestas a las preguntas teóricas son correctas, aunque podrían ser más detalladas o claras. La documentación del código es adecuada.

3. **Satisfactorio (3.6-4.1):** El programa compila y se ejecuta, pero con algunos errores menores que afectan parcialmente el comportamiento de las operaciones o no cumple del todo con los tiempos de complejidad solicitados. Las respuestas a las preguntas teóricas son en su mayoría correctas, pero algunas presentan ambigüedades o falta de claridad. La organización del código y la documentación podrían mejorarse.

4. **Suficiente (3.0-3.5):** El programa compila, pero tiene problemas significativos en la ejecución de las operaciones o en la complejidad del tiempo. Las respuestas a las preguntas teóricas son incompletas o incorrectas en varias partes. El código puede ser difícil de seguir, estar incompleto o carecer de una adecuada documentación.

5. **Insuficiente (0.0-2.9):** El programa no compila o no puede ser ejecutado debido a errores importantes en la implementación de las operaciones. Las respuestas a las preguntas teóricas son incorrectas o están ausentes, y el código no está documentado ni organizado de manera comprensible.

Escala de Evaluación

La escala de evaluación se construyó de acuerdo con el Artículo 79 del Decreto Rectoral No. 1751 del 31 de octubre de 2022.