Tercer examen integrador - Programación II
Lunes Noche - 1C 2025

Profesor Monzón, Nicolás Alberto

23 de junio de 2025

Requisitos para aprobar el integrador

A. Por actos de deshonestidad académica será sancionado.

B. Para poder aprobar este examen Ud. deberá estar administrativamente en condiciones de poder rendir el examen. En caso de no estarlo y rendir el examen, este no va a ser corregido y quedará anulado.

C. Deberá comprimir la carpeta `src` de su proyecto junto a los `txt` utilizados (también se admiten `pdf` y `MD`). El archivo comprimido deberá tener como nombre el número de grupo. Deberá enviarlo por mail a `nimonzon@uade.edu.ar`.

D. Deberá desarrollar un set de prueba para demostrar que funciona su código (no se piden test unitarios, pero si alguna prueba en el `main` del proyecto).

E. Solo podrá utilizar técnicas vistas en este curso. Está prohibido el uso de librerías, estructuras de datos abstractas que vienen por defecto en la JRE y el uso de genéricos. No respetar este punto es suficiente para desaprobar el integrador.

F. Deberá ser entregado dentro de las 3 horas y 30 minutos. a partir del horario de inicio.

G. CONDICIONES PARA APROBAR: el examen se mide con una escala logarítmica donde obtener al menos 60 puntos corresponderá a un 4 y se deberá cumplir con todos los puntos anteriores. En caso de no cumplir con alguno de los puntos anteriores, el examen queda desaprobado sin excepción.

Marco teórico

Definimos un árbol binario indexado como un árbol binario donde cada nodo, además de su valor, tiene un índice. Este índice indica en qué orden debe recorrerse el árbol.

Ejercicio 1 (10 %)

Modificar el TDA `BinaryTree` visto en la cursada, para que sea posible asignar un índice a un nodo.

Tip: No es necesario agregar a la definición un método que dado un índice devuelva el valor del nodo en ese índice. Esto se pide más adelante en este integrador pero en una clase utilitaria.

Ejercicio 2 (20 %)

Crear un `BinaryTreeHelper` que permita chequear que los índices no se repiten entre los nodos de un árbol.

Ejercicio 3 (10 %)

Crear un método en una clase utilitaria, que realice un mapeo desde un árbol binario a un árbol binario indexado. Los índices deberán colocarse usando *preorder*.

Ejercicio 4 (20 %)

Crear un método en una clase utilitaria que, a partir de un árbol binario indexado, y un índice, devuelva el elemento en ese índice. Optimizar el resultado sabiendo que los índices se colocaron usando *inorder*.

Ejercicio 5 (20 %)

Crear un método en una clase utilitaria para verificar si el árbol está ordenado. *La definición de árbol ordenado es la más natural: está ordenado si, al recorrer el árbol en base a sus índices, entonces los valores obtenidos también están ordenados.*

Ejercicio 6 (20 %)

Crear un método que permita recorrer un árbol por niveles de forma recursiva. Utilice una solución basada en el ejercicio 3 (no se permite utilizar la solución usual basada en árboles binarios).