Estructura de Datos y Algoritmos
Grado en Ingeniería Informática
Doble Grado Ingeniería Informática y Administración de Empresas
Universidad Carlos III de Madrid

Curso 2021-2022

uc3m | Universidad Carlos III de Madrid

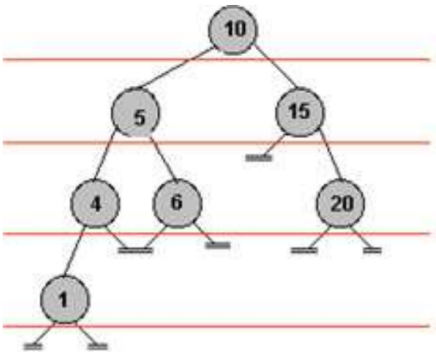
CC BY NC SA

Caso Práctico

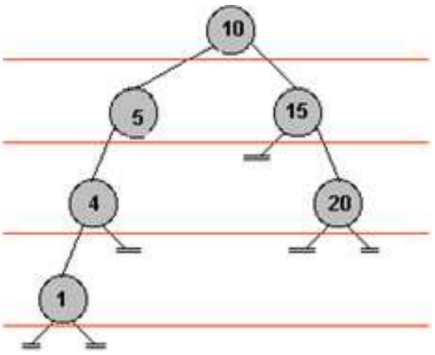
El caso práctico se compone de tres fases. Este documento describe el enunciado de la segunda fase. Los enunciados de la fase 3 se publicarán en las próximas semanas.

Fase 2 - Estructuras de datos no lineales ADT (árboles AVL)

Un árbol AVL es un árbol de búsqueda binario, en el que las alturas de los dos subárboles hijos de cualquier nodo difieren como máximo en uno. Cualquier árbol de búsqueda binaria T que satisfaga la propiedad de equilibrio de altura se dice que es un árbol AVL, llamado así por las iniciales de sus inventores: Adel'son-Vel'skii y Landis.



Un árbol de búsqueda equilibrado AVL



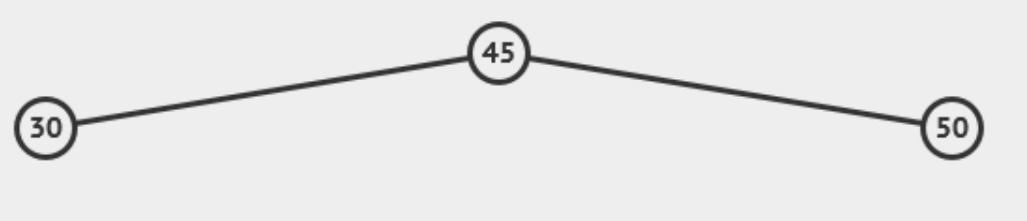
no es AVL porque la altura izquierda y la altura derecha del nodo 5 difieren en 2.

Los árboles AVL deben mantenerse equilibrados. Por tanto, después de una inserción de un nuevo nodo, el algoritmo de insertar deberá comprobar si en la rama donde se ha insertado el nuevo nodo, algún nodo ha quedado desequilibrado, y tendrá que equilibrarlo. De igual forma, después de borrar un nodo, el algoritmo de borrar deberá comprobar si en la rama donde se ha eliminado el nodo, algún nodo ha quedado desequilibrado, y tendrá que equilibrarlo.

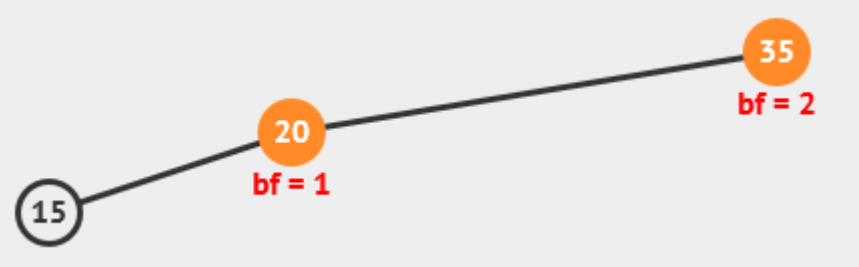
Hay cuatro casos que hay que considerar para equilibrar un nodo desequilibrado. Estos casos se muestran a continuación:

- **Rotación Simple Izquierda**: en la siguiente imagen, el nodo con elemento 30 está desequilibrado. Rotar un nodo nos va a permitir pasarlo de su rama más larga a su rama más corta. En este caso, la rama del nodo con elemento 30 es su rama derecha, por tanto, deberemos rotar el nodo a la rama izquierda. Su actual hijo derecho, nodo con elemento 45, se deberá convertir en la nueva raíz del árbol, y el nodo con elemento 30 pasará a ser su hijo izquierdo. El nodo con elemento 50, sigue siendo hijo derecho del nodo con elemento 45.

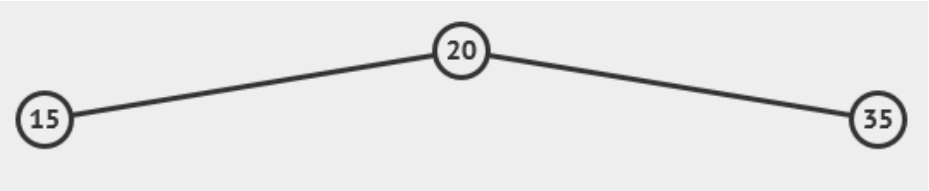
La rotación simple izquierda producirá el siguiente árbol, donde todos los nodos ya están equilibrados



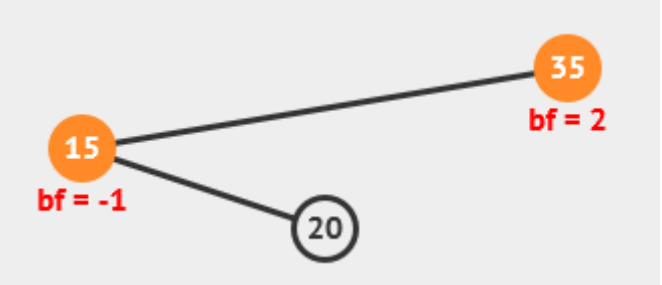
- **Rotación Simple Derecha**: como muestra la siguiente imagen, el nodo con elemento 35 está desequilibrado. La rama más larga que cuelga del nodo es su rama izquierda.



Si rotamos el nodo con elemento 35 a su rama derecha, estaremos consiguiendo reducir la longitud de dicha rama. Así, el nodo con elemento 35 girará a la derecha, convirtiéndose en el nuevo hijo derecho del nodo con elemento 20. Este nodo se convierte en la nueva raíz del árbol. El nodo con elemento 15 sigue siendo hijo izquierdo del nodo con elemento 20. La rotación simple derecha producirá el siguiente árbol, donde todos los nodos ya están equilibrados

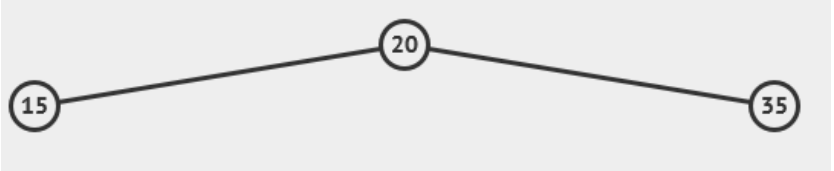


- **Rotación doble izquierda-derecha**. Estas rotaciones se aplican cuando la rama más larga que cuelga del nodo desequilibrado hace una especie de zigzag, como se muestra en la imagen siguiente. El nodo con elemento 35 está desbalanceado, y su rama más larga primero va a la izquierda, y luego a la derecha. En este tipo de casos, se deben realizar dos rotaciones:

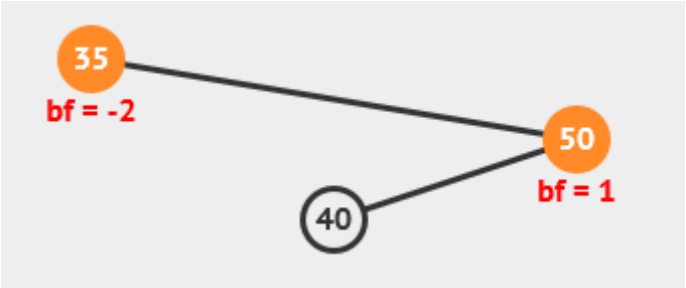


La primera rotación a aplicar será una rotación izquierda, para que el nodo con elemento 20, rote a la izquierda y se convierta en el nuevo hijo izquierdo del nodo desequilibrado (nodo con elemento 35), consiguiendo como resultado el siguiente árbol. El nodo con elemento 35 sigue estando desequilibrado.

La segunda rotación a aplicar, ya la conocemos, es la rotación simple derecha, porque el nodo con el elemento 35 pasará a ser el subárbol derecho de la nueva raíz, el nodo con elemento 20. En el árbol resultante, ya están todos los nodos equilibrados.

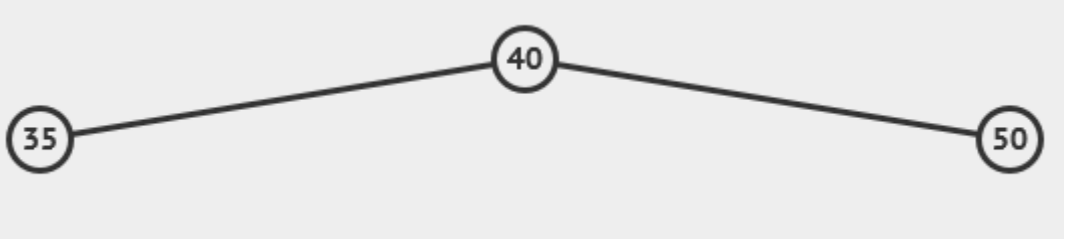


- **Rotación doble derecha-izquierda**. Como muestra la siguiente imagen, el nodo con elemento 35 está desequilibrado, y su rama más larga hace un zig-zag, primero se va a la derecha y luego a la izquierda. Aquí también deberemos aplicar una rotación doble.



Primero aplicaremos una rotación a la derecha, para que el nodo con el elemento 40, rote a su derecha, convirtiéndose en el hijo derecho del nodo con elemento 35.

El nodo con elemento 35 sigue estando desequilibrado, pero ahora ya podemos aplicarle una rotación simple izquierda, rotando dicho nodo para que se convierta en el nuevo subárbol izquierdo del nodo con elemento 40, que a su vez se convertirá en la nueva raíz del árbol. En el árbol resultante, todos los nodos ya están equilibrados



En aula global, podéis encontrar más información sobre este tipo de árboles y sus rotaciones.

En la fase 2, tenéis que implementar un árbol AVL. Como se dijo anteriormente, un árbol AVL es un árbol binario de búsqueda con la propiedad que sus operaciones de inserción y borrado aseguran que el árbol resultante esté equilibrado.
Ya os proporcionamos un esqueleto de código para que empecéis a trabajar en la fase 2. Este esqueleto contiene varios ficheros:

- Los ficheros bintree.py y bst.py contienen la implementación del árbol binario y árbol binario de búsqueda, respectivamente. **Estos ficheros no pueden ser modificados bajo ningún concepto.**

- El fichero phase2_test.py que contiene las pruebas unitarias necesarias para comprobar que tu solución es correcta y robusta. Este fichero no puede modificarse bajo ningún concepto.
- El fichero phase2.py, donde deberás implementar tu solución. Este fichero contiene la clase AVLTree que es una subclase de BinarySearchTree, y que por tanto, hereda todos sus atributos y funciones. Sin embargo, será necesario sobreescribir las funciones insert y remove para asegurar que tras la inserción o borrado de un nodo, se realicen las operaciones necesarias para que el árbol resultante esté equilibrado.

Además de implementar las funciones insert y remove en la clase AVL, se recomienda la implementación de otras funciones que os ayuden a calcular el factor de equilibrio de un nodo, detectar si un nodo está desequilibrado y qué rotaciones deberían aplicarse, así como la implementación de las rotaciones específicas.

Normas:

1. Una solución se considera correcta si es robusta (no tiene errores para ninguna entrada), es correcta (resuelve el problema) y es eficiente (pueden existir varias soluciones, debes tratar de buscar siempre la solución más eficiente)
2. En la implementación del caso práctico, no está permitido utilizar estructuras de Python como Listas, Queues o Diccionarios, etc.
3. El caso práctico debe ser realizado por un grupo formado por dos miembros (ambos deben pertenecer al mismo grupo reducido). En ningún caso se permitirá grupos con más de dos miembros. Tampoco se permiten grupos individuales ya que una de las competencias a evaluar será el trabajo en equipo. Si no tienes compañero, por favor, envía un correo a tu profesor de prácticas.
4. Modo de entrega: En el grupo reducido de aula global, se publicará una tarea titulada '**Entrega Fase 2 y 3**'. La fecha de Entrega para esta primera fase será el **16 de mayo, 9.00 am**.
5. Formato de entrega: un zip cuyo nombre está formado por los dos NIAS de los estudiantes que forman el grupo, separados por un guión. Por ejemplo, *10001234-10005678.zip.* Únicamente uno de los dos miembros, será el encargado de subir la solución del grupo a la tarea. El zip deberá contener todos los ficheros del esqueleto (con el fichero avl.py modificado con tu solución).
6. **Defensa**: **16 mayo de 2022**. Se citará al grupo a una determinada hora. La defensa del caso práctico es un examen oral. La asistencia es obligatoria. Si algún alumno no asiste, su calificación en el caso práctico será NP. Durante esta defensa, el profesor planteará a cada miembro del equipo una serie de preguntas que deberá responder de forma individual. Cada miembro del equipo debe ser capaz de discutir cualquiera de las decisiones tomadas en el diseño e implementación de cualquiera de las funcionalidades descritas en el caso práctico. La nota final del caso práctico estará condicionada por la nota obtenida en la defensa. Si un alumno no es capaz de

discutir y defender ciertas decisiones, no será calificado con respecto a estas, aunque hayan sido correctamente implementadas.

7. Se recomienda seguir las recomendaciones descritas en Zen of Python (https://www.python.org/dev/peps/pep-0020/) y la guía de estilo (https://www.python.org/dev/peps/pep-0008/) publicada en la página oficial de Python.