CS 284 B Quiz 5: GPA database

1 Problem Description

In this quiz, we will see how data structures such as heap and self-balancing trees can be used to solve real world problems. Your task is to use the `MaxHeap` and `AVLTree` we learned in class to implement a class `GPADatabase` that manages the GPAs of students. `GPADatabase` has 4 operations:

- `public GPADatabase(int size)`: initialize the heap and avltree in the GPA database, size is the size of the heap;
- `public void insertStudent(String student_name, Double gpa)`: insert the student with student_name and gpa in the database;
- `public ArrayList<String> removeTopkStudent(int k)`: Remove the top k students from this database (from both the avl tree and the heap). Return the names of the top k students, sorted by the descending order of their GPAs. If k is larger than the database size, return all students names in the database sorted by the descending order of their GPAs, do not throw an exception. If the student database is empty, return an ArrayList of size 0;
- `public Double searchStudentGPA(String student_name)`: search the GPA of student whose name is student_name, return a Double variable which is the GPA of the student with student_name. if the student_name does not exist in the database, return null, do not throw an exception.

To help you implement `GPADatabase`, we have provided two generic classes in your template: `MaxHeap<S, T>` and `AVLTree<S, T>`. Use the APIs in the two classes to implement the 4 operations above.

Each node in `MaxHeap<S, T>` and `AVLTree<S, T>` are defined as `class HeapNode` and `class AVLNode`. `AVLNode` extends `HeapNode`. `HeapNode` contains two class variables: `value1` and `value2`. `S` is the type of `value1` and `T` is the type of `value2`. Use one of `value1` and `value2` to represent the student's name, the other to represent the student's GPA.

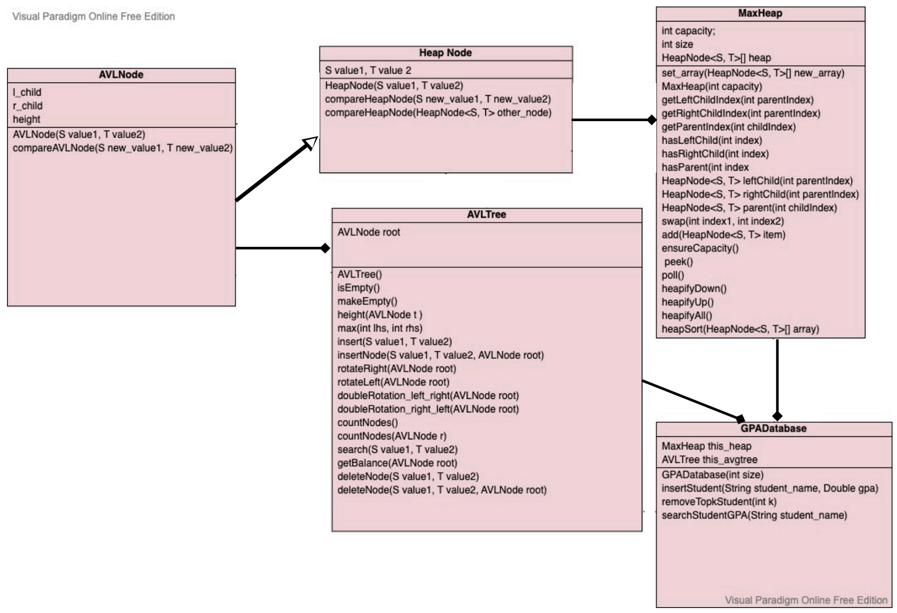
Notice `MaxHeap<S, T>` and `AVLTree<S, T>` are not finished. You will need to implement the compare functions for the two classes:

- `HeapNode.compareHeapNode(S new_value1, T new_value2)`: this function compares the current heap node with the new heap node with `value1` and `value2`;
- `AVLNode.compareAVLNode(S new_value1, T new_value2)`: this function compares the current heap node with the new avl node with `value1` and `value2`;

In total, you will implement 6 functions. You can quickly navigate them by ctrl+F "TODO".

2 UML Diagram

This is the UML Diagram to help you understand the relation between the classes.



Visual Paradigm Online Free Edition
AVLNode
l_child
r_child
height
AVLNode(S value1, T value2)
compareAVLNode(S new_value1, T new_value2)
Heap Node
S value1, T value 2
HeapNode(S value1, T value2)
compareHeapNode(S new_value1, T new_value2)
compareHeapNode(HeapNode<S, T> other_node)
MaxHeap
int capacity;
int size
HeapNode<S, T>[] heap
set_array(HeapNode<S, T>[] new_array)
MaxHeap(int capacity)
getLeftChildIndex(int parentIndex)
getRightChildIndex(int parentIndex)
getParentIndex(int childIndex)
hasLeftChild(int index)
hasRightChild(int index)
hasParent(int index
HeapNode<S, T> leftChild(int parentIndex)
HeapNode<S, T> rightChild(int parentIndex)
HeapNode<S, T> parent(int childIndex)
swap(int index1, int index2)
add(HeapNode<S, T> item)
ensureCapacity()
peek()
poll()
heapifyDown()
heapifyUp()
heapifyAll()
heapSort(HeapNode<S, T>[] array)
AVLTree
AVLNode root
AVLTree()
isEmpty()
makeEmpty()
height(AVLNode t)
max(int lhs, int rhs)
insert(S value1, T value2)
insertNode(S value1, T value2, AVLNode root)
rotateRight(AVLNode root)
rotateLeft(AVLNode root)
doubleRotation_left_right(AVLNode root)
doubleRotation_right_left(AVLNode root)
countNodes()
countNodes(AVLNode r)
search(S value1, T value2)
getBalance(AVLNode root)
deleteNode(S value1, T value2)
deleteNode(S value1, T value2, AVLNode root)
GPADatabase
MaxHeap this_heap
AVLTree this_avgtree
GPADatabase(int size)
insertStudent(String student_name, Double gpa)
removeTopkStudent(int k)
searchStudentGPA(String student_name)
Visual Paradigm Online Free Edition

Figure 1: UML diagram

3 Test cases

- `public void insertStudent(String student_name, Double gpa):`

After inserting the following 4 students: ("C", 0.0), ("B", 1.0), ("D", 2.0), ("A", 3.0), the MaxHeap of your GPADatabase should look like:

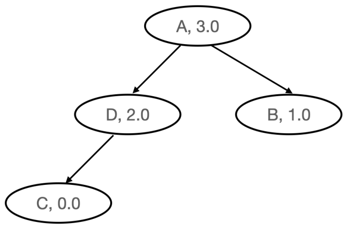


Figure 2: The MaxHeap after inserting ("C", 0.0), ("B", 1.0), ("D", 2.0), ("A", 3.0)

the AVLTree of your GPADatabase should look like:

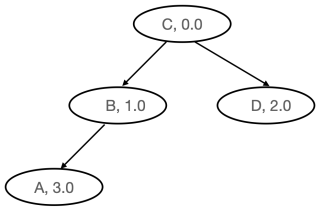


Figure 3: The AVLTree after inserting ("C", 0.0), ("B", 1.0), ("D", 2.0), ("A", 3.0)

- `public ArrayList<String> removeTopkStudent(int k)`:

  After inserting the following 4 students: ("C", 0.0), ("B", 1.0), ("D", 2.0), ("A", 3.0), removeTopkStudent(2) should return the ArrayList⟨String⟩ which is ["A", "D"]. The remaining MaxHeap should look like:

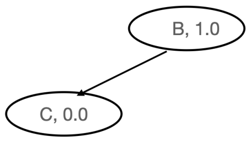


Figure 4: The MaxHeap after inserting ("C", 0.0), ("B", 1.0), ("D", 2.0), ("A", 3.0) then removing the top-2 students

the remaining AVLTree should look like:

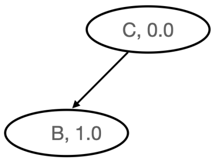


Figure 5: The AVLTree after inserting (”C”, 0.0), (”B”, 1.0), (”D”, 2.0), (”A”, 3.0) then removing the top-2 students

- `public Double searchStudentGPA(String student_name)`:
  After inserting the following 4 students: (”C”, 0.0), (”B”, 1.0), (”D”, 2.0), (”A”, 3.0), searchStudentGPA(”B”) should return 1.0, while searchStudentGPA(”E”) should return null;