Yurii Huba
SFU ID: 301540732
Instructor: David Mitchell
Course: CMPT225

Assignment 3

To organize my priority queue for this assignment I create Avl Tree (AT) “assignments” that sorts all tasks by their ID’s and Binary Heap (BH) “precedences” that sorts tasks by priorities. Implementations of BH and AT classes are in separate .h files with corresponding names.

BH stores array of nodes (heapNode) and each of those includes priority of task, task identificator (ID’s) and pointer to the corresponding AT node (AT node with the same task ID). As I mentioned, BH nodes are sorted by priorities. HeapNodes are defined as a struct separate from BH class. HeapNode struct is placed in BinaryHeap.h file.

Each of AT nodes contains a task ID and integer variable that stores index of the corresponding node (node with the same task ID) in BH array. Nodes in AT are sorted by task ID’s. AvlNodes are defined (similarly to HeapNodes) as a struct separate from AvlTree class. AvlNode struct is placed in AvlTree.h file.

In the subsequent paragraphs I will describe how each function of PQ works.

PQ zero constructor uses zero constructor of Binary Heap (which sets initial capacity of heapNode array) and zero constructor of AvlTree (which sets root of Avl Tree to nullptr);

PQ constructor with arguments of two vectors (tasks and priorities). Firstly, I call constructor of Binary Heap (precedences) that creates array of heapNodes with given tasks and priorities but sets all pointers to AvlTree nodes to nullptr. Then I use loop to check which task ID is in the particular position of the Binary Heap array (function checkTaskforIndex returns ID of task for given index of Binary Heap array) and insert them in Avl Tree (passing task ID and it’s found position in Binary Heap array). After that, I again scan array in Binary Heap, find each task in Avl Tree using function findTask (this function returns a pointer to AvlNode that has task ID as element) and use function setTaskPointer that updates pointer of heapNodes.

PQ isEmpty() function uses another isEmpty function from Binary Heap, that returns bool value whether size of array of heapNodes is 0.

PQ deleteMin() function uses checkTaskforIndex function from Binary Heap to find out which task uses array position 1 (has smallest priority). Then it removes this task by it’s ID from Avl Tree. After that this function removes this task from Binary Heap array using deleteMin function for precedences (this deleteMin deletes an array element with index 1, returns reference to it and repairs Binary Heap). Then I scan throw Binary Heap array checking task ID at each index with function checkTaskforIndex and update array indexes for each node in Avl Tree using function setBinaryHeapPosition.

PQ findMin() uses precedences’ findMin() to return a Binary Heap array element with index 1 (smallest priority).

PQ insert() firstly inserts an element to Avl Tree with index 0, after that uses findTask() function to copy the pointer to the node in Avl Tree it was inserted. Then, it inserts the task into Binary Heap array using copied pointer to Avl Tree node, repairs Binary Heap and updates all indexes in Avl Tree.

PQ updatePriority() checks if task is in the Avl Tree: if it is, function uses getBinaryHeapPosition() to find out task's index in the array, changes task's priority in the array, repairs Binary Heap and updates all indexes in Avl Tree. If task is not in Avl Tree, function just inserts it.

PQ size() uses precedences' size() function to return size of the Binary Heap array.