DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Pratical Class 8

Priority Queues - Heaps

Estruturas de Informação

An efficient way of implementing priority queues is using the **binary heap** data structure that allows for insertions and removals in logarithmic time.

A heap is a binary tree that satisfies two additional properties:

- for every position p other than the root, the key stored at p is greater than or equal to the key stored at p's parent.
- the heap must be complete; as a consequence, the height of a heap containing *n* entries will be proportional to *log(n)* and therefore heap update operations can be made so as to take logarithmic time.

Inserting an entry

To maintain the complete binary tree property, the new node should be placed at a position just beyond the rightmost node at the bottom level of the tree, or as the leftmost position of a new level, if the bottom level is already full (or if the heap is empty). Furthermore, in order to ensure that the heap-order property is maintained, we will have to compare the new node with its parent, swapping the contents if its key is greater than its parent's. This process goes on upwards until no violation of the heap-order is detected. This upward movement of the newly inserted entry by means of swaps is usually called ***percolate up***.

Removing an entry

The entry to be removed is the one stored at the root of the heap and therefore the one with the smallest key but it cannot be simply removed because it would create two disconnected trees. We ensure that the shape of the heap respects the complete binary tree property by deleting the leaf at the tree's last position, defined as the rightmost position at the bottommost level of the tree, and copying it to the root (in place of the entry with minimal key that is being removed by the operation). But because most likely this new root node will now violate the heap-order property, we will have to compare it with its children and perform a swap if necessary. This process goes on with successive swaps until the heap order is restored in what is usually called ***percolate down***.

Exercise 1

Add the two methods ***percolateUp()*** and ***percolateDown()*** to the class *HeapPriorityQueue*, to perform the bubbling up and down of an entry, by means of successive swaps, in order to restore the heap-order property. These methods will be used by the methods **insert()** and **removeMin().**

Exercise 2

Add the method ***buildHeap()*** to the class *HeapPriorityQueue*. This method should perform a batch bottom-up construction of the heap.

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Pratical Class 8

Priority Queues - Heaps

Estruturas de Informação

Complementary Exercise

1. Implement the AirTrafficCC class with the purpose of simulating an Air-traffic Control Centre main data structure. Each plane has a string identifier and an associated numerical priority. Implement methods that:

a) add a new plane to the landing queue
b) clear a plane for landing, removing it from the queue
c) return the number of planes currently waiting for landing
d) return the estimated time to clear a certain plane for landing, based on a 5 minute time slot per plane
e) change the priority of a given plane