DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Laboratory Class 8

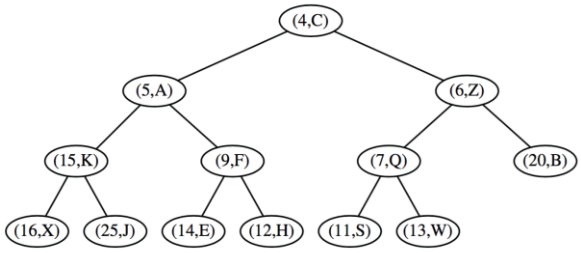
Priority Queues - Heaps

Estruturas de Informação

Introduction

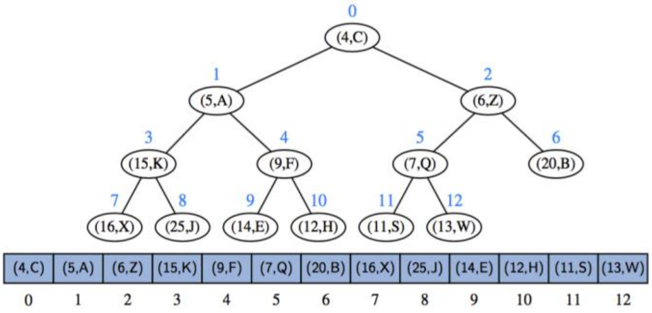
A priority queue (PQ) is a queue-like structure that delivers its contents according to the priority of the elements stored in it. An entry in such a structure is made of an element and its associated priority or key. The element with the *minimal* key will be the next to be removed from the queue. An 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.



An efficient way of representing a complete binary tree is to store its elements in an array-based list such that the element at position *p* is stored at an index equal to the level number *f*(*p*) of *p*:

- If p is the root, then f(p)=0.
- If p is the left child of position q, then f(p) = 2f(q)+1.
- If p is the right child of position q, then f(p) = 2f(q)+2.



Exercises

Start by downloading and extracting the project `PL8_PriorityQueue_initial` from Moodle to your Projects folder. The contents of the project are:

- An AbstractPriorityQueue class that implements the PriorityQueue interface and contains an inner class PQEntry implementing the Entry interface
- A DefaultComparator class to be used by the AbstractPriorityQueue class
- A PriorityQueue and Entry interface files
- A HeapPriorityQueue class that extends the AbstractPriorityQueue class

The public interface for the class HeapPriorityQueue is the following:

`Entry<K,V> insert(k, v)` - Creates an entry with key k and value v in the PQ.
`Entry<K,V> min()` - Returns without removing a PQ entry<k,v>
`Entry<K,V> removeMin()` - Removes and returns the entry<k,v> with minimal key from the PQ or null if it is empty.

int size() - Returns the PQ's number of entries.
boolean isEmpty() - Returns a boolean indicating whether the PQ is empty.

Exercise 1

a) Implement the methods ***insert()***, ***min()*** and ***removeMin()*** of the HeapPriorityQueue interface. Refer to the algorithms described in the Lecture notes and the Practical worksheet.

b) Test your implementation running the tests in the HeapPriorityQueueTest class.

Exercise 2

Add the methods ***toString()*** and ***clone()*** to the class *HeapPriorityQueue.*

Exercise 3

Implement the PrintQueue class with the purpose of simulating a printing queue system managing documents characterised by their Id and number of pages. Use an inner class Document with these attributes. Attached to each document there is a numeric priority, according to the type of the user that requests its printing.

Add the following methods to this class:

addDoc2Queue() - add a Document to the printing queue
send2Printer() - send a Document to printer, removing it from the queue
nextDoc2Print() - returns the next Document in line to be printed
time2print() - returns the estimated time before the printing of a specific document starts, considering that the printer takes in average 2 seconds to print each page