Guillermo Escobar P cod 256243

Exercises 6.5-3

Write pseudocode for the procedures HEAP-MINIMUM, HEAP-EXTRACT-MIN, HEAP-DECREASE-KEY, and MIN-HEAP-INSERT that implement a min-priority queue with a min-heap.

Solucion :

El seudocodigo es :

```
Heap-Minimum(A)
return A[1]

Heap-Extract-Min(A)
if heap-size[A] < 1
   then error "heap underflow"
min ← A[1]
A[1] ← A[heap-size[A]]
A[heap-size] ← A[heap-size[A]]-1
Min-heapify(A,1)
return min

Heap-Decrease-Key(A,i,key)
if A[i] < key
    then error "new key is greater than current key"
A[i] ← key
while i > 1 and A[i] < Parent(i)
   do exchange A[i] ←→A[Parent(i)]
   i ← Parent(i)

Min-Heap-Insert(A,key)
heap-size[A] ← heap-size[A] + 1
A[heap-size[A]]←∞
Heap-Decrease-Key(A,heap-size[A],key)
```

Exercises 6.5-4

Why do we bother setting the key of the inserted node to -∞ in line 2 of MAX-HEAP-INSERT when the next thing we do is increase its key to the desired value?

Solucion :

Since line 1 of the `Heap-Increase-Key` procedure needs a value for `A[i],` the one to be inserted. Moreover, line 2 of the same procedure requires that such a value has to be smaller than the one to be increased to. Thus, we simply set it to be the smallest possible one.

Exercises 6.5-5

Argue the correctness of HEAP-INCREASE-KEY using the following loop invariant:

- At the start of each iteration of the **while** loop of lines 4-6, the array $A[1 \ldots heap\text{-}size[A]]$ satisfies the max-heap property, except that there may be one violation: $A[i]$ may be larger than $A[\text{PARENT}(i)]$.

Exercises 6.5-6

Show how to implement a first-in, first-out queue with a priority queue. Show how to implement a stack with a priority queue. (Queues and stacks are defined in Section 10.1.)

Exercises 6.5-7

The operation HEAP-DELETE(A, i) deletes the item in node i from heap A. Give an implementation of HEAP-DELETE that runs in $O(\lg n)$ time for an n-element max-heap.

Exercises 6.5-8

Give an $O(n \lg k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists. (*Hint:* Use a min-heap for k-way merging.)

Problems 6-1: Building a heap using insertion

The procedure BUILD-MAX-HEAP in Section 6.3 can be implemented by repeatedly using MAX-HEAP-INSERT to insert the elements into the heap. Consider the following implementation:

```
BUILD-MAX-HEAP'(A)
1  heap-size[A] ← 1
2  for i ← 2 to length[A]
3     do MAX-HEAP-INSERT(A, A[i])
```

a. Do the procedures BUILD-MAX-HEAP and BUILD-MAX-HEAP' always create the same heap when run on the same input array? Prove that they do, or provide a counterexample.
b. Show that in the worst case, BUILD-MAX-HEAP' requires $\Theta(n \lg n)$ time to build an n-element heap.