Algoritmia y Complejidad, Laboratorio 2

Diego Quan, UFM

August 9, 2018

1 Problema 1: HeapSort

El código de Python *HeapSort.py*, itera sucesivamente el arreglo del heap e imprime el arreglo ordenado y las veces que se itero a traves de dicho arreglo. En el ejemplo proveído en el laboratorio, este itera 7 veces a traves del arreglo.

2 Problema 2: Check Heap

En el archivo *CheckHeap.py*, el programa revisa si el arreglo utilizado como parametro es un *MaxHeap* o no. En el caso del ejemplo el arreglo no es.

3 Problema 3: Iterative Heapify

Algorithm 1: Iterative Heapify

```
1  Function Left(i: indice):
2  |  return (2i)
3  end
4  Function Right(i: indice):
5  |  return (2i + 1)
6  end
7  Function IterativeHeapify(A: arreglo, i: indice):
8  |  while i <= size[A] do
9  |  |  l = Left(i);
10 |  |  r = Right(i);
11 |  |  if A[l] >= A[i] then
12 |  |  |  largest = l
13 |  |  else
14 |  |  |  largest = i;
15 |  |  end
16 |  |  if A[r] >= A[largest] then
17 |  |  |  largest = r
18 |  |  end
19 |  |  if largest! = i then
20 |  |  |  A[i], A[largest] = A[largest], A[i]
21 |  |  else
22 |  |  |  break loop
23 |  |  end
24 |  end
25 end
```