Maman 12 - Semester 05b

Question 4. Professor Pinocchio suggests a new data structure: “Pinocchio's Heap”, the data-structure has the same performance as the Fibonacci heap and in addition it supports the increase-key(H, x, k) operation that increases the value of the node x in the heap H to k. The amortized-cost (“alut leshiurin”) of the increase-key operation is $O(1)$. Do you believe that Professor Pinocchio's data-structure really exists? justify your answer.

(Note that in the Hebrew version of the question you were not asked about amortized-cost but rather on worst case cost, however if the answer to the amortized-cost version is negative, clearly this would be the answer of the original question as well).

Answer. The standard way for proving lower-bounds (e.g. an implementation of a given data-structure cannot be too efficient) is by a reduction to another problem that has a proven lower-bound. The lower bound we will use is the following:

Theorem 1 *In a model that only allows comparisons between elements ("comparison model": the algorithm decisions are based only on the results of comparison between the input elements), the running time of any algorithm for sorting n elements is $\Omega(n\log(n))$.*

Remark: A formal way to define the comparison model: the branching of the program run depends only on results of comparisons between input elements.

We will prove that the existence of a Pinocchio's Heap which is based on comparisons between elements (such as the implementations we have seen of Binomial and Fibonacci heaps) contradicts the above theorem. Note that in order to get contradiction we must assume that Pinocchio's algorithm can be run in the "comparison model".

Note that Theorem 1 does not contradict an implementation of a Pinocchio's Heap if this implementation can not be run in the "comparison model" (e.g. if the program decisions depend also on other things but comparisons).

Assuming that there exists an implementation of a Pinocchio's Heap (in the "comparison model") the following algorithm sorts a n elements array in time $O(n)$ (in the "comparison model").

```
$Sort(a_1, \dots, a_n)$

1. Let $H$ be a Pinocchio's Heap. Insert $a_1, \dots, a_n$ into $H$, while doing it find $m \overset{\text{def}}{=} \max\{a_1, \dots, a_n\}$.
2. for $i = 1$ to $n$:
    (a) Let $x =$ find-min($H$).
    (b) Print $x.key$.
    (c) increase-key($H, x, m$)
```

It is easy to see that the above algorithm does only comparisons between the elements $a_1, \dots, a_n$ and that its running time is $O(n)$. Thus it contradicts Theorem 1 and hence Pinocchio's Heap that does only comparisons does not exist.