Homework 03: Binary Heaps

Roberto Corti

July 11, 2020

Exercise 1

By modifying the code written during the last lessons, provide an array-based implementation of binary heaps which avoids to swap the elements in the array A.
(*Hint*: use two arrays, key pos and rev pos, of natural numbers reporting the position of the key of a node and the node corresponding to a given position, respectively)

This new two-array based representation of the binary heap has been implemented inside this folder. I take as reference the same implementation done during the Homework 02.
Inside the struct `binheap_type` I add two new members: `key_pos` and `rev_pos`. The first stores in its i-th position the index inside the node's array of the i-th node, while the second stores in the i-th position the index in the heap order of the i-th element of the node's array. With this new implementation the functions `extract_min`, `decrease_key`, `heapify` and `print_heap` don't modify the array of nodes `A` and all the operations related to the nodes involve just `key_pos`.

Exercise 2

Consider the next algorithm:

```
Ex2(A):
  D ← build(A)
  while do ¬is_empty(A)
    extract_min(A)
  end while
```

where `A` is an array. Compute the time-complexity of the algorithm when:

- `build`, `is_empty` $\in \Theta(1)$, `extract_min` $\in \Theta(|D|)$;
- `build` $\in \Theta(|A|)$, `is_empty` $\in \Theta(1)$, `extract_min` $\in O(\log |D|)$;

In the first case, calling the function `build` will costs $\Theta(1)$, then the `while`-loop statement will be called $|D|$ times since `extract_min` will remove at every iteration only one node. However, this last function costs $\Theta(|D|)$, so the cost of performing this function will be $\Theta(|D|^2)$. More formally,

$$T(|D|) = \Theta(1) + \sum_{i=1}^{|D|}(\Theta(1) + \Theta(|D|)) = \Theta(1) + \Theta(|D|) + \Theta(|D|^2) \in \Theta(|D|^2)$$

In the second case, assuming that $|A| = n = |D|$, the `while`-loop statement will be called n times, but now `extract_min` is a$O(\log n)$, then the overall cost of this part will raise to $O(n \log n)$:

$$T(n) = \Theta(n) + \sum_{i=1}^{n}(\Theta(1) + O(\log n)) = \Theta(n) + O(n \log n) \in O(n \log n)$$