03-Report

May 28, 2020

1 Binary Heaps: Homework 2

1.1 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)

Solution: By using the code and following the hint, `void *key_pos` added to struct of the `binary_heap` and after that to be able to avoid swapping memory, `swap_keys` are modified by using the defined array which is `key_pos`. Therefore all changes are made in `key_pos` not `A`.

1.2 Exercise 2

Consider the next algorithm:

```
def Ex2 (A)
    D ← build (A)
    while ¬ is_empty (D)
        extract_min (D)
    endwhile
enddef
```

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` $\in \mathcal{O}(\log n)$;

Solution: In the first case, `D ← build (A)` block will cost $\Theta(1)$ and while loop will be run $|D|$ times. Since complexity of `extract_min` is $\Theta(|D|)$, overall complexity will be $\Theta(|D|^2)$

In the second case, `D ← build (A)` block will cost $\Theta(|A|)$ and again while loop will be run $|D|$ times. However this time complexity of `extract_min` is $\mathcal{O}(\log n)$. So overall complexity will be upper bounded by the complexity of `extract_min` which is $\mathcal{O}(|D|.\log n)$