Homework 2 on Heaps

Edoardo Alessandroni

Exercise 1:

The code written during the lessons have been modified to avoid explicitly swapping the elements in the array A. This is accomplished by using 2 other arrays: `key_pos`, which return the position in the array A (ie the position of the key) of a given node given as input, and `rev_pos`, which reversely with respect to `key_pos` returns the node corresponding to a given position in the array A.
All of this required changing some macros and the functions `swap`, `build_heap`, `delete_heap`, `insert` and `decrease_key`.
After that, we can test again the execution time and compare a plain array and two kinds of heaps: the new one (called in the table heap_opt) and the old one (called heap).

size	array	heap	heap_opt
0	0.000004	0.000018	0.000021
1820	1.322257	0.195529	0.043853
3640	5.298625	0.407519	0.096019
5461	11.948188	0.639430	0.152205
7281	21.253756	0.890647	0.205707
9102	36.783261	1.048974	0.261656
10922	51.955237	1.464539	0.313633
12743	65.118875	1.764456	0.376570
14563	91.919831	2.044871	0.440642
16384	110.092251	3.940060	0.505217

As we can observe, `heap_opt` is better than `heap`, which is much better than `array`.
Regarding the complexity of the full algorithm of building the data structure and removing the min until it's empty (see text of the following exercise), as we'll see in the next theoretical exercise, working with arrays has a quadratic dependence with respect to `n`, while working with heaps has a complexity $O(n\ log\ n)$, which is clearly better.
Finally, the improvement of `heap_opt` basically comes from reducing the constant which multiplies the function of n, but the function itself stays the same.

Exercise 2:

- `build`, `is_empty` $\in \Theta(1)$, `extract_min` $\in \Theta(|D|)$

Just summing up the different contributions (one `build`, $|D|+1$ times `is_empty` and $|D|$ times `extract_min`), calling $n = |D|$.

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

where the summation comes from the fact that `extract_min` is applied to a data structures that decreases in size.

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

  Summing up the various contributions as before, with similar calculations:

$$T(|A|,|D|) = \Theta(|A|) + (|D|+1)\Theta(1) + \sum_{i=1}^{|D|} O(log\ i) \leq \Theta(|A|) + \Theta(|D|) + O(|D|log|D|)$$
$$\in \Theta(|A|) + O(|D|log|D|)$$

  Moreover, if we assume that $|A| \propto |D|$, then:

$$T(|D|) \in O(|D|log|D|)$$