Binary Heaps

Exercise 1

In order to implement this new version of the binary heap to avoid swaps in `A`, I added into the `binheap_type` struct the two auxiliary arrays of integers: `key_pos` and `rev_pos`.

A Makefile can be produced by using `cmake` as follows:

```
cmake -G "Unix Makefiles" CMakeLists.txt
```

Compiling the code, it is possible to test the performance by executing the programs in folder `tests`.

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` $\in$ O(log |D|).

In the first case, the function `build` takes $\Theta(1)$, `is_empty` takes $\Theta(1)$ and it is called $|D|$ times since `extract_min` removes one node at time. Thus, we have:

$T_{Ex2}(|D|) = \Theta(1) + \sum_{i=1}^{|D|}(\Theta(1) + \Theta(|D|)) = \Theta(1) + \Theta(|D|) + \Theta(|D^2|)$.

Thus, in the first case the overall time-complexity of `Ex2` function belongs to $\Theta(|D^2|)$.

In the second case, the function `build` takes $\Theta(|D|)$, `is_empty` takes $\Theta(1)$ whereas `extract_min` takes O(log |D|). Thus, we have:

$T_{Ex2}(|D|) = \Theta(|D|) + \sum_{i=1}^{|D|}(\Theta(1) + O(log|D|)) = \Theta(|D|) + \Theta(|D|) + O(|D|log|D|)$.

Thus, in the second case the overall time-complexity of `Ex2` function belongs to $O(|D|log|D|)$.