Data Structures, excercises week 9, 29-31.3 ja 8-9.4

Note: Easter holiday 1-7.4.

1. Give an algorithm for finding the *smallest* element in a max-heap. Assume the usual array implementation for the heap. The algorithm should examine as few elements in the heap as possible.

2. Give an algorithm that prints all elements in a max-heap that are greater than a given value x. The contents of the heap should remain unchanged. Your algorithm should run in time linear in the number of printed elements.

3. (a) Define operation `heap-delete-key(H,i)`, that deletes from a heap the key at position i. What is the time and space complexity of the algorithm?

   (b) Define operation `heap-change-key(H,i,k)` that changes the key at position i to value k:n. The value of the key can increase or decrease. What is the time and space complexity of the algorithm?

   Bonus: how `heap-change-key(H,i,k)` could be implemented without recursion?

4. How could an operation `heap-inc-all-keys(k)` that increases every key with the value k be defined such that it would only take time $O(1)$ and the time complexity of the rest of the operations would stay unchanged?

5. Implement max-heap with some programming language. Implement at least operations `heapify`, `heap-insert`, `heap-max` and `heap-del-max`.

6. Implement with some programming language heap sort, insertion sort and bubble sort. Measure the performance of the algorithms with inputs of various size. Do the measurements such that the input is

   - at random order
   - already in increasing order
   - in decreasing order