6. Problem set on

Algorithms, Data Structures and Data Abstraction WS 18/19

Wolfgang Mulzer

Due on 30. November 2018, 10:00, in the respective TA's mailbox

Problem 1 Further Heap-Operations *10 Points*

(a) Write code for a binary heap with n elements, stored as an array. Also, implement the following two operations, with worst-case running time $O(\log n)$.

- `decreaseKey`(i, k): Set the key at position i to k and restore the heap property (Precondition: k is less than or equal to the old key at i).
- `delete`(i): Remove the element at position i and restore the heap property.

Both `decreaseKey` and `delete` may use `bubble-up` and `bubble-down`. Briefly justify correctness and running time of your implementation.

(b) We would like to extend the interface `PriorityQueue` with the additional operations from (a). Explain briefly why an efficient implementation needs a "pointer" into the data structure (i.e., why is it not enough to pass only the old and the new key to `decreaseKey`, even if the old key appears only once?). How would you implement this without violating the information hiding principle? Define a new interface `Position` that represents a pointer into the priority queue. Explain briefly how to modify `PriorityQueue` in order to use `Position`. What problems does this create? Sketch a possible solution. (You do not need to write any code.)

Problem 2 Elementary Probability Theory *10 Points*

We have N bins, and we throw n balls into them such that each throw is independent of the others and such that each bin is hit with the same probability $1/N$.

(a) Compute the probability that bin i remains empty, for $i = 1, \ldots, N$. Furthermore, let Y_i be the indicator random variable that is 1, if bin i remains empty, and 0, otherwise. Compute the expected value $E[Y_i]$ of Y_i.

(b) Let X be the random variable that indicates the number of bins that remain empty. Compute the expected value of X using the expected values $E[Y_i]$, for $i = 1, \ldots, N$.

(c) Determine a good bound $f(N)$ so that the following holds: suppose we throw n balls into N bins, independently and uniformly at random. If $n \geq f(N)$, then with probability at least $1/2$ there is at least one bin with at least two balls.

Hint: Imagine that the n balls are thrown successively. What is the probability that the next ball ends up in an empty bin, provided that all the previous balls have ended up in an empty bin? You might find the bound $1 + x \leq e^x$ useful. It holds for all $x \in \mathbb{R}$.

Problem 3 Hashing *10 Points*

(a) Successively insert the keys 5, 28, 19, 15, 20, 33, 12, 17, 10 into a hash table with 9 positions. Use the hash function $h(k) = k \bmod 9$, and resolve conflicts with chaining.

(b) Successively insert the keys 10, 22, 31, 4, 15, 28, 17, 88, 59 into a hash table with 11 positions. Use the hash function $h(k) = k \bmod 11$, and resolve conflicts using open addressing with linear probing.

(c) Successively insert the keys 10, 22, 31, 4, 15, 29, 17, 88, 59 into a hash table with 11 positions. Resolve conflicts with cuckoo hashing, using $h_1(k) = k \bmod 11$ and $h_2(k) = (k \bmod 13) \bmod 11$.

Show the individual steps.

Hint: Pseudo-code for the hashing operations is available on the course website.