Midterm Data Structures and Algorithms 2018-2019

Friday September 28, 2018, 15.15-16.45

6 exercises

No motivation for your answers required!

Exercise 1. (*30 points*)

Indicate for every statement whether it is true or false.
If relevant for the sub-question, we consider an input-array of length n.

(a) The worst-case time complexity of insertion sort is in $\Theta(n)$.

(b) The best-case time complexity of quicksort is in $\Theta(n \log n)$.

(c) Insertion sort can be faster than quicksort.

(d) The worst-case time complexity of quicksort is in $\mathcal{O}(n \log n)$.

(e) The worst-case time complexity of merge sort is in $\mathcal{O}(n \log n)$.

(f) Counting sort can be faster than quicksort.

(g) The worst-case time complexity of heapsort is in $\Theta(n^2)$.

(h) An algorithm with worst-case time complexity in $\mathcal{O}(n^2)$ is always faster than an algorithm with worst-case time complexity in $\mathcal{O}(n \log n)$.

(i) For the worst-case, selection sort is asymptotically faster than heapsort.

(j) Counting sort is a stable sorting algorithm.

Exercise 2. (*5+5+5 points*)
This exercise is concerned with merge sort. We assume that the subroutine 'merge' merges two sorted sub-arrays into one sorted sub-array, and has time complexity in $\Theta(n)$ with n the total number of merged elements.

(a) Give a recurrence equation for the worst-case time complexity $T(n)$ of merge sort with n the number of elements in the input-array.

(b) What is the height of the recursion tree for your recurrence?

(c) The algorithm for merge sort uses the divide-and-conquer programming paradigm. Give an example of another sorting algorithm that also follows that paradigm.

Exercise 3. (*5+5 points*)
This exercise is concerned with quicksort. We assume that the subroutine 'partition' uses the last element of the sub-array it will partition as pivot.

(a) Give an example of an input-array of length 5 that 'partition' splits into a sub-array of 4 elements and a sub-array of 0 elements (so an input-array that intuitively gives an unbalanced partitioning.)

(b) What is the worst-case time complexity of quicksort in terms of Θ?

Exercise 4. (*5+5 points*)
We consider a decision tree for insertion sort for an input-array of length n.

(a) What is the minimum length of a path from the root to a leaf?

(b) How many leaves should we have at least?

Exercise 5. (*6+5+4 points*)
This exercise is concerned with heaps and heapsort.

(a) Give (in a picture) three different max-heaps with keys $1, 2, 3, 4, 5, 6, 7$.

(b) Give a max-heap H with the following property:

Applying HeapExtractMax (which removes and returns the maximum key) to H returns 10, and yields as remaining max-heap $[6, 3, 5, 2, 1]$.

(c) What is, in terms of $\mathcal{O}$, the worst-case time complexity for turning an array of length n into a max-heap, using the bottom-up max-heap construction?

Exercise 6. (*5+5 points*)
This exercise is concerned with max-priority queues, with update operations for insert (an element with arbitrary key), and delete (an element with maximum key).

(a) We implement a max-priority queue using an array, maintaining an increasing order on the keys.

Which update operation can be done in elementary time, and which update operation can take a lot of time?

(b) We implement a max-priority queue using a max-heap.

What are, in terms of $\mathcal{O}$, the worst-case time complexities for the update operations insert and delete?

The mark for the midterm is (the total number of points plus 10) divided by 10.