Midterm Data Structures and Algorithms 2019-2020

Thursday September 26, 09.00-10.45

6 exercises

No motivation asked unless otherwise specified

Exercise 1. (8 points)

We sort an array A of length n. Give in terms of Θ

(a) the worst-case time complexity of insertion sort,

(b) the best-case time complexity of insertion sort,

(c) the worst-case time complexity of merge sort,

(d) the worst-case time complexity of heapsort,

(e) the worst-case time complexity of quicksort,

(f) the best-case time complexity of quicksort,

(g) the worst-case time complexity of counting sort, assuming the keys are in $\{0, \ldots, k\}$,

(h) the average-case time complexity of bucket sort, assuming the keys are uniformly distributed over $[0, 1)$.

Exercise 2. (8 points)

Indicate for every sentence whether it is true or false.

(a) Swapping two elements of an array can be done in constant time.

(b) The smallest element of a max-heap is at the leftmost leaf.

(c) The worst-case time complexity of any algorithm for turning an array into a max-heap is in $\Omega(n \log n)$.

(d) The worst-case time complexity of any comparison sort algorithm is in $\Omega(n \log n)$.

(e) Heapsort is a divide–and–conquer algorithm.

(f) Insertion sort can be faster than merge sort.

(g) Heapsort is asymptotically optimal.

(h) Bucket sort needs additional memory.

Exercise 3. (2+2+2)
(The picture-notation and array-notation are both ok.)

(a) Give all possible max-heaps with elements $1, 2, 3, 4$.

(b) Give an example of an input A consisting of 7 different elements that, together with input-index $i = 1$, gives worst-case behaviour of MaxHeapify.

(c) Give an example of an input A consisting of 7 different elements that gives worst-case behaviour of BuildMaxHeap.

Exercise 4. (1+1+2+2)

(a) When does the worst-case behaviour of quicksort occur?

(b) When does the best-case behaviour of quicksort occur?

(c) Give a recurrence equation for the function $T(n)$ for the best-case time complexity of quicksort.

(It is not asked to solve the recurrence equation.)

(d) Give a recursion tree for the best-case time complexity of quicksort.

Exercise 5. (2+2 points)

(a) Give a concrete example showing that radix sort is not correct if the subroutine for sorting is not stable.

(b) Describe the divide-conquer-combine steps of merge sort.

Give a short description, approximately one sentence per step.

Exercise 6. (2+2 points)

(a) We implement a stack using an array S of size N.

Give pseudocode for the operation push that takes as input a stack S and an element x, and that adds x to S; include a test on overflow.

(b) We implement a queue using a circular array Q of size N.

Give the test for 'the queue is full' in pseudocode.

The grade is $(\frac{n}{36} \times 9) + 1$ *for* n *the number of obtained points.*