Resit Data Structures and Algorithms 2018-2019

Tuesday January 8, 2019, 18.30–21.15

6 exercises

Shortly explain your answers unless otherwise specified

Exercise 1. (*5+5+5 points*)

This exercise is concerned with sorting.

(a) Give the worst-case time complexity in terms of $\mathcal{O}$ for sorting an array of n numbers using (i) selection sort, (ii) quicksort, (iii) merge sort, (iv) counting sort, (v) heapsort. No motivation needed.

(b) Give the recursion tree for the application of merge sort to the input array $[1, 5, 6, 4, 8, 7, 2, 3]$.

(c) Give a recurrence equation describing the worst-case time complexity of merge sort.

Solve your recurrence equation to give the worst-case time complexity of merge sort in terms of Θ.

Exercise 2. (*6+6+3 points*)

This exercise is concerned with heaps.

(a) Consider the max-heap $H = [9, 8, 6, 4, 7, 5, 1, 2, 3]$. Apply 'on the fly' the algorithm HeapExtractMax (that removes and returns the maximum element of a max-heap) to H; give your answer in pictures.

(b) Apply 'on the fly' the algorithm for bottom-up max-heap construction to the array $A = [1, 2, 3, 4, 5, 6, 7]$; give your answer in pictures.

(c) What is the worst-case time complexity in terms of $\mathcal{O}$ for the algorithm for bottum-up max-heap construction on an input-array of length n?

Exercise 3. (*6+4 points*)

(a) Give an implementation (use pseudo-code) of a queue with operations enqueue and dequeue, using a singly linked list, where we have the following: for an element x in the list, we have operations $x.next$ and $x.key$ with the suggested meaning. For a list L we have operation $L.head$.

Give (no motivation needed) the worst-case time complexity of your operations.

(b) Does an algorithm A with worst-case time complexity in $\mathcal{O}(n)$ always perform better than an algorithm B with worst-case time complexity in $\mathcal{O}(n^2)$? (Explain your answer.)

Exercise 4. *(5+5+5+5 points)*

This exercise is concerned with binary search trees and AVL-trees.

(a) Give in pictures all binary search trees with labels 1, 2, 3.

(b) What is the worst-case time complexity, in terms of $\mathcal{O}$, of adding an element to (i) a Binary Search Tree (BST) consisting of n elements, (ii) a min-heap consisting of n elements, (iii) a AVL-tree consisting of n elements? No motivation needed.

(c) Consider binary search trees implemented as a linked structure (for example, $v.key$ is the key at node v).

Give pseudo-code for a non-recursive procedure for searching a key in a a sub-tree of a binary search tree. The inputs for the procedure are a pointer x to a node in the binary search tree, and a key k. The output is a pointer to a node with key k if such node exists in the sub-tree rooted at node x, and nil otherwise.

(d) Construct an AVL-tree by inserting one by one the keys

3 5 4 2 1 6 7

starting from the empty tree. After each insertion, rebalance the tree if needed. Give your answer in pictures.

Exercise 5. *(5+5+5 points)*

This exercise is concerned with greedy choice.

(a) Consider the knapsack01 optimization problem: given a set S of items all with benefit and weight, and given a maximum weight W, select an optimal choice $C \subseteq S$ in the sense that the total benefit of C is maximal under the constraint that the total weight of C does not exceed W.

Give a small example showing that the greedy choice for an item with maximal benefit does not necessarily yield an optimal solution.

(b) Consider the activity selection problem: given a finite set of activities all with start time and finish time, select a maximum-size subset of mutually compatible activities.

Argue that the greedy choice for an activity with minimal end time yields an optimal solution.

(c) We consider the following problem: we are given a finite set of activities, each with start time and finish time. We have available infinitely many lecture rooms.

Describe (informally but precisely) a greedy algorithm for scheduling all activities using a minimum number of lecture rooms.

Exercise 6. *(5+4+6 points)*

(a) Consider the algorithm for a longest common subsequence (LCS) of input sequences $X = \langle x_1, \ldots, x_m \rangle$ and $Y = \langle y_1, \ldots, y_n \rangle$:

```
Algorithm LCS(X, Y):
    new array C[0...m, 0...n]
    for i := 0 to m do
        C[i, 0] := 0
    for j := 0 to n do
        C[0, j] := 0
    for i := 1 to m do
        for j := 1 to n do
            if x_i = y_j then
                C[i, j] := C[i − 1, j − 1] + 1
            else
                C[i, j] := max(C[i, j − 1], C[i − 1, j]
    return C
```

Apply the algorithm to the following input: $X = \langle A, B, C, D, E, F \rangle$ and $Y = \langle A, B, D, F, C, D \rangle$. Give your answer in the form of a table and give explicitly the longest common subsequence(s) that is (are) found.

(b) What is the worst-case time complexity of the LCS algorithm of 6(a) in terms of $\mathcal{O}$? (Explain your answer.)

(c) Describe an algorithm that takes as input an array of integers and that finds a longest increasing subsequence of the input-array. What is the worst-case time complexity of your algorithm?