Information

Date: Sunday, February 5, 2023
Time: 9:00—14:00
Teachers: Pontus Ekberg, Sarbojit Das, Stephan Spengler

This re-exam has 8 problems, for a total of 100 points. The grade thresholds are

- 50 points for grade 3,
- 75 points for grade 4,
- 90 points for grade 5.

Important instructions:

1. No books/notes or computer/phone/calculator allowed, only writing tools (pencil, etc.).
2. Only write on one side of each sheet of paper.
3. Start your answers to each problem on a new sheet of paper.
4. Hand in your answer sheets sorted in the same order as the problems.
5. Answer questions precisely and concisely. Avoid excessively long answers.
6. Write your answers in English.
7. Draw figures when requested, but feel free to do so for other questions as well, if you think they help.
8. When writing pseudocode you can use a reasonable syntax of your choice, but make sure that the meaning is unambiguous.
9. When asked to describe a new algorithm, you are allowed to use algorithms taught in the course as subroutines.
10. You can keep this question sheet after the exam.

Good luck!

Problem 1 (12 points in total)

For each of the claims below, state whether it is *true* or *false*. No motivation is needed.

a) $n^3 = O(n^2)$ (1)

b) $n^3 = \Omega(n^2)$ (1)

c) $n^3 = \Theta(n^2)$ (1)

d) $n \log n = \Omega(n^2)$ (1)

e) $n^{10} = O(2^n)$ (1)

f) $n^5 = O(n!)$ (1)

g) $100n^4 = \Theta(n^4)$ (1)

h) $3^n = O(2^n)$ (1)

i) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$ (2)

j) If $f(n) = O(n^2)$, then $f(n) = O(n^3)$ (2)

Problem 2 (18 points in total)

a) Give the *worst-case* running time of INSERTION-SORT in Θ-notation. (2)

b) Describe what the input arrays that give rise to the *worst-case* running time of INSERTION-SORT look like. Explain clearly why these inputs lead to the worst-case running time given in Problem 2a) above. (7)

c) Give the *best-case* running time of INSERTION-SORT in Θ-notation. (2)

d) Describe what the input arrays that give rise to the *best-case* running time of INSERTION-SORT look like. Explain clearly why these inputs lead to the best-case running time given in Problem 2c) above. (7)

Problem 3 (10 points in total)

a) Give the worst-case running time of BINARY-SEARCH in Θ-notation. (2)

b) What is the maximum number of comparisons done by BINARY-SEARCH on an array with $2^{30} - 1 = 1,073,741,823$ elements? Explain your answer clearly. (3)

c) Explain clearly how BINARY-SEARCH proceeds if we call BINARY-SEARCH$(A, 10)$, where (5)

$$A = [2, 3, 5, 6, 8, 10, 14, 17, 22, 45, 99, 100, 102, 200, 250].$$

Make sure to explain which elements in A are compared with, and in which order.

Problem 4 (10 points in total)

Simulate MERGE-SORT on the following array A:

$$A = [7, 5, 10, 12, 2, 9, 3, 1]$$

You should show all intermediate arrays created, and the relationship between them. You do *not* need to show intermediate array states inside loops (i.e., inside the MERGE procedure).

Problem 5 (15 points in total)

Draw the Binary Search Trees (BSTs) that we would get after performing the following list of tree operations on an initially empty BST, in the order that the operations are written. The operations (INSERT/DELETE) should be as given in the lectures and the text book, or otherwise clearly explained and motivated.

(For each of problems a—c below, start with a new empty BST.)

a) INSERT(1), INSERT(2), INSERT(3), INSERT(4), INSERT(6), INSERT(5) (2)

b) INSERT(3), INSERT(2), INSERT(5), INSERT(4), INSERT(1), INSERT(6) (2)

c) INSERT(3), INSERT(2), INSERT(5), INSERT(4), INSERT(1), INSERT(6), DELETE(3) (3)

d) The height of a BST is the number of edges in the longest path between the root and any leaf. What are the minimum and maximum possible heights of a BST with n nodes? Explain your answers. (4)

e) What are the minimum and maximum possible number of leaves in a BST with n nodes? Explain your answers. (4)

Problem 6 (14 points in total)

Assume that we have a hash table of size 12. The hash function used is $h(k) = k \mod 12$ (let the hash table be indexed starting from 0).

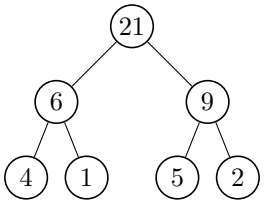
a) Let the hash table use *chaining* for collision handling. Draw the hash table after inserting the following keys in order: 7, 11, 15, 18, 19, 22, 13, 29 (4)

b) Let the hash table instead use *linear probing* for collision handling. Draw the hash table after inserting the same keys in order: 7, 11, 15, 18, 19, 22, 13, 29 (4)

c) Assume that you have a hash table (with chaining) of size m that stores a total of n different keys. What are the best, worst, and average running times for a key lookup in this hash table in Θ-notation? Explain your answers. For the average running time you can assume that the keys are evenly distributed in the hash table. (6)

Problem 7 (11 points in total)

Let H be the max-heap shown here:



a) Show how H is stored in an array as a well-indexed tree. Make sure that it is clear which node is stored at which array index. (3)

b) In general, if a node of a heap (as a well-indexed tree) is stored in array index i, at which array indices are the left and right children of that node stored (assuming both children exist)? (3)

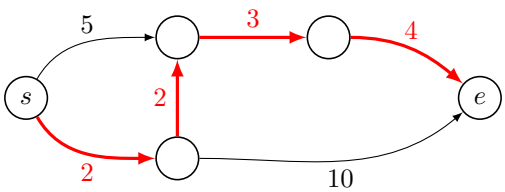
c) Draw the resulting heap after inserting the key 7 into H. (2)

d) Draw the resulting heap after running EXTRACT-MAX on H (use the original heap H here, not the one resulting after the insertion in the previous question). (3)

Problem 8 (10 points in total)

Consider the following problem. We are given a Directed Acyclic Graph (DAG) G, where every edge is labeled by a positive number denoting the cost of traveling along that edge. We want to find a path from a given start node s to a given end node e with minimal total cost.

In the example DAG given below the cheapest path would be the marked path with total cost $2 + 2 + 3 + 4 = 11$. Note that if every edge has the same cost, then the cheapest path is simply the path with the fewest number of hops, which we could find with the ordinary DFS algorithm, for example.



Propose an algorithm CHEAPEST-PATH(G, s, e) that prints a path with minimal total cost from s to e in G. If there is no path from s to e, then it should print nothing. Describe your algorithm in pseudocode or unambiguous English. Explain clearly why the algorithm is correct (i.e., why it prints the cheapest path) and what its worst-case running time is. Partial or overly inefficient solutions can give partial credit.