Information

Date: Wednesday, August 23, 2023
Time: 8:00—13:00
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. In general, partial solutions can give partial credit.
7. Write your answers in English.
8. Draw figures when requested, but feel free to do so for other questions as well, if you think they help.
9. When writing pseudocode you can use a reasonable syntax of your choice, but make sure that the meaning is unambiguous.
10. If asked to describe a new algorithm, you are allowed to use algorithms taught in the course as subroutines.
11. 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) $n^2 = O(n^3)$ (1)

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

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

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

e) $n^5 = O(1.5^n)$ (1)

f) $n^{10} = O(n!)$ (1)

g) $500n^2 = \Theta(n^2)$ (1)

h) $4^n = O(2^n)$ (1)

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

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

Problem 2 (14 points in total)

a) Give the best-case running time of MERGE-SORT in Θ-notation. Explain and motivate your answer clearly. (7)

b) Give the worst-case running time of MERGE-SORT in Θ-notation. Explain and motivate your answer clearly. (7)

Problem 3 (10 points in total)

Say that you have a computer capable of carrying out 2^{30} operations per second. Also say that you have an algorithm ALGO that takes exactly 2^n operations to terminate on input instances of size n.

a) What is the maximum input instance size that could be solved using ALGO on your computer in one second? (2)

b) What is the maximum input instance size that could be solved using ALGO on your computer in two seconds? (2)

c) What is the maximum input instance size that could be solved using ALGO on your computer in one year? (A year is approximately 2^{25} seconds.) (3)

d) What is the maximum input instance size that could be solved using ALGO in one second on a computer that is 1024 times as fast? (3)

Problem 4 (10 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(4), INSERT(3), INSERT(5), INSERT(6) (3)

b) INSERT(4), INSERT(3), INSERT(1), INSERT(7), INSERT(5), INSERT(8) (3)

c) INSERT(4), INSERT(3), INSERT(1), INSERT(7), INSERT(5), INSERT(8), DELETE(4) (4)

Problem 5 (18 points in total)

a) 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)

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

c) Write down an algorithm HEIGHT(T) that takes a binary search tree (BST) T as input and returns the height of T. Explain why the algorithm works and give its worst-case running time in O-notation. (10)

Write your algorithm in pseudocode or in unambiguous English. For full credit the algorithm should have a worst-case running time of $O(n)$, where n is the number of nodes in T.

Problem 6 (16 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).

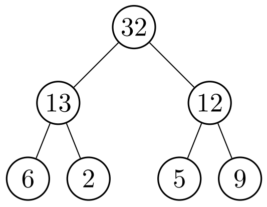
a) Let the hash table use *chaining* for collision handling. Draw the hash table after inserting the following keys in order: 7, 14, 19, 17, 9, 31, 5, 29 (5)

b) Let the hash table instead use *linear probing* for collision handling. Draw the hash table after inserting the same keys in order: 7, 14, 19, 17, 9, 31, 5, 29 (5)

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 (12 points in total)

Let H be the max-heap shown here:



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

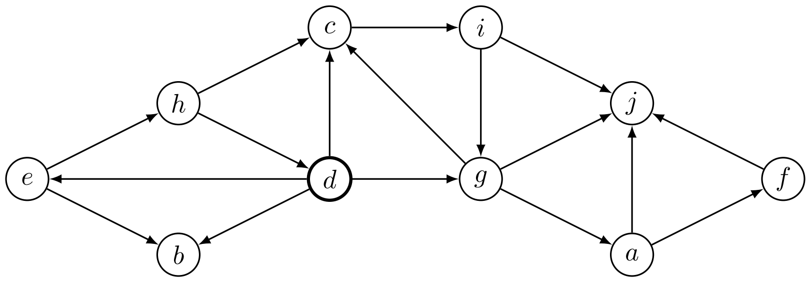
b) In general, if a node of a heap (as a well-indexed tree) is stored in array index i, at which array indices are the left and right children of that node stored (assuming both children exist)? (3)

c) Draw the resulting heap after inserting the key 15 into H. (3)

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 (8 points in total)

Consider the directed graph G with node set $V = \{a, b, c, d, e, f, g, h, i, j\}$ shown here:



We will use the Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms, but let the nondeterministic choices usually present in those algorithms be determined by always looping over adjacent nodes in alphabetical order. (For example, when looping over the adjacent nodes to node a in either BFS or DFS, node f will be considered in the loop before node j, since "f" comes before "j" in the alphabet.)

a) Starting in node d, state the order in which all nodes are discovered by BFS. (4)

b) Starting in node d, state the order in which all nodes are discovered by DFS. (4)