MATEMATISKA INSTITUTIONEN
STOCKHOLMS UNIVERSITET
Avd. Beräkningsmatematik
Examinator: Lars Arvestad

Take-home exam in
DA3018 Computer Science for Mathematicians
7.5 hp
2020-05-25

- **Write clearly.** Hard-to-read answers risk zero points.
- Justify all your answers (unless otherwise stated).
- If you search for information in books or on the internet, clearly state your sources!
- It is not allowed to ask questions in online forums of any sort.
- **Grading thresholds:** E: 20, D: 24, C: 28, B: 32, A: 36

Mandatory preparatory assignment

Read, reflect, and adapt the following text (replace `<your_name>` with your name) to the top of your hand-in!

> *I, `<your_name>`, guarantee that I have solved the assignments on this take-home exam independently and without help from some else. In particular, I have not discussed the exam and my solutions with anyone.*

This is mandatory and required for a passing grade on the exam.

Assignments

1. State and justify the time complexity of the following problems. Also state what assumptions you might be making about the computational model (i.e., what the elementary operations are and what time they take).

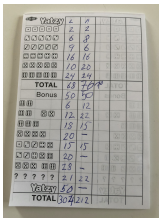
(a) Sort n strings, each of length L. (2p)

(b) Find the smallest element in a list of n floats. (2p)

(c) Find the smallest element in a list of n floats by creating a min-heap from the list elements and then looking at the top. (2p)

(d) Simulate a game of Yahtzee with two participants. (2p)

See `https://en.wikipedia.org/wiki/Yahtzee` for US (?) Yahtzee rules. You can also use Scandinavian rules.



The protocol for a game of Yahtzee. (I won.)

2. A *Magic square* is an $n \times n$ matrix filled with distinct elements from $[1, n^2]$ such that each row sum, column sum, and the two diagonal sum are equal. The sum is unique: $n(n^2+1)/2$. In an exam for the course *Programming paradigms*, two days ago, there was an assignment to write a program (in Prolog) to find magic squares using a naive algorithm:

- Generate every possible matrix M containing the elements $[1, n^2]$
- and test if M is a magic square at each iteration.
- Stop and output M when a magic square is found.

Make a worst-case time-complexity analysis of this algorithm! (5p)

3. Hash-based datastructures (such as HashMap in Java and dictionary in Python), that insert pairs of key and value, (k, v), tend to require that the key cannot be modified. Why is that? (3p)

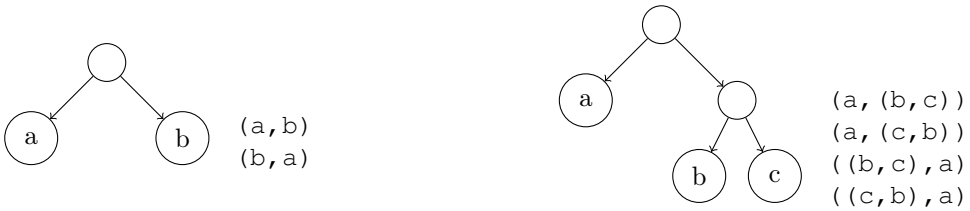
4. Suppose there is an established algorithm for a problem P that, on input of size n, does about $n^{3/2}$ calls to an expensive function f, in which most of the computation time is spent. You find a *heuristic* for P that only needs $O(n \log n)$ calls to f.

(a) Under reasonable assumptions, quantify how much computation time in percent you can expect to save when using the heuristic on input of size $n = 10^4$? (3p)

(b) Given that it is a heuristic, what kind of drawback could there be with using it? (2p)

5. Leaf-labeled rooted binary trees have a natural string representation, see Figure 1 for examples. In this problem we look at trees where all labels are unique (so n different labels are used for n leaves). Let a leaf be represented by its label, and let a tree T with a root r and subtrees T_1 and T_2 under r be represented by (T_1, T_2), but with T_1 and T_2 substituted by their string representations. So (a, b) is a tree with two leaves, a and b, and a root node. This string representation is not unique, because (a, b) represents the same tree as (b, a).

Present a *recursive* algorithm that, given a non-empty tree T, returns the number of ways T can be represented as a string. (5p)



Figur 1: Two trees that can be represented by strings in more than one way.

6. Consider the following computational problem:

- **Input:** a list of strings T corresponding to the lines of a text.
- **Output:** a list of triples (w_1, w_2, f), where f indicates the frequency of word w_2 coming after word w_1 in the text.

A possible solution could look like this:

```
frequencies(T):   // T is a list of strings
   f = new FrequencyTable() // Made-up datastructure
   for line in T:
       w1 = null
       words = line.split() // splits at non-letters
       for w2 in words:
           w2 = lowercase(w2)
           if w1:
               f.update(w1, w2)   // Increments pair w1,w2
               w1 = w2
   return f
```

What algorithmic bug is there in this pseudocode? (3p)

7. Suppose you are assigned to suggest an algorithm for sorting about 100 terabytes of data, which is 100 times more than your available computer can have in RAM at one time. You have to assume that data will be kept on disk, and remember that accessing file contents is more expensive that accessing RAM. You are specifically asked whether to use Mergesort, Quicksort, or Heapsort.

Based on what you learned in the course, justify a choice of sorting algorithm, by succinctly describing possible advantages and disadvantages with applying these three sorting algorithms to your task. (5p)

8. The *Recent relative* problem is stated as follows:

- Input: a rooted binary tree T and two leaves a and b.
- Output: the vertex v that is on both paths from the root to a and b, and as far away from the root as possible.

We say that a vertex u *dominates* a vertex v if u is on the path from the root to v.

Figure 2 suggests an algorithm for the Recent relative problem.

(a) Explain why the algorithm is correct. (3p)

(b) Analyze the algorithm's time complexity. (3p)

```
recent_relative(a, b):
    while a != b:
        if a dominates b:
            b := parent(b)
        else:
            a := parent(a)
    return a
```

Figur 2: Pseudocode for computing the recent relative of two vertices, a and b, in a rooted tree. The tree T is implied from a and b, but is not needed explicitly as input. The function *parent* returns the parent node of its argument.