KØBENHAVNS UNIVERSITET

Introduktion til diskret matematik og algoritmer: Problem Set 2

Due: Wednesday February 25 at 12:59 CET.
Submission: Please submit your solutions via *Absalon* as a PDF file. State your name and e-mail address close to the top of the first page. Solutions should be written in LaTeX or some other math-aware typesetting system with reasonable margins on all sides (at least 2.5 cm). Please try to be precise and to the point in your solutions and refrain from vague statements. Never, ever just state the answer, but always make sure to explain your reasoning. *Write so that a fellow student of yours can read, understand, and verify your solutions.* In addition to what is stated below, the general rules for problem sets stated on *Absalon* always apply.
Collaboration: Discussions of ideas in groups of two to three people are allowed—and indeed, encouraged—but you should always write up your solutions completely on your own, from start to finish, and you should understand all aspects of them fully. It is not allowed to compose draft solutions together and then continue editing individually, or to share any text, formulas, or pseudocode. Also, no such material may be downloaded from or generated via the internet to be used in draft or final solutions. Submitted solutions will be checked for plagiarism.
Grading: A score of 120 points is guaranteed to be enough to pass this problem set.
Questions: Please do not hesitate to ask the instructor or TAs if any problem statement is unclear, but please make sure to send private messages—sometimes specific enough questions could give away the solution to your fellow students, and we want all of you to benefit from working on, and learning from, the problems. Good luck!

1 (60 p) Provide formal proofs of the following claims using proof techniques that we have learned during the course.

1a Define $a_1 = 1$ and $a_n = 2a_{n-1} + 1$ for $n \geq 2$. Prove that for all positive integers $n \in \mathbb{N}^+$ it holds that $a_n = 2^n - 1$.

1b Prove that for all non-negative integers $n \in \mathbb{N}$ it holds that $3 \mid 4^n + 5$.

2 (60 p) For each of the propositional logic formulas below, determine whether it is a tautology or not. If the formula is not a tautology, show how to add a single connective to make it into a tautology. Please make sure to justify your answers (e.g., by presenting truth tables, or by using rules for rewriting logic formulas that we have learned in class).

2a $\neg\big((p \to q) \vee r\big) \to \big((\neg q \wedge \neg r) \wedge p\big)$

2b $\big((p \wedge q) \to r\big) \leftrightarrow \big((q \vee r) \vee \neg p\big)$

3 (80 p) One slightly annoying feature of the *merge sort* algorithm is that it ignores long runs of elements that are already sorted. Jakob has therefore devised an algorithm that will make use of already sorted runs, and your task in this problem is to help him analyse this algorithm.

The *merge* part of the algorithm would be essentially the same as before:

```
merge (L, R)
    m := length (L)
    n := length (R)
    M := new array of length m + n
    i := 1
    j := 1
    for k := 1 upto m + n {
        if (i <= m) {
            if (j <= n and R[j] <= L[i]) {
                M[k] := R[j]
                j    := j + 1
            }
            else {
                M[k] := L[i]
                i    := i + 1
            }
        else {
            M[k] := L[j]
            j    := j + 1
        }
    }
    return M
```

For the main recursive sorting method, however, the array should be split only when we detect the first element that is not in sorted order, which Jakob is trying to achieve as follows:

```
mergerunssort (A)
    n := length (A)
    i := 1
    while (i < n and A[i] <= A[i + 1] {
        i := i + 1
    }
    if (i < n) {
        L := new array of length i
        R := new array of length n - i
        for j := 1 upto i {
            L[j] := A[j]
        }
        for j := 1 upto n - i {
            R[j] := A[i + j]
        }
        R := mergerunssort (R)
        A := merge (L, R)
    }
    return A
```

3a Is the pseudocode algorithm above correct in that for any input array `A` it will be the case that `mergerunssort(A)` returns the same array but sorted in increasing order?

3b Regardless of whether the sorting algorithm is correct or not, does it always terminate (assuming that the input is an array of elements that can be compared with `<=`) and, if so, what is the worst-case time complexity?

3c Can you give any example of a family of input arrays of growing size for which Jakob's *merge runs sort* algorithm will output a correctly sorted array and be asymptotically faster than the *merge sort* algorithm that we have covered in class? Can you give any example of a family of input arrays of growing size for which *merge runs sort* will be asymptotically slower than standard *merge sort*?

4 (60 p) When constructing this problem set, Jakob ran into some very unfortunate problems. When he wanted to add a couple of more examples illustrating the power of inductive proofs, the proofs turned out to be a little bit too powerful. Specifically, Jakob was able to use mathematical induction to show

1. that all swans have the same colour (presumably white, so that there are no black swans after all), and
2. that all positive integers are in fact equal.

Both of these claims are fairly disturbing from a mathematical point of view. Please help Jakob by pointing out clearly what goes wrong in his induction proofs below.

4a **Theorem.** All swans have the same colour.

Proof. We prove by induction over n that any set of n swans have the same colour.

For the base case, if we have a set of $n = 1$ swan, then this swan vacuously has the same colour as itself.

For the induction step, assume as our induction hypothesis that all sets of n swans have the same colour, and consider a set of $n + 1$ swans. Fix some swan S_1 in this set. If we remove S_1, then we have n swans left, and by the induction hypothesis they all have the same colour. Let C be this colour.

Now consider another swan S_2 in the set. If we remove S_2 from our set of $n + 1$ swans instead of S_1, then we again have n swans left, and they all have the same colour by the induction hypothesis. Since S_1 is one of the swans in this set, it must have the same colour C as all the others. Hence, all $n + 1$ swans have the same colour.

It follows by the principle of mathematical induction that any set of n swans must always have the same colour.

4b **Theorem.** All positive integers are equal.

Proof. By induction over n.

For the base case, the positive integer 1 is equal to itself.

For the induction step, assume as our induction hypothesis that $n - 1 = n$. Adding 1 to both sides of this equality, we derive that $n = n + 1$.

It follows from this by the principle of mathematical induction that for all integers n the equality $n = n + 1$ holds. But then by transitivity we obtain that all integers are equal, and the claim in the theorem statement has thus been established.