Diskret Matematik og Formelle Sprog: Problem Set 2

Due: Wedneday February 23 at 12:59 CET.
Submission: Please submit your solutions via *Absalon* as 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. *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 the internet and/or used verbatim. 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 In this problem we wish to compare different sorting algorithms.

1a (40 p) The exercises in Chapter 2 of CLRS mention the *bubblesort* algorithm, which can be further optimized as follows:

```
OptimizedBubbleSort (A)
    i       := 1
    swapped := true
    while (i <= length(A) and swapped) {
        swapped := false
        for j := 1 upto size(A) - i {
            if (A[j] > A[j + 1]) {
                tmp      := A[j]
                A[j]     := A[j + 1]
                A[j + 1] := tmp
                swapped  := true
            }
        }
        i := i + 1
    }
```

Run the optimized bubblesort algorithm by hand on the array

$$A = [5, 2, 19, 7, 6, 12, 10, 17, 13, 14] \tag{1}$$

and show how the elements in the array are moved (similarly to what was done for insertion sort in class). Argue formally why this algorithm is guaranteed to always sort an array correctly. Analyse the time complexity of the algorithm.

Hint: Try to find a nice invariant for the inner while loop to help you argue correctness.

1b (30 p) Run merge sort by hand on the array in (1) (as in the notes for the lectures). Show in every step of the algorithm what recursive calls are made and how the results from these recursive calls are combined, and make sure to explain the final (and most interesting) merge step carefully. (Any clear way of explaining is fine—you do not have to learn how to draw pictures in LaTeX if you do not want to.)

1c (20 p) Suppose that we are given another array B of size n that is already sorted in increasing order. How fast do the merge sort and optimized bubblesort algorithms run in this case? Is any of them asymptotically faster than the other as the size of the array B grows?

1d (20 p) Suppose that we are given a third array C of size n that is sorted in *decreasing* order, so that it needs to be reversed to be sorted in the order that we prefer, namely increasing. How fast do the merge sort and optimized bubblesort algorithms run in this case? Is any of them asymptotically faster than the other as the size of the array C grows?

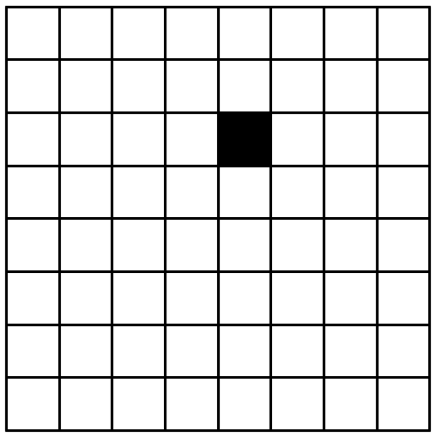
2 Last year DIKU celebrated its 50th anniversary with a lot of pomp, although slightly less emphasis was given to the fact that it was done one year late due to the pandemic. Even less publicity was given to the public outreach day organized in Fælledsparken for school children by the Algorithms and Complexity Section as part of the anniversary, for reasons that might become clearer after you have studied the problems below.

2a (30 p) In one of the events of the AC Section outreach day, Jakob had arranged so that 51 children[1] were given brightly coloured balls, and were positioned in a field in such a way that all the pairwise distances between the children were distinct. The children were then asked to identify which other child was closest to them and, at a given signal, to throw their ball to this child (and hopefully also receive an incoming ball from somewhere).

This turned out to be a public relations catastrophe. However the children were positioned as described above, every time at least one child ended up without a ball (but instead with tears in the eyes). This did not at all generate the goodwill DIKU was hoping for.

What went wrong? Was Jakob just immensely unlucky? Or can you prove mathematically that it was unavoidable that at least one child would end up without a ball? Would this had been different if Jakob had not insisted on 51 children, but had accepted the proposal by his colleagues to have 50 children? Or if not all distances would have had to be different?

[1] Well, because it was a 51st anniversary, after all.


(a) Punctured square grid.

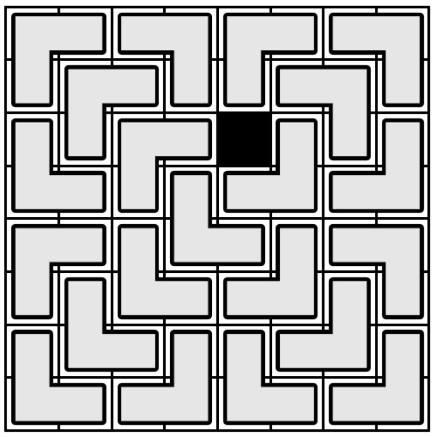

(b) Tiling of punctured square grid.

Figure 1: Tiling a punctured 2^n-by-2^n grid with L-shaped tiles (for $n = 3$).

2b (30 p) In another event, Jakob built a 5-kilometre car track across the park. On this track, 51 electric cars were placed at random locations (but all pointing in the same direction clockwise around the circuit). One car battery was sufficient for exactly one full lap if charged to 100% capacity. However, instead the batteries of all the cars were charged partially in such a way that the total charge of all the batteries together was sufficient for one car to travel exactly the full distance of 5 kilometres. After this, the batteries were distributed to the cars in some random way.

The children were given the challenge to start driving one car in such a way that one full lap of the track would covered. The rules were that if one car travelled far enough to bump into the rear of the car in front, then this next car could continue, and also the battery from the car behind could be shifted to the car in front so that the front car could use any remaining charge (and similarly for any other batteries picked up along the way). If, however, a car would run out of batteries before reaching the car in front, or before the full lap was completed by all the cars together, this was a failure.

This event went slightly better, in that the children were able to figure out a solution to the challenge most of the time. Jakob prided himself with that this was thanks to the fact that he had given the friendly advice, to avoid more embarrassment as in Problem 2a, that a good strategy was to start with the car with the most charge in its battery. Were the children just lucky this time, or can you prove that there is always a solution to this challenge? And is it a good idea to follow Jakob's advice, or does it seem more likely that the children figured out something smarter?

Example: If we for simplicity consider 3 cars on a perfectly circular track, with Car 1 placed at 12 o'clock with a 10% charge, Car 2 placed at 3 o'clock with a 60% charge, and Car 3 placed at 9 o'clock with a 30% charge, then starting with Car 2 is a winning strategy whereas starting with Car 1 or Car 3 leads to failure.

2c (30 p) In the final event of the day, a big 2^n-by-2^n grid was constructed, after which one cell in the grid was removed by placing a black square on it as illustrated in Figure 1a. The children were then given the task to cover all the other cells in the grid by placing L-shaped tiles in such a way that every cell was covered exactly once, and nothing outside of the grid was covered, as shown in Figure 1b.

By now the children were fairly fed up with these strange games, however, and Jakob's colleagues also started getting a bit annoyed, wondering if the strange shape of the tiles was somehow a not-so-subtle attempt to push for a competing foreign university instead, and the day did not end on a festive note at all. Disregarding this unfortunate turn of events, can you prove that it is actually true that for any 2^n-by-2^n grid, regardless of how it is punctured by removing a cell, it is always possible to tile the rest of the grid with L-shaped tiles?