KØBENHAVNS UNIVERSITET

Diskret Matematik og Formelle Sprog: Problem Set 2

Due: Wednesday February 24 at 23:59 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 in the course information 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 pseudo-code. 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 on the problems. Good luck!

1 Suppose that we are given an array $A = [5, 6, 4, 7, 3, 8, 2, 9, 1, 10]$ to be sorted in increasing order.

1a (30 p) Run insertion sort by hand on this array (as the lecturer has been doing in class), and describe in detail in every step of the algorithm how the elements in the array are moved.

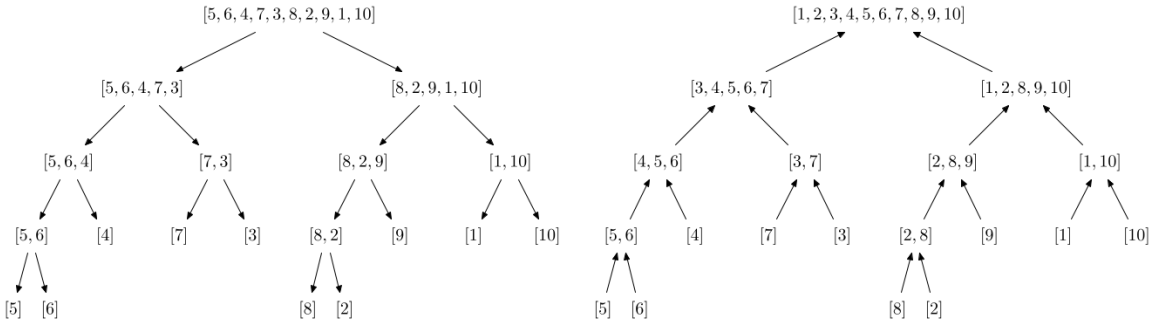
1b (30 p) Run merge sort by hand on this array (as the lecturer has been doing in class), and show in detail in every step of the algorithm what recursive calls are made and how the results from these recursive calls are combined.

Solution: See Figure 1 for an illustration of what happens in the algorithm.

We first recursively split the list in two part, where the first part is one element larger if the list size is odd, until all lists have sizes at most 1. This leads to the split lists in the leaves of the recursion tree in Figure 1a. (As a side note, when implementing merge sort it would make more sense to stop when lists are of sizes at most 2. These lists are either single elements, which are already sorted, or pairs of elements, that can be sorted by swapping places if needed. However, here we stick to the description given during the lecture just to avoid any unnecessary confusion.)

In the merge phase we work bottom up from the level above the leaves. Every parent node P takes its two children lists C_1 and C_2, which have previously been sorted, and merges them. We do so by placing to pointers e_1 and e_2 at the start of C_1 and C_2, respectively, and then adding the smallest of these elements to the list under construction after which the corresponding pointer is advanced.

For instance, at the bottom left we first merge [5] and [6] to a list $[5, 6]$. Consider now this list and its "sibling list" [4] to the right. We have $e_2 = 4 < 5 = e_1$, so 4 is added first, and since this



(a) Tree of recursive calls splitting list.

(b) Bottom-up sorting of split lists.

Figure 1: Merge sort on array $A = [5, 6, 4, 7, 3, 8, 2, 9, 1, 10]$.

empties list C_2 we then add 5 and 6 to get the list $[4, 5, 6]$. Moving further right, $[7]$ and $[3]$ are merged to a list $[3, 7]$, where the merging simply swaps the positions of the two elements. When $[4, 5, 6]$ is merged with $[3, 7]$, then 3 is the smallest element and goes first. After this, 4, 5, and 6 are all smaller than 7, which goes last. We get the list $[3, 4, 5, 6, 7]$. The recursive calls in the right subtree of the root are dealt with similarly (and this should be described in the solutions, although if you have explained clearly what is going on in the left subtree, then this description can be quite brief).

In the final step, we need to merge $[3, 4, 5, 6, 7]$ and $[1, 2, 8, 9, 10]$. Here we start with $e_2 = 1 < 3 = e_1$, so 1 goes first, updating e_2 to 2. Now we still have $e_2 = 2 < 3 = e_1$, so we add 2 to the list under construction, updating e_2 to 8. Since 8 is larger than all elements in C_2, this whole list will be added, after which 8, 9, and 10 will be added from list C_1. This produces the sorted list at the root of the tree in Figure 1b.

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

Solution: Merge sort will be the same, i.e., $O(n \log n)$. Insertion sort just verifies in linear time that the array is already sorted.

1d (20 p) Suppose that we are given a third array C 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 insertion sort and merge sort algorithms run in this case? Is any of them asymptotically faster than the other as the size of the array C grows?

Solution: Merge sort will be the same. Insertion sort will run in quadratic time.

2 Provide formal proofs of the following claims using mathematical induction.

2a (20 p) For all $K \in \mathbb{Z}^+$ it holds that $\sum_{s=1}^{K} s \cdot s! = (K+1)! - 1$.

Solution: We prove this equality by induction over K.

Base case: For $K = 1$ we have $\sum_{s=1}^{1} s \cdot s! = 1 \cdot 1! = (1+1)! - 1 = 1$.

Induction step: Assume that the equality holds for K and consider $K+1$. We have

$$\begin{aligned}\sum_{s=1}^{K+1} s \cdot s! &= \sum_{s=1}^{K} s \cdot s! + (K+1)(K+1)! \\ &= (K+1)! - 1 + (K+1)(K+1)! && \text{[by the induction hypothesis]} \\ &= (K+2)(K+1)! - 1 && \text{[rearranging terms]} \\ &= (K+2)! - 1\end{aligned}$$

which is the desired equality.

The claim now follows by the induction principle.

2b (20 p) For all $s \in \mathbb{N}$ and all $k > 0$ it holds that $1 + sk \leq (1+k)^s$.

Solution: First, we can note that the condition $k > 0$ in the problem statement is overly cautious—actually, $k > -1$ is enough, and we will see this in the proof below (although this requires some extra care in the argument which we do not need to worry about if we make k strictly positive). Second, since k could be any positive real number we cannot do inductive reasoning over k, but we can try to do induction over the integer s as follows.

Base case: For $s = 1$ we have $1 + k = (1+k)^1$ (and so the inequality $1 + k \leq (1+k)^1$ certainly also holds).

Induction step: Suppose that $1 + sk \leq (1+k)^s$ holds and consider $s+1$. We have

$$\begin{aligned}(1+k)^{s+1} &= (1+k)(1+k)^s \\ &\geq (1+k)(1+sk) && \text{[by the induction hypothesis (and since } 1+k > 0\text{)]} \\ &= 1 + sk + k + sk^2 && \text{[multiplying out]} \\ &= 1 + (s+1)k + sk^2 && \text{[rearranging terms]} \\ &\geq 1 + (s+1)k && \text{[since } sk^2 \geq 0 \text{ as long as } s \geq 0\text{]}\end{aligned}$$

which is the desired inequality.

The claim now follows by the induction principle.

Another way of establishing the inequality is to just do binomial expansion to compute

$$(1+k)^s = \binom{s}{0}1^s k^0 + \binom{s}{1}1^{s-1}k^1 + \ldots \binom{s}{s}1^0 k^s = 1 + sk + \cdots + k^s \geq 1 + ks$$

where the final inequality can be seen to hold since $k > 0$ (but note that the first proof presented above does not need this stronger assumption).

2c (30 p) For all $q \in \mathbb{N}$ it holds that $2^{4q+2} + 3^{q+2}$ is a multiple of 13.

Solution: This is very similar to a problem we did in class, and we will solve this one in the same way by arguing by induction over q.

Base case: For $q = 1$ we have $2^6 + 3^3 = 64 + 27 = 91 = 13 \cdot 7$.

Induction step: Suppose that $13 \mid 2^{4q+2} + 3^{q+2}$. Equivalently, this means that there is some integer N such that $2^{4q+2} + 3^{q+2} = 13N$. Fixing this N, and working on the expression for $q+1$, we can then write

$$\begin{aligned} 2^{4(q+1)+2} + 3^{(q+1)+2} &= 16 \cdot 2^{4q+2} + 3 \cdot 3^{q+2} \\ &= 3 \cdot \left(2^{4q+2} + 3^{q+2}\right) + 13 \cdot 2^{4q+2} && \text{[rearranging terms]} \\ &= 3 \cdot 13N + 13 \cdot 2^{4q+2} && \text{[by the induction hypothesis]} \\ &= 13 \cdot \left(3N + 2^{4q+2}\right) \end{aligned}$$

which shows that this expression is divisible by 13.

The claim follows by the induction principle.

2d (30 p) Let the sequence (T_i) be defined by

$$T_i = \begin{cases} 0 & \text{for } i = 1, \\ 1 & \text{for } i = 2, \\ T_{i-1} + T_{i-2} & \text{for } i \geq 3. \end{cases}$$

Prove that for all $i \geq 2$ it holds that

$$1 \leq \frac{T_{i+1}}{T_i} \leq 2 \, .$$

Solution: We prove this by induction over the numbers in the sequence, i.e., by induction over i. As a side note, the inequalities are actually sharp (i.e., $<$ rather than $\leq$) for $i \geq 4$, and this is easily seen from the proof that will follow below, but since the problem statement does not say anything about this there is no reason to comment on it in your solutions.

In order to make the proof clearer and avoid hidden assumption, we start by spelling out in full detail what induction hypotheses we will need.

Induction hypotheses: It holds for i that

1. $T_i > 0$ and $T_{i+1} > 0$;
2. $T_{i+1} \geq T_i$;
3. $T_i \leq T_{i+1} \leq 2 \cdot T_i$.

Note that we get the statement we need from this by dividing the inequalities in item 3 by T_i, which is in order since $T_i > 0$ by item 1.

Base case: For the base case $i = 2$ it is straightforward to verify for $T_2 = 1$ and $T_3 = T_2 + T_1 = 1$ that all of the induction hypotheses hold. (And yes, just writing like this in your exam solutions is perfectly in order, because you have said exactly what needs to be done, and the verification that what you say is true is fully mechanical and thus trivial.)

Induction step: Suppose that the induction hypotheses hold for i and consider $i+1$. Let us consider the hypotheses one by one.

1. Since T_i and T_{i+1} are both strictly positive by the induction hypotheses, it holds that $T_{i+2} = T_{i+1} + T_i$ is the sum of two strictly positive numbers and hence also strictly positive.

2. Since $T_i > 0$ by the induction hypotheses, we have $T_{i+2} = T_{i+1} + T_i > T_{i+1}$.
3. Since $T_{i+1} \geq T_i$ by the induction hypotheses, we can plug this into the recurrence relation to get $T_{i+2} = T_{i+1} + T_i \leq 2 \cdot T_{i+1}$. (And note that since $T_4 > T_3$ we could get a strict inequality here if we wanted to.)

The inequalities follow by the induction principle.