Algorithms and Data Structures I
May 6th, 2024

11. tutorial

1. Space complexity of Karatsuba. Prove that Karatsuba's multiplication algorithm works with linear space complexity (it requires space $\mathcal{O}(n)$). Recall that space complexity can be analyzed by consider the (worst-case of) root-leaf paths in the recursion tree – at any point in the run of the algorithm, only data related to some such path is in memory.

2. Recurrence 1. Solve the recurrence $T(n) = 2T(n/2) + \Theta(n \log n)$, $T(1) = 1$.

3. Recurrence 2. Solve the recurrence $T(n) = n^{1/2} \cdot T(n^{1/2}) + \Theta(n)$, $T(1) = 1$.

4. Towers of Hanoi. In the Hanoi Towers task we have 3 pillars A, B, C, and pillar A contains n disks ordered from biggest (bottom) to smallest (top). How to move all disks from pillar A to pillar B if we can only move one disk at a time and it is forbidden to place a bigger disk on a smaller disk? How many moves are necessary?

5. Pancake Sorting. Imagine we have a stack of disks which have numbers from 1 to n written on them, but they have been shuffled, so not necessarily in order. You want to sort them, but the only tool you are allowed to use is a pancake flipper, which you can insert between any two adjacent disks, and flip the stack above your flipper upside down, thus reversing its order.
Design an algorithm which sorts the stack of disks in $O(n)$ flips. (Note that the algorithm itself may not run in $O(n)$ time, but it should only make $O(n)$ flips.)

6. Twisted cable. We have a long cable with n wires on both ends. Each wire on the left end is connected to exactly one wire on the right end, and we want to find out which one. We can use the following operations to do that: (1) connect electricity to a given wire on the left end, (2) disconnect electricity from a given wire on the left end, and (3) measure electricity on a given wire on the right end. Design an algorithm which finds out which wire is connected to which, minimizing the number of these operations.

7. Twisted cable – nonadaptive version. Can you slightly change the algorithm above to work *non-adaptively* – that is, it's next step doesn't depend on it's previous step, i.e., it gathers some data in a way which doesn't depend on the input, and then reports its answer?