Tutorial 6

Data Structures 1, 28. 3. 2025 `https://iuuk.mff.cuni.cz/~chmel/2425/ds1en/`

Exercise 1 (Better fill factor)
Try to tweak the (a, b)-tree and its operations Insert, Delete so that all its nodes can be a little more full. In particular, we want to be able to build a $\left(\frac{2}{3}b, b\right)$-tree (for b a multiple of three).

Exercise 2 ((a, b)-trees on hard drives)
You set up an (a, b)-tree such that $b = d = 2a$, where d is the largest integer such that d items and $d+1$ pointers to children fit into a single block (note that this is intentionally off-by-one so that we do not have to worry about overflowing nodes).
What is the I/O complexity of the Insert, Delete, Find operations on (a, b)-trees? How many blocks does the data structure use?

Exercise 3 (Sorting without recursion)
Analyse the I/O-complexity of the implementation of a mergesort, where we have two arrays. In one of the arrays, we have sorted subsequences, and we merge them into the second array which will then have sorted subsequences with double the length.
(That is, the basis of the algorithm is an iterative for-loop instead of recursion.)
You may assume that $M \geq 3B$ – in particular, we have at least three blocks.

Exercise 4 (Programming II flashbacks)
Compute the I/O (and time) complexity of a k-way merge-sort. Compared with the previous algorithm, we always merge k subsequences at once and uses a k-element minimum heap for computing the minimum. We assume $M \geq (2k + 1) \cdot B$, so that we can fit both the heap and the currently read-through sequences into the "cache".

Exercise 5 (Matrix multiplication)
Analyse the I/O-complexity of the following algorithms for computing the matrix multiplication $C = A \cdot B$, where A, B are given square matrices $n \times n$.

a) Assume that A is stored in row-major order and B is stored in column-major order. Analyse the direct algorithm that follows from the definition.

b) Next, consider the recursive matrix multiplication algorithm. (The simplest one[1], where you split the matrices into quarters and multiply these quarters recursively and them sum the results together.) You may assume that the cache is tall: $M \in \Omega(B^2)$.

Bonus exercises

Exercise 6 (Finding the median)
Consider the following (Blum) algorithm for computing the median

1. Imagine that we split the array into $\lceil n/5 \rceil$ five-tuples.
2. In each of these five-tuples, compute its median.
3. Recursively compute the median of medians $\hat{M}$.
4. Split the elements into two sets depending on whether they are larger or smaller than $\hat{M}$.
5. Depending on the size of these sets, we recurse into the set with more elements.

Recall, that for the usual RAM model without memory hierarchy, the reccurent formula for the complexity is as follows: $T(n) = 7n/5 + T(n/5) + (n - 1) + T(7n/10) \in \mathcal{O}(n)$. Compute the I/O complexity of this algorithm, you may assume that $M \geq 3B$.
It might also be useful to know that the solution of the equation $\left(\frac{1}{5}\right)^c + \left(\frac{7}{10}\right)^c = 1$ is $c \approx 0.83978$.

[1]Though if you *really* want to, you can also do the Strassen algorithm.