Tutorial 7

Data Structures 1, 4. 4. 2025 https://iuuk.mff.cuni.cz/~chmel/2425/ds1en/

Exercise 1 (LRU vs OPT)
What is the largest ratio of cache misses of LRU and OPT strategies if their cache sizes are the same? (Assume that at the beginning, the cache is empty.)

Exercise 2 (Transposition with non-power-of-two matrices)
In the cache oblivious algorithm for matrix transposition, we have assumed that the matrix size is a power of two (that is, the matrix size is $2^n \times 2^n$). What would happen if this were not the case? Prove that for a matrix $n \times n$, all submatrices during the recursion are *almost square*, that is the numbers of rows and columns differ at most by one. Then, observe that the algorithm really works even for general matrix sizes.

Exercise 3 (Literally transpose-and-swap)
In the cache-oblivious algorithm, a common mistake is performing transpose-and-swap literally. That is, we first recursively transpose both matrices, and after that, we swap them. Analyse *time* complexity of this algorithm and then also the I/O complexity.

Exercise 4 (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 5 (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.