- This problemset has *four* questions. Please email your answers in a pdf file to `gmayank@mpi-inf.mpg.de`. The pdf filename should be "EDS-A8-<your_user_name>.pdf"
- The deadline is **15.07.2014** anywhere on Earth.

(25) 1. **Multi-dictionary problem** Let $A[1..n]$ be a fixed array of distinct integers. Given an array $X[1..k]$, we want to find the position (if any) of each integer $X[i]$ in the array A. In other words, we want to compute an array $I[1..k]$ where for each i, either $I[i] = 0$ (so zero means "none") or $A[I[i]] = X[i]$. Determine the exact complexity of this problem, as a function of n and k, in the binary decision tree model.

[Problem 1 in Jeff Erickson's notes on decision trees and leaf counting. `http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/28-lowerbounds.pdf`]

(25) 2. **Nuts and bolts** We are given n bolts and n nuts of different sizes, where each bolt exactly matches one nut. Our goal is to find the matching nut for each bolt. The nuts and bolts are too similar to compare directly; however, we can test whether any nut is too big, too small, or the same size as any bolt.

1. Prove that in the worst case, $O(n \log n)$ nut-bolt tests are required to correctly match up the nuts and bolts.
2. Now suppose we would be happy to find most of the matching pairs. Prove that in the worst case, $O(n \log n)$ nut-bolt tests are required even to find $n/2$ arbitrary matching nut-bolt pairs.

[Problem 3a-b in Jeff Erickson's notes on decision trees and leaf counting. `http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/28-lowerbounds.pdf`]

(25) 3. **Evasiveness** Let us assume we are given as input an undirected graph G with its binary adjacency martrix A. Thus $A[i, j] = 1$ if the edge (i, j) is present in the graph, and $A[i, j] = 0$ otherwise. A graph property is called evasive if an algorithm must look at all the $\binom{n}{2}$ entries in the matrix in order to decide if the graph has that property. Prove that checking whether a graph has a cycle or not(is acyclic) is an evasive property using an adversary argument.

[Problem 4 in Jeff Erickson's notes on adversary arguments. `http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/29-adversary.pdf`].

(25) 4. **Ternary tree labelling** Let T be a perfect ternary tree where every leaf has depth ℓ. Suppose each of the 3^ℓ leaves of T is labeled with a bit, either 0 or 1, and each internal node is labeled with a bit that agrees with the majority of its children. Prove that any deterministic algorithm that determines the label of the root must examine all 3^ℓ leaf bits in the worst case.

Problem 6a in Jeff Erickson's notes on adversary arguments.

`http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/29-adversary.pdf`