Motivation

We practise adversary arguments for comparison-based algorithms, probabilistic arguments, and applications of sorting.

I hope that all of you can do items 1 to 4 and the first part of item 5. The second part of item 5 (the lower bound) and item 6 are more demanding.

1. We have \( n \) items drawn from a linearly ordered set. Give an algorithm for finding the maximum. How many comparisons does it take? Prove that any comparison-based algorithm requires at least \( n - 1 \) comparisons.

2. There are three boxes. One of them contains a bar of gold. If you identify the correct box, the bar is yours. The protocol is as follows: You choose a box at random. Then one of the two other boxes is opened; it is guaranteed not to contain the bar. Now, you may or may not switch.

Should you switch?

3. Let \( A \) be an array of \( n \) elements. Design an algorithm that reorders the elements of \( A \) such that after execution of the algorithms all \( n! \) arrangements are equally likely. Your algorithm should use only \( O(1) \) extra space. You may use a function \( \text{random}(k) \) that returns a random integer in \([1..k]\). Argue the correctness of your algorithm.

The following program does not solve the problem as it does not generate all arrangements with equal probability. Prove this.

\[
\text{for } i := 1 \text{ to } n \text{ do}
\]
\[
j := \text{random}(n);
\]
\[
\text{swap}(A[j], A[i]);
\]

4. (A Scheduling Problem) A hotel manager has to process \( n \) advance bookings of rooms for the next season. His hotel has \( k \) identical rooms. Bookings contain an arrival date and a departure date. He wants to decide whether his rooms suffice to satisfy all bookings. Design an \( O(n \log n) \) algorithm for the problem.

5. (Finding Duplicates, Element Uniqueness): We work in the comparison model. You are given \( n \) items from a linearly ordered set. Decide whether the items are pairwise distinct or not. How many comparisons do you need? Can you prove a corresponding lower bound?
6. (Checking Equality of Multi-Sets): It is easy to check whether a sorting routine produces a sorted output. It is less easy to check whether the output is also a permutation of the input. But here is a fast and simple Monte Carlo algorithm for integers: (a) Show that \((e_1, \ldots, e_n)\) is a permutation of \((e'_1, \ldots, e'_n)\) iff the polynomial

\[
q(z) := \prod_{i=1}^{n}(z - e_i) - \prod_{i=1}^{n}(z - e'_i)
\]

is identically zero. Here, \(z\) is a variable. (b) For any \(\varepsilon > 0\), let \(p\) be a prime with \(p > \max\{n/\varepsilon, e_1, \ldots, e_n, e'_1, \ldots, e'_n\}\). Now the idea is to evaluate the above polynomial mod \(p\) for a random value \(z \in [0..p-1]\). Show that if \((e_1, \ldots, e_n)\) is not a permutation of \((e'_1, \ldots, e'_n)\), then the result of the evaluation is zero with probability at most \(\varepsilon\). Hint: a nonzero polynomial of degree \(n\) has at most \(n\) zeros.

Have fun with the solutions.