University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour, M. Fuchs, A. Malyusz

Algorithm Theory
Exercise Sheet 12

Due: Friday, 24th of January, 2025, 10:00 am

Exercise 1: Hidden numbers *(8 Points)*

(a) You are given a uniform random permutation of the numbers of $1, \ldots, n$. Prove that if we run the following algorithm

Algorithm 1 Finding the Maximum Element in a Permutation

Require: A uniform random permutation $A[1 \ldots n]$ of the numbers $1, \ldots, n$.
Ensure: The maximum element in A.

```
1: maxSoFar ← A[1]                      ▷ Initialize the maximum element as the first element
2: for i ← 2 to n do                    ▷ Iterate through the array starting from the second element
3:     if A[i] > maxSoFar then
4:         maxSoFar ← A[i]              ▷ Update the maximum element
5: return maxSoFar                      ▷ The maximum element in A
```

the maxSoFar value will, in *expectation*, be updated (line 4) at most H_n times where H_n is the n-th harmonic number defined by $H_n := \sum_{i=1}^{n} 1/i$. *(4 Points)*
Hint: Define

$$X_i := \begin{cases} 1 & \text{if } A[i] \text{ is larger than all values in the prefix } A[1,...,i-1] \\ 0 & \text{else} \end{cases}$$

and think about its expected value and how can you use it for this task?

There are n hidden integers a_i, each of them belonging to the range $[1, d]$. In a single query, you may choose two integers x and y ($1 \leq x \leq n$, $1 \leq y \leq d$) and ask the following question:

"Is $a_x \geq y$?"

Your goal is to determine the value of the largest element in the hidden array.

(b) Give a (deterministic) algorithm that finds the largest element in the array using $O(n \cdot \log_2 d)$ queries. *(1 Point)*

(c) In this task we want to improve the query complexity. Your objective is to modify the algorithm from b) such that, in expectation, at most $O\big(n + \ln n \cdot \log_2 d\big)$ queries are needed to find the maximum element. The algorithm itself should still be deterministic! *(3 Points)*
Hint: Use the result of task a) and the fact that $H_n \leq 1 + \ln n$.

Exercise 2: Randomized Coloring *(12 Points)*

Let $G = (V, E)$ be a simple, undirected graph with maximum degree Δ. A (node) coloring of the graph is an assignment of colors to the nodes in a way that no two adjacent nodes are assigned with

Algorithm 2 Randomized Coloring

Ensure: ϕ is a proper $\Delta + 1$ coloring

Let $L_v := \{1, 2, \ldots, \Delta + 1\}$
for each uncolored node $v \in V$ in parallel **do**
  v becomes active with probability $p = \frac{1}{2}$
  if v is active **then**
    Let v choose a color $x_v \in L_v$ uniformly at random
    if no neighbor u picked x_v as well **then**
      $\phi(v) := x_v$ ▷ v is colored now!
  if v is still uncolored **then**
    delete $\phi(u)$ from L_v for all colored neighbors u. ▷ Update L_v

the same color. More formal: A coloring is a mapping $\phi : V \rightarrow C$ of nodes in V to some color space C s.t. $\phi(u) \neq \phi(v)$ if $\{u, v\} \in E$.
Consider Algorithm 2 to assign colors from the colors pace $\{1, 2, \ldots, \Delta + 1\}$ to the nodes. Let L_v be the lists of **available** colors of v, that initially is set to the color space.
Note that in every iteration, $|L_v|$ is larger than the number of uncolored neighbors of v.

(a) Show that a node v that is still uncolored will be colored in the next iteration with probability at least 1/4. *(6 Points)*
Hint: Assume v is active and has k uncolored neighbors. What is the probability that v gets colored?

(b) After how many iterations is a node $v \in V$ colored in expectation? *(2 Points)*

(c) Show that Algorithm 2 terminates in $O(\log n)$ iterations **with high probability**.
That is for a given constant $c > 0$, all nodes are colored within $O(\log n)$ iterations with probability at least $1 - \frac{1}{n^c}$. *(4 Points)*
Hint: Use the result of a) for tasks b) and c) even if you didn't manage to come up with a solution.