Selected Topics in Algorithms
K. Mehlhorn
Exercise 8

Summer 2009
We will discuss this exercise sheet on July 20th and July 24th.

I will be about 15 minutes late for class on Monday, July 20th. The Informatics Department is considering the appointment of Benjamin Doerr to adjunct professor (Honorarprofessor); the appointment committee meets on Monday for a first meeting.

Please start discussing the exercises.

A van-de-Waerden-type theorem

Consider the following infinite graph. The nodes are the natural numbers and we have an edge (i, j) whenever $i < j$. The edges are colored with k colors.
Prove the existence of an infinite monochromatic path, i.e., show that there are nodes i_1, i_2, i_3, ... such that all edges $(i_\ell, i_{\ell+1})$, $\ell \geq 1$, have the same color.
Hint: Consider the case $k = 2$ first.

Monitoring Data Structures: Dictionaries

For this exercise a dictionary is a data structure that realizes the following behavior.
For any linearly ordered set U, it allows to maintain a subset S of U under the following operations.

- Make S the empty set.
- Membership: given $x \in U$, test whether $x \in S$. The answer is yes or no.
- Insert: given $x \in U$, replace S by $S \cup \{x\}$.
- Predecessor: given $x \in U$, return the largest $y \in S$ with $y \leq x$. If there is no such $y \in S$, return a special element $\perp$.

Mr. Evil provides you with an implementation of a dictionary. You are not sure whether you can trust Mr. Evil. How can you safeguard?
More precisely, can you write a wrapper for his implementation that will raise an alarm, if the implementation behaves incorrectly?

Monitoring Data Structures: Priority Queues

For this exercise a priority queue is a data structure that realizes the following behavior.
For any linearly ordered set U, it allows to maintain a subset S of U under the following operations.

- Make S the empty set.
- Insert: given $x \in U$, replace S by $S \cup \{x\}$.
- DeleteMin: Delete and return the smallest element of S.

Consider an execution trace, e.g.,
Insert(5); Insert(3); DelMin(3); Insert(7); Insert(2); DelMin(2); DelMin(5); Delmin(7);
where Insert(x) stands for the insertion of x and DelMin(y) stands for a DelMin operation with result y.

- Characterize correct traces.
- Describe an algorithm for recognizing correct traces. What is the running time of your algorithm?
- Assume that $U = \mathbb{N}$ and that each integer in $\{1, \ldots, n\}$ is inserted exactly once into S. Can you speed-up your algorithm for recognizing correct traces?