Parallel and Distributed Algorithms Winter 2009/2010	Blatt **2**

Ausgabe: 26.10.2006 **Abgabe: 2.11.2006**

Information

Solutions in english or german are fine.

2.1. Aufgabe (8)

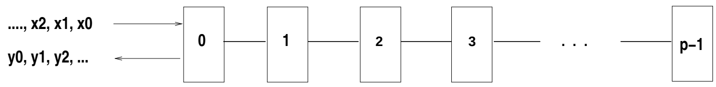
A *finite impulse response* (FIR) *filter* of order p receives a data stream $x_0, x_1, \ldots$ and outputs the data stream $y_0, y_1, \ldots$ with

$$y_t = \sum_{k=0}^{p-1} a_k \cdot x_{t-k}$$

for $t \geq p-1$. Hence FIR produces the output stream

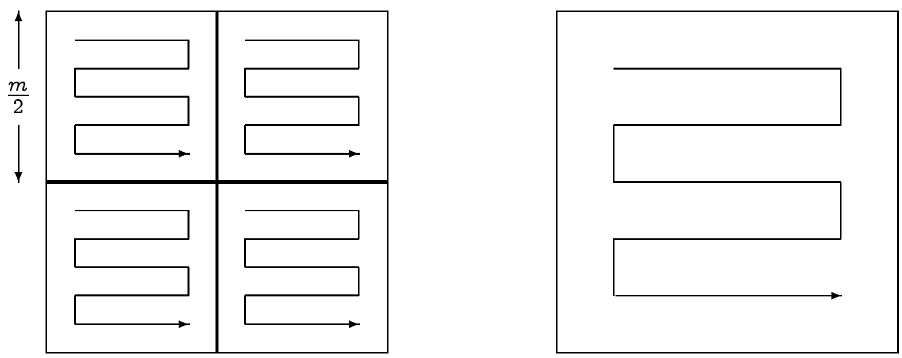
$$\begin{array}{lcllll} y_p & = & a_0 \cdot x_{p-1} & + \; a_1 \cdot x_{p-2} & + \cdots + & a_{p-1} \cdot x_0 \\ y_{p+1} & = & a_0 \cdot x_p & + \; a_1 \cdot x_{p-1} & + \cdots + & a_{p-1} \cdot x_1 \\ y_{p+2} & = & a_0 \cdot x_{p+1} & + \; a_1 \cdot x_p & + \cdots + & a_{p-1} \cdot x_2 \\ & \vdots & & & & \end{array}$$

Show how to construct a FIR filter with the help of a linear array with p processors so that x_t is input at step t and y_t is output at step at most $t + 2p$.



2.2. Aufgabe (8)

Based on the idea of Shear-Sort we define a recursive algorithm for sorting n keys on the two-dimensional mesh $M_{\sqrt{n}}$: we recursively sort the four quadrants of $M_{\sqrt{n}}$ in snakelike order. Then we sort the rows in snakelike order followed by sorting the columns. Finally we sort the "snake" with $2 \cdot \sqrt{n}$ steps of odd-even transposition sort.

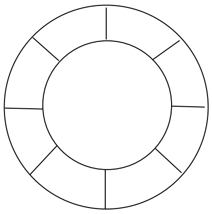


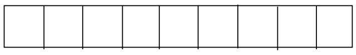
Let n be a power of two. Show that the recursive algorithm sorts n keys in $O(\sqrt{n})$ compute steps on the two-dimensional $\sqrt{n} \times \sqrt{n}$ mesh and verify that the algorithm sorts correctly.

Hint: 0-1-Principle.

2.3. Aufgabe (8)

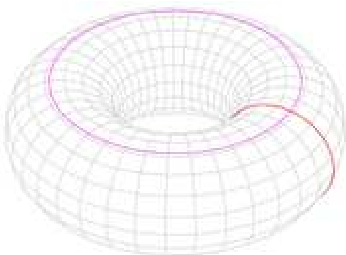
(a) The n-cell ring results from the n-cell linear array by identifying the first and the last cell.





Show that an n-cell linear array can simulate an n-cell ring with a slowdown of at most 2. (You may assume that n is even.) *Hint:* Assign cells of the ring in a clever way to cells of the array.

(b) The $n \times n$ torus results from the $n \times n$ two-dimensional mesh after identifying the first and last column as well as the first and last row.



Show that an $n \times n$ torus and an $n \times n$ mesh have equivalent computational power up to constant factors. In particular, show that an $n \times n$ mesh can simulate an $n \times n$ torus with a slowdown of at most two.