Aufgabe 1. Gegeben sei eine Datei, in der ausschließlich Zahlen stehen. In der ersten Zeile stehe eine natürliche Zahl, die angibt wie viele Zahlen noch folgen.

a) Schreibe ein Programm, dass diese Datei einliest, die Zahlen sortiert und die Datei mit der sortierten Liste überschreibt.

b) Modifiziere dein Programm nun so, dass in der ersten Zeile nicht mehr stehen muss, wie viele Zeilen noch folgen.

Aufgabe 2. Erstelle ein Programm `uniq`, dass zwei Dateinamen als Argumente erhält, die erste Datei wortweise[1] einliest und ohne doppelte Einträge in die zweite Datei schreibt (in irgendeiner Reihenfolge).

Aufgabe 3. Schreibe ein Programm, dass ein Labyrinth aus einer Datei einliest:

```
XXXXXXXXXXXXXXXX
X  X  XXXXXXXXXX*X
X$X  XX      XXX  X
X  X  XX  XXX  XXX  X
X     XX  XXX  XXX  X
XXX  X    XX  XXX  X
XXX    X          X
XXXXXXXXXXXXXXXX
```

Bemerkung: Wir spezifizieren das Labyrinth hier nicht viel näher, entscheide dich selbst vorher für ein Format. Soll die Größe des Labyrinths variabel sein oder fest? Soll die Größe in der ersten Zeile der Datei stehen oder nicht? Soll das Labyrinth quadratisch sein oder nicht? Soll es außen herum immer mit Xen begrenzt sein oder hast du vielleicht eine andere Lösung?

Das Programm soll einen Weg vom Startpunkt (dem Stern) zum Schatz (dem Dollarzeichen) finden. Die Xe sind Wände und Leerzeichen sind Pfade. Markiere einen Weg mit Punkten und gebe das Labyrinth mit Weg in der Konsole aus.

[1] Ein "Wort" ist eine Zeichenfolge, die keine Whitespaces enthät

```
XXXXXXXXXXXXXXXX
X  X  XXXXXXXXXX*X
X$X  XX      XXX.X
X.X  XX  XXX  XXX.X
X...XX  XXX  XXX.X
XXX.X...XX  XXX.X
XXX...X........X
XXXXXXXXXXXXXXXX
```

Aufgabe 4. Implementiere den Strassen-Algorithmus zur schnellen Matrixmultiplikation: Für $C = A \cdot B$ mit $A, B, C \in \mathbb{R}^{2^n \times 2^n}$ sei folgende Partitionierung gegeben:

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}; B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}; C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix};$$

wobei alle Untermatrizen gleich groß sind. Definiere 7 neue Matrizen:

$$\begin{aligned} M_1 &:= (A_{1,1} + A_{2,2}) \cdot (B_{1,1} + B_{2,2}) \\ M_2 &:= (A_{2,1} + A_{2,2}) \cdot B_{1,1} \\ M_3 &:= A_{1,1} \cdot (B_{1,2} - B_{2,2}) \\ M_4 &:= A_{2,2} \cdot (B_{2,1} - B_{1,1}) \\ M_5 &:= (A_{1,1} + A_{1,2}) \cdot B_{2,2} \\ M_6 &:= (A_{2,1} - A_{1,1}) \cdot (B_{1,1} + B_{1,2}) \\ M_7 &:= (A_{1,2} - A_{2,2}) \cdot (B_{2,1} + B_{2,2}) \end{aligned}$$

daraus ergibt sich dann für die Lösung C:

$$\begin{aligned} C_{1,1} &= M_1 + M_4 - M_5 + M_7 \\ C_{1,2} &= M_3 + M_5 \\ C_{2,1} &= M_2 + M_4 \\ C_{2,2} &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

Dabei soll zur Berechnung der M_i natürlich wieder der gleiche Algorithmus zur Matrixmultiplikation verwendet werden. Man spart insgesamt eine Matrixmultiplikation und verringert so den Aufwand von $O(n^3) = O(n^{\log_2(8)})$ auf $O(n^{\log_2(7)})$ also ungefähr $O(n^{2,807})$.