Fortgeschrittene funktionale Programmierung in Haskell

Übungszettel 3

Aufgabe 3.1:

Fünf Seemänner stranden auf einer Insel, deren einziger anderer Bewohner ein Affe ist. Um ihr Überleben zu sichern sammeln sie am ersten Tag n Kokosnüsse auf Vorrat.

In der Nacht wacht ein Seemann auf und befürchtet, am nächsten Tag nicht seinen fairen Anteil der Nüsse zu bekommen. Also teilt er den Haufen Kokosnüsse in fünf gleiche Teile auf. Dabei bleibt genau eine Kokosnuss übrig, die er dem Affen gibt. Der Seeman versteckt darauf hin sein Fünftel der Nüsse und tut alle verbleibenden wieder auf einen Haufen und legt sich schlafen.

Kurz darauf wacht der zweite Seemann auf und auch er fürchtet um seinen Anteil. Er teilt den (jetzt kleineren) Stapel ebenfalls in fünf gleiche Teile, abermals bleibt dabei eine Kokosnuss für den Affen übrig. Er versteckt seinen Teil, tut den Rest wieder auf einen Haufen und legt sich wieder schlafen.

Dieser Vorgang passiert mit allen fünf Seemännern. Jedes Mal wird der Stapel kleiner und jedes Mal bleibt eine Kokosnuss für den Affen. Am nächsten Tag wachen alle Seemänner auf und teilen den verbleibenden Haufen gerade durch fünf auf (dieses Mal bleibt nichts für den Affen). Alle merken, dass der Haufen kleiner ist am Vortag, allerdings sagt niemand etwas weil alle wissen, dass sie mitschuld sind.

Die Frage ist: Mit welcher Anzahl Kokosnüsse sind die Seemänner gestartet? Schreiben Sie ein `Haskell`-Programm, dass die (kleinste) Lösung dieses Rätsels berechnet. Verallgemeinern Sie danach, sodass ihr Programm auch den allgemeinen Fall von k Seemännern bearbeiten kann.

Aufgabe 3.2:

Stellen Sie sich vor, Sie sind im Dienst der Majestät für den britischen Geheimdienst `MI6` tätig, werden allerdings auf einer Mission gefangen genommen. Als Sie aufwachen, befinden sich zwischen einer Klippe (zwei Schritte hinter Ihnen) und einem Nest voller giftiger Schlangen mit sehr schlechtem Atem (zwei Schritte vor Ihnen).
Ihr Gegenspieler versichert Ihnen, dass er Sie freilassen wird, wenn sie eine Sequenz von 12 Instruktionen angeben können, die sie weder in das Schlangennest noch über die Klippe treten lässt (eine einzelne Instruktion ist entweder „ein Schritt nach vorne“ oder „ein Schritt nach hinten“). Sie werden jedoch gewarnt, dass sich ihr Gegenspieler, *nachdem* Sie die Sequenz angegeben haben, entscheiden kann, ob Sie die ganze Sequenz ausführen müssen oder nur jede zweite (Instruktionen $2, 4, 6, \dots$), dritte (Instruktionen $3, 6, \dots$), vierte, fünfte oder sechste (nur Instruktionen 6 und 12) Instruktion. Ihre Sequenz muss also auch für all diese Fälle geeignet sein.

`discrepancy.png`

Finden Sie eine gute Repräsentation des Problems in `Haskell` und schreiben Sie ein Programm, um herauszufinden, ob es eine Sequenz von 12 Instruktionen gibt, die Sie auf jeden Fall überleben lässt. Wie sieht es mit 11 oder 13 oder n Instruktionen aus?

Hinweis: Wer mehr über den mathematischen Hintergrund dieses Problems erfahren will, wird schnell unter dem Suchwort „Erdős discrepancy problem“ fündig.