Praktikum V – Zellularautomaten

GTI SoSe 2018 Profs. A. Siebert, S. Hauke

Aufgabe 1. Stochastische Matrizen

Alois Naturhansl verbringt seine Freizeit mit Radln, Klettern, Biergarting und Fensterln, und zwar nach festem Ritual:

- Nach dem Radln geht er in 20% der Fälle nochmals Radln, in 30% der Fälle Klettern und in 50% der Fälle Biergarting.
- Nach dem Klettern geht er in der Hälfte der Fälle Radln, sonst Biergarting.
- Nach dem Biergarting geht er immer Fensterln.
- Nach dem Fensterln geht er fast immer (zu 90%) Radln, sonst Klettern.

a. Über sein ganzes Leben betrachtet, wieviel Zeit verbringt Alois anteilig mit seinen vier Aktivitäten?

b. Wenn Alois gerade Fensterln war, wie groß ist dann die Wahrscheinlichkeit p, dass seine übernächste Aktivität Radln ist?

Aufgabe 2. 1-d Zellularautomaten

Implementieren Sie in Java 1-d Zellularautomaten.

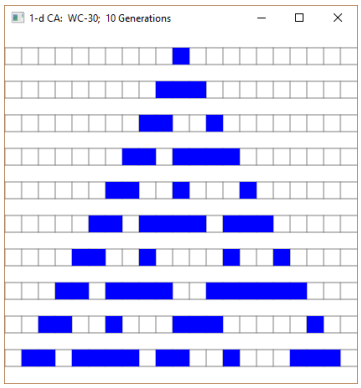
Eingabe für Ihr Programm sind der Wolfram Code `WC` des CA, $0 \leq WC \leq 255$, und die Anzahl `GENS` der Generationen, die der CA laufen soll.

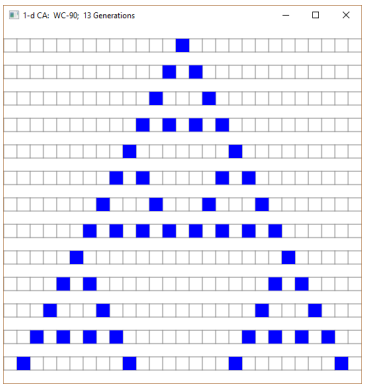
Die Ausgabe der vom CA generierten Konfigurationenfolge soll auf zwei Arten erfolgen: Einmal in ASCII und einmal graphisch mittels JavaFX.

Beispiele: WC-30 mit `GENS` = 10 Generationen; WC-90 mit `GENS` = 13.

```
 1     0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
 2     0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
 3     0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0
 4     0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0
 5     0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0
 6     0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0 0 0
 7     0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0
 8     0 0 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0
 9     0 0 0 1 1 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0
10     0 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0
```

```
 1      0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 2      0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 3      0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
 4      0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0
 5      0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 6      0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
 7      0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
 8      0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0
 9      0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
10      0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0
11      0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0
12      0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0
13      0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
```





Startkonfiguration ist jeweils eine einzige lebende Zelle.
Wenn Sie `GENS` Generationen erzeugen wollen, dann soll Ihr sichtbarer Zellraum die Größe 2·`GENS`+1 haben.

Randbehandlung: Der Zellraum eines CA ist theoretisch unendlich, aber Ihnen stehen nur endlich viele Zellen zur Verfügung. Am linken und rechten Ende Ihres Zellraums nehmen Sie deshalb an, dass es jeweils noch eine weitere (unsichtbare) Zelle gibt, die immer tot ist.

Als Rahmen steht Ihnen die Datei `CA.java` zur Verfügung. Der Code in `CA` zeichnet leere Zellen ähnlich der Graphik in Skript 02/Folie 177 mittels JavaFX.
Der Parameter `SF` (*scaling factor*) passt die Größe der Graphik an: Eine Zelle des CA soll die Ausgabegröße `SF`×`SF` haben.

Die Details von JavaFX sind hier belanglos. Sie dürfen sich gerne in die Semantik der einzelnen Befehle einarbeiten, aber genausogut können Sie den vorgegebenen Code als Blackbox betrachten, der Sie an geeigneter Stelle ein paar `fillRect()`-Methoden zum Einzeichnen der lebenden Zellen hinzufügen.

Experimentieren Sie mit unterschiedlichen Wolfram-Codes.