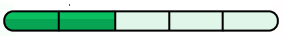
008 Problem Solving: Set

Maximilian Noppel[1]

[1]**max [at] vspace.one**

Zusammenfassung

We write a solver for the game 'Set'.

Schwierigkeit: 

EINFÜHRUNG

Wir schreiben einen Solver fuer das Spiel 'Set' (siehe `https://en.wikipedia.org/wiki/Set_(card_game)`). Dabei werden in jeder Runde n Karten praesentiert. Alle Spieler muessen gleichzeitig versuchen Triplets - also drei Karten, Reihenfolge egal - von Karten zu finden die bestimmte Regeln erfuellen. Was diese sind, dazu spaeter mehr. Die gefundenen Triplets werden entfernt und durch drei zufaellige neuen Karten (ohne zuruecklegen) ersetzt. Jedes Triplet gibt einen Punkt. Wer nach N gezeigten Karten die meisten Punkte hat, hat gewonnen. Das heisst am Ende bleiben n Karten liegen.

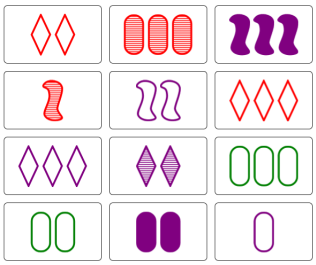


Figure 1. Some example 'Set' cards.

Regeln Nun zu den Regeln. Die Karten sehen aus wie in Fig. 1 dargestellt. Jede Karte hat 4 Eigenschaften:

- Anzahl: 1 bis 3
- Farbe: Rot, Gruen, Lila (eng. **r**ed, **g**reen, **p**urple)
- Fuellung: Keine, Gestrichelt, Voll (engl. **n**on-filled, **l**ine-filled, **f**illed)
- Form: 'Rundish', Raute, 'Komisch' (engl. **o**val, **d**iamond, **s**quiggle)

Die Regeln sind nun folgende; Man muss drei Karten finden, sodass die drei Karten fuer jede Eigenschaft (Anzahl, Farbe, Fuellung und Form) entweder drei verschiedenen Werte oder alle den gleichen Wert haben. Also Beispiel: Eine rote leere Raute, eine lila gestrichelte Raute und eine gruene ausgemalte Raute. Dieses Set wuerde also die Regeln erfuellen. Die Farbe ist immer verschieden, die Fuellung auch. Die Form und die Anzahl ist aber bei allen drei Karten gleich, naemlich eine Raute.

AUFGABE

Schauen wir uns also an wie die Eingabe in das Programm aussieht;

Eingabe

Das Programm wird mit einen String $s \in \{\{1,2,3\} \times \{\mathtt{r},\mathtt{g},\mathtt{p}\} \times \{\mathtt{n},\mathtt{l},\mathtt{f}\} \times \{\mathtt{o},\mathtt{d},\mathtt{s}\}\}^n$ als einzigem Parameter aufgerufen. Ein Beispiel: `1rno2gls...` bedeutet: One red non-filled oval and two green line-filled squiggles.

Ausgabe
Das Programm hat zwei Moeglichkeiten darauf zu reagieren: (1) Es gibt den Exitcode `1` um zu signalisieren dass das Kartendeck kein Triplet enthaelt. (2) Oder es gibt den Loesungsstring $s \in \{\{1,2,3\} \times \{r,g,p\} \times \{n,l,f\} \times \{o,d,s\}\}^3$ bestehend aus drei Karten auf stdout aus und beendet sich mit Exitcode `0`.

Notiz Es ist klar, dass der Loesungsstring nicht eindeutig ist, weil die drei Karten keine Reihenfolge haben. Das muss in den Unittest abgefangen werden. Just saying. ;) Ausserdem kann es natuerlich mehrere Triplets geben. Das Programm soll aber immer nur eins ausgeben! Wenn eins gefunden wurde kann die Suche also abgebrochen werden.

BONUS: WER IST SCHNELLER?

Wenn sich jemand berufen fuehlt ein kleines Framework aufzusetzen in dem wir die Bots gegeneinander antreten lassen koennen - nur zu! Evtl. muessen wir dafuer die Anzahl an Karten n (aktuell 12), die Anzahl an Eigenschaften (aktuell 4) und die Anzahl an Werten je Eigenschaft (aktuell 3) erhoehen um die Rechenzeit besser beurteilen zu koennen.

Anbieten wuerde sich hier vermutlich dass man Docker Container einreicht, die auf einem TCP Port Kartenstapel einlesen und darauf antworten. Dann kann man auch die Resourcen schoen beschraenken. Wir haben doch so Dockerprofis hier ;).