Kombinatorische Spieltheorie

Gewinnen mit Mathematik!
Workshop für TryScience

erkennen.
beweisen.
anwenden.

Dr. Friederike Stoll
Prof. Dr. Michael Eisermann
eiserm.de/popularisation

Universität Stuttgart
17. März 2023

Habe Mut, dich deines eigenen Verstandes zu bedienen!

Much to learn, you still have.
This is just the beginning.

Herzlich willkommen zu TryScience!

Mathematik ist Kunst und Handwerk, wunderschön und nützlich.
Mathematisch arbeiten bedeutet: erkennen. beweisen. anwenden.

Schüler:innen kommen mit Mathematik leider kaum noch in Kontakt; sie üben etwas Rechnen, aber Mathematik wird ihnen vorenthalten.

Mathematik ist nicht nur die sture *Anwendung* vorgefertigter Formeln, sondern auch und vor allem die *Entwicklung* neuer (Denk-)Werkzeuge. Mathematik (gr. μαθηματική τέχνη) ist die *Kunst des Erkennens/Lernens*. Sie ist ein schöpferisch-kreativer Prozess zum Lösen von Problemen.

Was zeichnet mathematische Arbeit aus? Ehrlich sein zu sich selbst und zu allen anderen, präzise formulieren, sorgfältig argumentieren, nachvollziehbar, nach logischen Regeln, alle Fälle berücksichtigen. Sorgfalt und Ehrlichkeit sind mühsam, doch die Mühe lohnt sich!

Was Sie einmal als richtig erkannt und sorgfältig nachgewiesen haben, behält seine Gültigkeit, auch nach Jahrhunderten, für immer und ewig! Andere Bereiche des Wissens sind vielleicht modischer, aber flüchtiger.

Was lernen Sie in unserem Mathe-Workshop?

Sie wollen Mathematik erleben? Dazu bieten Ihnen unsere Workshops die *mathematical experience*, ein intensiver Nachmittag zu spannenden Themen, ein Sprung ins Schwimmerbecken, anstrengend aber lohnend. Mathematik ist so viel mehr als Rechnen, lassen Sie sich begeistern!

Spielen macht Spaß und mobilisert all unsere mentalen Fähigkeiten: Sie wollen gewinnen? Dazu müssen Sie vorausschauen und planen! Genau dazu hilft uns Mathematik. Weil sie damit überall erfolgreich ist, wird sie ständig weiter entwickelt. Gewinnen Sie mit Mathematik!

Die Spieltheorie ist ein riesiges Gebiet. Wir bieten in diesem Workshop eine kurze Einführung, freundlich und gründlich, in die kombinatorische Spieltheorie. Wir präsentieren den berühmten Satz von Sprague–Grundy, mit dem Sie viele Spiele effizient lösen können. Abstraktion wirkt!

Sie programmieren gerne? Am liebsten in Verbindung mit Mathematik? Dann bietet Ihnen die Spieltheorie reichlich Inspiration und spannende Projekte. Zur Illustration habe ich kurze Python-Skripte eingefügt.

Wir feiern Rekursion und Induktion!

Bei der Suche nach optimalen Entscheidungen lernen Sie spielerisch Rekursion und Induktion, bei Spielen heißt dies oft *Rückwärtsinduktion*. Sie ist allgegenwärtig und nützlich und recht betrachtet sehr einfach: Wir beginnen mit den einfachsten Fällen und arbeiten uns dann empor.

So lösen Sie kombinatorische Spiele, indem Sie sie vom Ende her denken. So lösen Sie viele Optimierungsfragen, indem Sie schrittweise vorgehen. So lösen Sie komplizierte Aufgaben, indem Sie sie in einfache zerlegen. Kurzum: Rekursion und Induktion bilden ein universelles Werkzeug.

Erschreckende Schulrealität: Diese wunderbare, universelle und einfache Technik wird heutigen Schüler:innen nicht mehr zugetraut. Ein Fehler! Sie wird als ungeheuer schwierig diffamiert, als angeblich kompliziertes Spezialthema im Giftschrank versteckt, in den Vertiefungskurs verbannt.

Wer spielt und strategisch denkt, weiß es besser! In diesem Workshop zelebrieren wir die Rückwärtsinduktion zur Untersuchung von Spielen und entwickeln den Satz von Sprague–Grundy als effiziente Lösung.

Aufbau und Ablauf unserer Veranstaltung

Wir beginnen den TryScience-Workshop mit einer kurzen Vorstellung unserer Stuttgarter Mathematik-Studiengänge. Dabei klären wir auch allgemeine Fragen: Wo, woran und wie arbeiten Mathematiker:innen? Was macht das Mathestudium so wunderschön und so anspruchsvoll? Welche Voraussetzungen fordert und welche Fähigkeiten fördert das Mathestudium? Ausdauer? Kommunikation? Kreativität? Kopfrechnen?

Durch diese Präsentation können wir zwar einen guten Überblick geben, aber konkret vorstellen kann man sich ein Mathestudium dadurch noch nicht. Es gilt wie mit so vielem: Mathematik muss man selbst erleben! Dazu präsentieren wir ein sorgsam ausgewähltes Thema, diesmal die kombinatorische Spieltheorie, in Form einer Vorlesung und zugehöriger Übung, genau wie im richtigen Studium, nur etwas langsamer.

Wir haben uns ganz bewusst für dieses Format entschieden, damit Schülerinnen und Schüler einen möglichst realistischen Einblick in das Mathestudium bekommen und tatsächlich die Uni ausprobieren können. Das ist, wie eingangs gesagt, anstrengend aber lohnend.

Leseanleitung: Wie nutzen Sie diese Folien?

Vortrag und Skript haben verschiedene Ziele und ergänzen sich: Der Vortrag gibt einen Überblick, das Skript dient zur Vertiefung. Beide, Überblick und Vertiefung, sind in diesem Dokument verwoben. Nutzen Sie das gute Angebot, lernen Sie angeleitet, dann selbständig.

Die *Vortragsfolien* sind durch **blaue Titelbalken** leicht zu erkennen; dies kennzeichnet die Folien, die in der Vorlesung besprochen werden. Hier finden Sie alles Wesentliche, darauf sollten Sie sich konzentrieren beim ersten Durchgang. Manchen genügt bereits dieses Grundgerüst.

Die *Hintergrundfolien* mit **weißen Titelbalken** bieten Ihnen hilfreiche Erläuterung und Ausführung, Erinnerung und Ergänzung, zusätzliche Übungen mit Lösungen, weitere Beispiele und detaillierte Rechnungen, interessante Anwendungen und vieles mehr. Hier dosieren Sie selbst!

Diese Aufteilung folgt der Erfahrung, dass die Leserin und der Leser leichter eine vorhandene Übung, Erklärung oder Illustration übergehen können, als eine fehlende selbst (er)finden. Möge es beiden nützen!

Wozu dient mathematische Fachsprache?

Ich nutze in dieser Vorlesung neben Anschauung und Umgangssprache bewusst auch präzise Formulierungen in mathematischer Fachsprache. Das soll Sie nicht verwirren oder gar abschrecken. Ich nutze die Synopse gezielt, um Ihnen auch diesen Aspekt der Mathematik nahezubringen.

Jede Disziplin, insbesondere jede Wissenschaft, hat ihre Fachsprache. Das gilt für jedes Hobby, für jeden Sport und auch für jedes Studienfach. Das wird von Außenstehenden oft als „Fachchinesisch“ wahrgenommen, und tatsächlich kann es zur Ab- und Ausgrenzung missbraucht werden.

Fachsprache hat aber auch eine sehr nützliche und wichtige Funktion: Sie soll die Kommunikation über das Thema erleichtern, dabei möglichst flexibel und bequem sein und zugleich möglichst eindeutig und präzise. Das erfordert wie immer Gewöhnung und Übung, und es lohnt sich.

All das gilt besonders für die Mathematik als Sprache des systematischen logischen Denkens und Grundlage für Naturwissenschaft und Technik. Sie zu erlernen erfordert eine Investition und bringt reichen Ertrag.

Erleben Sie echte Mathematik!

Weite Teile der Wissenschaften – und die Mathematik bildet hier keine Ausnahme – erschließen sich erfahrungsgemäß nur durch ein längeres Studium und weisen jede flüchtige Besucher:in ab. Dennoch gibt es herausragende Einzelergebnisse, die sich zumindest in ihrer Aussage allgemein erläutern lassen. Das versuchen wir hier und noch mehr…

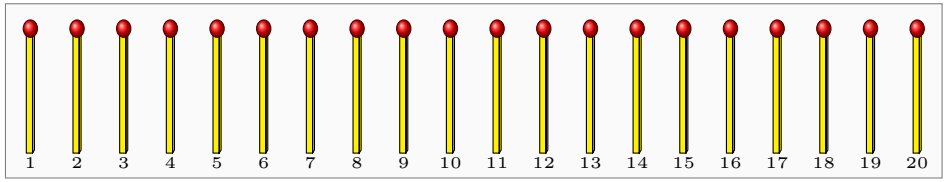
Wir haben das Thema so gewählt und zugeschnitten, dass Sie im Prinzip alles selbst nacharbeiten und mit der gegebenen Anleitung auch selbst beweisen können. Das ist Mathematik: erkennen, beweisen, anwenden. Diese Gewissheit und Autonomie sind einzigartig für die Mathematik! In der Schule wird Ihnen das vorenthalten, hier können Sie es erleben.

Die Spieltheorie bietet darüber hinaus einen weiteren großen Vorteil: Sie können Ihre neu erworbenen Erkenntnisse direkt anwenden und damit besser und effizienter spielen als zuvor. Mathematik hilft in nahezu allen Bereichen, hier können Sie es handfest und eigenständig erfahren.

Wir wünschen Ihnen viel Freude und viel Aha mit Mathematik!

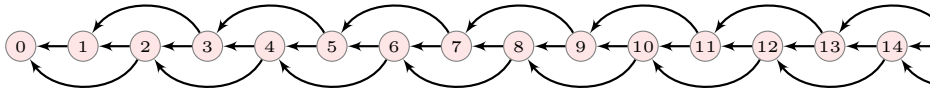
Nimm eins oder zwei!

Auf dem Tisch liegen anfangs $x \in \mathbb{N}$ Streichhölzer, Münzen, Steine, o.ä.



Beide Spieler ziehen abwechselnd, jeder entfernt ein oder zwei Hölzer.
Normalspiel / Misèrespiel: Wer nicht mehr ziehen kann, verliert / gewinnt.

Aufgabe: Übersetzen Sie die Spielregeln in einen Spielgraphen. **Lösung:**



🙂 Abstraktion ist die Kunst, Wichtiges von Unwichtigem zu trennen.
So können wir jedes Spiel knapp und übersichtlich als Graph codieren!
Er beantwortet die grundlegenden Fragen zum Spiel und seinen Regeln:
Welche Spielstände x gibt es? Welche Aktionen a sind in x möglich?

Nimm eins oder zwei!

Bevor Sie weiterlesen sollten Sie dieses Spiel einige Male durchspielen, am besten realistisch: gegeneinander! In der Vorlesung spielen Sie gegen mich an der Tafel... und langsam entwickeln Sie Ihre eigene Vermutung. Folgen Sie Ihrer natürlichen Neugier: Es macht Freude! So geht Lernen.

Beobachten Sie Ihren Lernprozess von *Whaaa?* über *Aha!* zu *Alles klar!* Anfangs werden Sie vermutlich wenig Struktur erkennen. Mit Erfahrung ahnen Sie gewisse Regelmäßigkeiten. Diese können Sie in den folgenden Aufgaben ausarbeiten und schließlich die allgemeine Regel formulieren.

Am Ende steht ein mathematischer Satz als Extrakt Ihrer Erfahrungen. Diesen können Sie induktiv beweisen und zukünftig getrost anwenden! Diesen mathematischen Reifeprozess *erkennen – beweisen – anwenden* möchte ich mit Ihnen hier zelebrieren, an mehreren Bei-Spielen.

Abstraktion vereinfacht: Der Spielgraph fasst alles wunderbar zusammen. Jede Ecke (Position) ist eine natürliche Zahl $x \in X := \mathbb{N} = \{0, 1, 2, 3, \dots\}$. Jede Kante (Zug) $a = (x, s) : x \to y$ erfüllt $y = x - s$ mit $s \in S = \{1, 2\}$. Dies ist ein einzeiliges Subtraktionsspiel mit Zugoptionen $S = \{1, 2\}$.

Nimm eins oder zwei! – Wer gewinnt und wie?

Aufgabe: Berechnen Sie für jede Position $x \in X$ Gewinn 1 oder Verlust 0.

- **Misèrespiel**, $\mu : X \to \{0, 1\}$: Wer nicht mehr ziehen kann, gewinnt.
- **Normalspiel**, $\nu : X \to \{0, 1\}$: Wer nicht mehr ziehen kann, verliert.

$$\nu(x) = \begin{cases} 0 & \text{falls } x \in \partial X, \text{ sonst} \\ 1 - \min\{\, \nu(y) \mid x \to y \,\}. \end{cases}$$

- Noch informativer als ν ist die **Sprague–Grundy–Funktion**:

$$\gamma : X \to \mathbb{N} : \gamma(x) = \operatorname{mex}\{\, \gamma(y) \mid x \to y \,\},$$
$$\operatorname{mex} : \{\, S \subsetneq \mathbb{N} \,\} \to \mathbb{N} : S \mapsto \min(\mathbb{N} \setminus S).$$

In Worten: Zu jeder Menge $S \subsetneq \mathbb{N}$ definieren wir $\operatorname{mex} S := \min(\mathbb{N} \setminus S)$ als das Minimum des Komplements [engl. *minimal excludant*, kurz *mex*]. also die kleinste natürliche Zahl, die nicht in der Menge S enthalten ist.

Beispiele: $\operatorname{mex}\{0, 1, 3, 5\} = 2$, $\operatorname{mex}\{0\} = 1$, $\operatorname{mex}\{1, 2, 3\} = 0$, $\operatorname{mex}\{\,\} = 0$.

Nimm eins oder zwei! – Wer gewinnt und wie?

Die Sprague–Grundy–Funktion γ erweist sich als Hauptakteurin dieser Theorie, ihre Bedeutung erschließt sich allerdings erst nach Satz 2B.

Vorrangig interessiert uns zunächst die Gewinnfunktion $\nu : X \to \{0, 1\}$. Sie sagt zu jeder Spielposition $x \in X$, ob wir in einer Verlustposition mit $\nu(x) = 0$ sind oder aber in einer Gewinnposition mit $\nu(x) = 1$. Genauer:
(0) Aus $x \in X$ mit $\nu(x) = 0$ führt jeder Zug $x \to y$ zu $\nu(y) = 1$.
(1) Aus $x \in X$ mit $\nu(x) = 1$ führt ein Zug $x \to y$ zu $\nu(y) = 0$.

Wir werden bald sehen, dass die Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N}$ noch nützlicher ist als ν. Satz 1D erklärt den genauen Zusammenhang:

$$\nu(x) = \begin{cases} 0 & \text{falls } \gamma(x) = 0, \\ 1 & \text{falls } \gamma(x) \geq 1. \end{cases}$$

Das Misèrespiel nenne ich hier zum Kontrast und als Variante zum Üben. Seine Theorie ist ganz anders, wesentlich mühsamer und weniger schön. In der Vorlesung gehe ich daher nicht genauer auf $\mu : X \to \{0, 1\}$ ein, doch hier in den Erläuterungen biete ich es Ihnen zur Ergänzung.

Nimm eins oder zwei! – naiv implementiert

Lösung: (0) Wir berechnen $\mu, \nu : X \to \{0,1\}$ rekursiv gemäß Definition:

```
def mu(x):
    if x == 0: return 1   # Wer nicht mehr ziehen kann, gewinnt.
    return 1 - min( mu(y) for y in range(x-2, x) if y >= 0 )
```

```
def nu(x):
    if x == 0: return 0   # Wer nicht mehr ziehen kann, verliert.
    return 1 - min( nu(y) for y in range(x-2, x) if y >= 0 )
```

Analyse: (a) Was ist schlecht an dieser Implementierung?
(b) Bestimmen Sie die Anzahl $f(x)$ der Funktionsaufrufe!

Antwort: (b) Wir finden $f(0) = 0$ und $f(1) = 1$ sowie für alle $x \in \mathbb{N}_{\geq 2}$ rekursiv $f(x) = f(x-1) + f(x-2)$. Dies ist die Fibonacci–Folge!

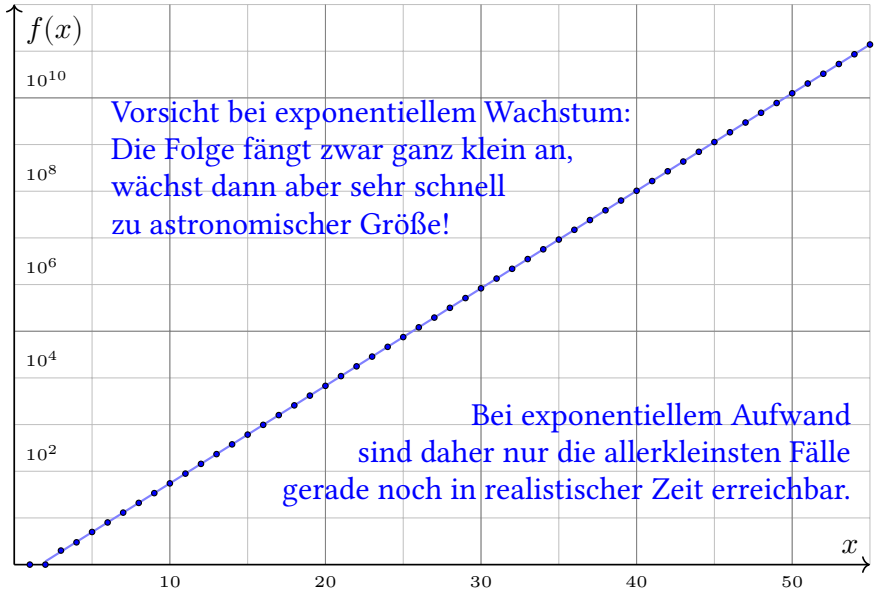
(a) Der Aufwand wächst exponentiell! Explizit gilt die Binet–Formel:

$$f(x) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^x - \left(\frac{1-\sqrt{5}}{2}\right)^x\right] \gtrapprox 0.447 \cdot 1.618^x$$

Übung: Beweisen Sie die Binet–Formel durch Induktion über $x \in \mathbb{N}$.

Nimm eins oder zwei! – naiv implementiert

Die Fibonacci–Folge in logarithmischer Darstellung:



Nimm eins oder zwei! – naiv implementiert

(0) Die Funktionen `min` und `max` bietet Python bereits standardmäßig, im Gegensatz dazu müssen wir die Funktion `mex` selbst programmieren:

```
def mex(theSet):          # mex = minimal excludant
    m = 0;
    while m in theSet: m += 1
    return m
```

Damit implementieren wir die Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N}$ ebenso leicht (und naiv) wie zuvor die Gewinnfunktion $\nu : X \to \{0,1\}$:

```
def gamma(x):             # Die Sprague-Grundy-Funktion
    return mex({ gamma(y) for y in range(x-2, x) if y >= 0 })
for x in range(0, 101):   # Probieren ergänzt Studieren!
    print( x, gamma(x) )  # Es gibt eine Überraschung...
```

Probe: Ist die Berechnung korrekt? Berechnen Sie kleine Beispiele, von Hand und mit Python, und vergleichen Sie die beiden Ergebnisse.

Aufwand: Ist die Berechnung schnell genug? Probieren Sie es selbst aus! Bis $x \approx 30$ geht es noch recht flott, dann nur noch quälend langsam.

Nimm eins oder zwei! – clever memoisiert

Wir merken uns, was wir berechnet haben, und recyclen es! Diese genial-einfache Idee heißt **Memoisation**, abgeleitet von lat. **Memorandum**, kurz **Memo**, *das zu Erinnernde*.

(1) In Python können wir Memoisation elegant wie folgt implementieren:

```
def memoize(f):             # Memoisation einer Funktion f
    memo = {}               # Liste aller vorigen Berechnungen
    def wrapper(x):         # Wir verpacken f in eine Hilfsfunktion.
        if x not in memo: memo[x] = f(x) # Berechnen und speichern
        return memo[x]      # Jeder Wert wird nur einmal berechnet.
    return wrapper          # Nur wrapper ist von außen sichtbar.
@memoize
def gamma(x):               # Die Sprague-Grundy-Funktion
    return mex({ gamma(y) for y in range(x-2, x) if y >= 0 })
```

☺ Naive Rekursion ist zwar korrekt, aber allzu verschwenderisch. Nachhaltige Buchführung reduziert den Rechenaufwand erheblich!

☺ In diesem Beispiel ist die Rechenzeit für γ nur noch linear in x. Probieren Sie es aus: vorher noch quälend langsam, nun blitzschnell!

Lösung: (1) Wir rechnen rekursiv, geschickt sortiert *bottom-up*:

$x=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mu=$	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1
$\nu=$	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1
$\gamma=$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2

Ebenso gut gelingt der rekursive Ansatz *top-down* mit Memoisation.

(2) Wie lautet die allgemeine Regel? Wir krönen unsere Bemühungen:

Satz 1A: einzeiliges Nim mit Zugoptionen $S = \{1, 2, \dots, n-1\}$, $n \geq 2$

Misèrespiel: Genau dann ist x eine Verlustposition, wenn $x \operatorname{rem} n = 1$.
Normalspiel: Genau dann ist x eine Verlustposition, wenn $x \operatorname{rem} n = 0$.
Die Sprague–Grundy–Funktion des Spiels ist $\gamma : \mathbb{N} \to \mathbb{N} : x \mapsto x \operatorname{rem} n$.

🙂 So erkennen und nutzen Sie Gewinnpositionen, wunderbar effizient.
Das ist Mathematik: (1) erkennen, (2) beweisen, (3) anwenden.

▶ Als mechanischer Computer erklärt von Matt Parker aka Stand-up Maths: *The Unbeatable Game from the 60s: Dr NIM.* youtu.be/9KABcmczPdg

Aufgabe: (2) Beweisen Sie diesen Satz per Induktion über $x \in \mathbb{N}$.

Lösung: (2a) Induktionsanfang: Zunächst gilt $\mu(0) = 1$ nach Misèreregel. Es folgt $\mu(1) = 0$, da nur der Zug $1 \to 0$ möglich ist; der Gegner gewinnt. Induktionsschritt: Sei $x \geq 2$ und die Aussage zu $\mu(y)$ gelte für alle $y < x$. Zu $x \operatorname{rem} n \neq 1$ existiert ein Zug $x \to y$ mit $x - y \in S$ und $y \operatorname{rem} n = 1$, also $\mu(y) = 0$; dieser Gewinnzug garantiert $\mu(x) = 1$. Von $x \operatorname{rem} n = 1$ führen alle Züge $x \to y$ mit $x - y \in S$ zu $y \operatorname{rem} n \neq 1$, also $\mu(y) = 1$.

(2b) Induktionsanfang: Zunächst gilt $\nu(0) = 0$ nach Normalspielregel. Induktionsschritt: Sei $x \geq 1$ und die Aussage zu $\nu(y)$ gelte für alle $y < x$. Zu $x \operatorname{rem} n \neq 0$ existiert ein Zug $x \to y$ mit $x - y \in S$ und $y \operatorname{rem} n = 0$, also $\nu(y) = 0$; dieser Gewinnzug garantiert $\nu(x) = 1$. Von $x \operatorname{rem} n = 0$ führen alle Züge $x \to y$ mit $x - y \in S$ zu $y \operatorname{rem} n \neq 0$, also $\nu(y) = 1$.

(2c) Sei $x \in \mathbb{N}$ und $r = x \operatorname{rem} n$. Wir zeigen $\gamma(x) = r$.
Induktionsvoraussetzung: Für alle $y < x$ gelte $\gamma(y) = y \operatorname{rem} n$.
Für $x < n$ gilt $r = x$ und wir finden $\gamma(x) = \operatorname{mex}\{0, \dots, r-1\} = r$.
Für $x \geq n$ gilt ebenso $\gamma(x) = \operatorname{mex}\{0, \dots, r-1, r+1, \dots, n-1\} = r$. QED

🙂 Die **Dynamische Programmierung** [*Dynamic Programming*, DP] ist ein Werkzeugkasten zur Optimierung rekursiver Berechnungen. Eine systematische Darstellung finden Sie in dem populären Lehrbuch 🕮 Cormen, Stein, Leiserson, Rivest: *Introduction to Algorithms*, §15.

Top-down mit Memoisation: Wir formulieren unsere Funktion rekursiv und speichern das Ergebnis jedes gelösten Teilproblems, etwa in einem assoziativen Array [*map*, *dictionary*] oder einer Tabelle [*hash table*].

Bottom-up topologisch sortiert: Jedes Teilproblem nutzt jeweils nur kleinere, bereits gelöste. Beide Formulierungen leisten meist dasselbe; Einsparungen entstehen, wenn top-down große Lücken überspringt.

🙂 Idealerweise können Algorithmen mit exponentiellem Aufwand so auf polynomiellen Aufwand reduziert werden, wie in unserem Beispiel: Unsere Lösung ist zunächst exponentiell in x, dann polynomiell in x, hier sogar linear in x, dank Satz 1A schließlich sogar nur logarithmisch in x. Das natürliche **Komplexitätsmaß** ist hier die Bitlänge $\operatorname{len}(x) \sim \log_2(x)$.

☹ **Rekursion** hat unter Anfänger:innen meist einen schlechten Ruf: Zuerst sind Denkweise und Progammiertechnik nicht leicht zu erlernen.

☹ Ist diese Hürde genommen, so folgt gleich die erste Ernüchterung: Naive Implementierung führt meist zu exponentiellem Aufwand.

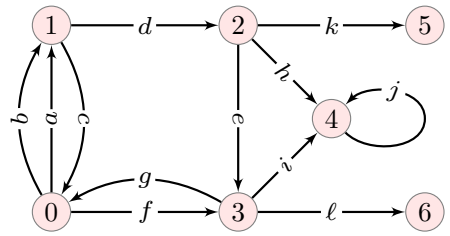
☹ Im Jobinterview machen Sie damit allein keinen guten Eindruck. „Never hire a developer who computes the factorial using recursion."

🙂 Rekursion entfaltet ihre wahre Kraft erst durch raffiniert-effiziente Implementierung: Die genial-einfache Idee hierzu heißt **Memoisation**, das geschickte Speichern der zuvor berechneten Zwischenergebnisse.

🙂 Im vorliegenden Falle vollendet Satz 1A unsere Lösung durch eine weitere dramatische Optimierung: Die Berechnung von $x \operatorname{rem} n$ benötigt nur noch logarithmischen Aufwand, gemäß Bitlänge $\operatorname{len}(x) \sim \log_2(x)$.

🙂 Sie ist zudem so einfach, dass wir sie im Kopf ausführen können! Das spüren Sie deutlich, wenn Sie dieses Spiel gegeneinander spielen. Beobachten Sie Ihren Lernprozess von *Whaaa?* über *Aha!* zu *Alles klar!*

Was genau ist ein Graph?



$$X = \{0, 1, \dots, 6\}$$
$$A = \{a, b, \dots, \ell\}$$
$$(\sigma, \tau) : a \mapsto (0, 1)$$
$$b \mapsto (0, 1)$$
$$\dots$$
$$\ell \mapsto (3, 6)$$

Definition 1B: gerichteter Multigraph $\Gamma = (\Gamma_0 \leftleftarrows \Gamma_1)$, kurz Graph

Ein **Graph** $\Gamma = (X, A, \sigma, \tau)$ besteht aus einer Eckenmenge $\Gamma_0 = X$ [*vertices*], einer Kantenmenge $\Gamma_1 = A$ [*edges*], mit $X \cap A = \emptyset$, sowie Randabbildungen $\sigma, \tau : A \to X$ [*boundary maps*], die jeder Kante $a \in A$ ihren Start $\sigma(a) \in X$ [*source*] und ihr Ziel $\tau(a) \in X$ [*target*] zuordnen.

Eine Kante $a \in A$ von $\sigma a = x$ nach $\tau a = y$ schreiben wir kurz $a : x \to y$ oder $x \xrightarrow{a} y$ und ebenso Wege $w = (x_0 \xrightarrow{a_0} x_1 \xrightarrow{a_1} \dots \longrightarrow x_n) \in \Gamma_n \subseteq \Gamma_*$. Der Graph Γ heißt **artinsch**, wenn er keine unendlichen Wege enthält, und **lokal-endlich** in $x \in X$, wenn die Menge $\{ a : x \to y \}$ endlich ist.

Was genau ist ein Graph?

Als Spiel Γ interpretieren wir jede Ecke $x \in X$ als möglichen **Zustand** oder **Position** und jede Kante $a \in A$ als mögliche **Aktion** oder **Zug**.

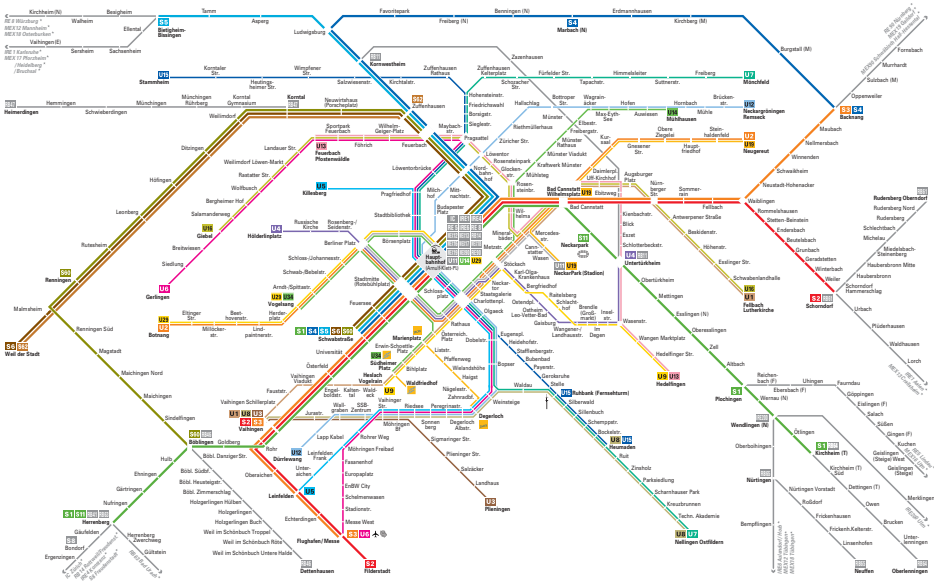
Zur Betonung nennen wir (X, A, σ, τ) auch **orientierten Multigraph**: Jede Kante $a : x \to y$ ist orientiert von ihrem Start x zu ihrem Ziel y. Zwischen je zwei Ecken $x, y \in X$ kann es mehrere Kanten geben. Wir erlauben Schleifen $a : x \to x$, also Kanten mit Start gleich Ziel.

Ecken heißen manchmal auch **Knoten** [*nodes*] oder **Punkte** [*points*], die Randabbildung $\partial = (\sigma, \tau) : A \to X \times X$ auch **Inzidenz** [*incidence*]. Der Graph heißt **einfach** [*simple*], wenn ∂ injektiv ist. Das heißt, von jedem Start $x \in X$ zu jedem Ziel $y \in X$ existiert höchstens eine Kante.

Im einfachen Graphen (X, A, σ, τ) können wir die Kantenmenge A durch ihr Bild $A' = \partial(A) \subseteq X \times X$ ersetzen und erhalten so den isomorphen Graphen (X, A', σ', τ') mit $\sigma' = \mathrm{pr}_1 : (x, y) \mapsto x$ und $\tau' = \mathrm{pr}_2 : (x, y) \mapsto y$.

Wir gelangen zu folgender Vereinfachung: Ein **einfacher Graph** (X, A) besteht aus einer Eckenmenge X und einer Kantenmenge $A \subseteq X \times X$; implizit mitgegeben sind die Projektionen $\sigma(x, y) = x$ und $\tau(x, y) = y$.

Alltägliche Beispiele für Graphen



Stuttgarter Verbundnetz, `www.vvs.de/karten-plaene/liniennetz`
Fahrgäste spielen „Wie komme ich am besten von A nach B?“

Einige Anwendungen für Graphen

Graphen haben zahlreiche Anwendungen in Mathematik und Informatik. Sie dienen häufig zur Modellierung, Abstraktion oder Vereinfachung, z.B. als Datenstruktur, Verkehrsliniennetz, Stammbaum, soziales Netz, etc.

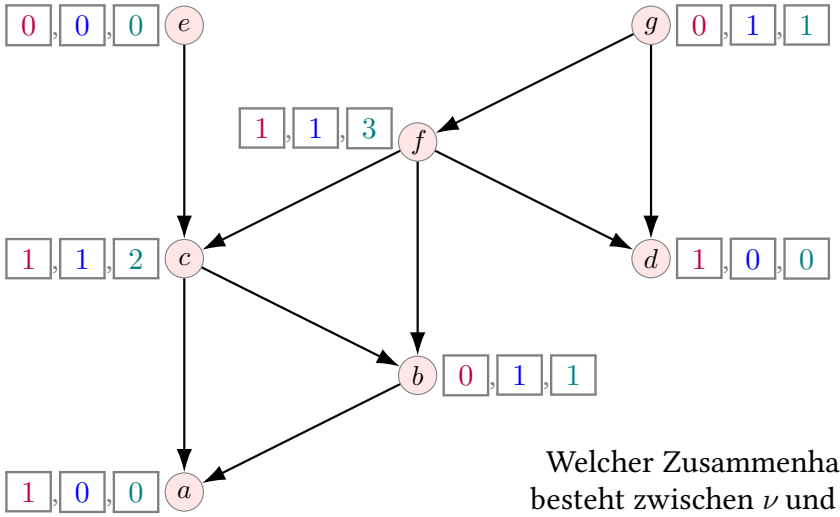
Wir formulieren daher eine möglichst umfassende allgemeine Definition. **Alternative Notationen** sind $G = (S, A, \sigma, \tau)$ oder $X = (X_0, X_1, \partial_0, \partial_1)$ oder $\Gamma = (V, E, s, t)$; das ist eine Frage von Traditionen und Vorlieben.

Google revolutionierte 1998 den Markt für Suchmaschinen, indem es das Internet ganz abstrakt als Graph aus Webseiten und Links analysierte: Der PageRank berechnet die Aufenthaltswkt $\sim$ Popularität $\sim$ Relevanz.

In der theoretischen Informatik besteht ein **Automat** $(S, A, \tau, \dots)$ aus **Zuständen** S und **Aktionen** A mit Übergangsfunktion $\tau : S \times A \to S$. Dies ist ein Graph mit der Kantenmenge $E = S \times A$, wobei $\sigma(s, a) = s$.

In der Algebra, speziell Darstellungstheorie, besteht ein **Köcher** [*quiver*] $Q = (Q_0, Q_1, \sigma, \tau)$ aus **Ecken** [*vertices*] und **Pfeilen** [*arrows*], und eine Darstellung $V : Q \to \mathtt{Vec}_{\mathbb{K}}$ ordnet jeder Ecke x einen $\mathbb{K}$–Vektorraum V_x zu und jeder Kante $a : x \to y$ eine $\mathbb{K}$–lineare Abbildung $V_a : V_x \to V_y$.

Aufgabe: Berechnen Sie $\mu, \nu, \gamma : X \to \mathbb{N}$ für das folgende Spiel G:



Welcher Zusammenhang besteht zwischen ν und γ?

☺ Das gelingt genauso für jeden lokal-endlichen artinschen Graphen G.

Jeden Graph $G = (X, A, \sigma, \tau)$ können wir als **Spiel** interpretieren. Zur einfachen Berechnung sei G zudem lokal-endlich und artinsch.

Wir interpretieren die Ecken $x \in X$ als **Zustände** des Spiels und die Kanten $a \in A$ als mögliche **Aktionen**. Wir unterscheiden dabei

$$\begin{aligned} &\text{aktive Zustände} && X^\circ = \{\, x \in X \mid \exists a : x \to y \,\}, \\ &\text{terminale Zustände} && \partial X = \{\, x \in X \mid \nexists a : x \to y \,\}. \end{aligned}$$

Für das **Misèrespiel** berechnen wir die Gewinnfunktion $\mu : X \to \{0, 1\}$: Für jeden terminalen Zustand $x \in \partial X$ gilt $\mu(x) = 1$: Wer nicht mehr ziehen kann, gewinnt. Für jeden aktiven Zustand $x \in X^\circ$ gilt rekursiv

$$\mu(x) = \begin{cases} 0 & \text{falls } \mu(y) = 1 \text{ für jeden Zug } a : x \to y, \\ 1 & \text{falls } \mu(y) = 0 \text{ für einen Zug } a : x \to y. \end{cases}$$

Im Zustand x sucht der ziehende Spieler einen Gewinnzug $a : x \to y$ mit $\mu(y) = 0$; damit übergibt er seinem Gegner eine Verlustposition y. Führt jeder Zug $x \to y$ zu $\mu(y) = 1$, so ist x eine Verlustposition.

Für das **Normalspiel** haben wir die Gewinnfunktion $\nu : X \to \{0, 1\}$ mit

$$\nu(x) = \begin{cases} 0 & \text{falls } \nu(y) = 1 \text{ für jeden Zug } a : x \to y, \\ 1 & \text{falls } \nu(y) = 0 \text{ für einen Zug } a : x \to y. \end{cases}$$

Insbesondere gilt $\nu(x) = 0$ für alle terminalen Zustände $x \in \partial X$. Logische Spitzfindigkeit: „für alle…“ ist wahr, wenn es keine gibt.

Die rekursive Logik ist dieselbe wie auf der vorigen Folie für μ erklärt:

Im Zustand x sucht der ziehende Spieler einen Gewinnzug $a : x \to y$ mit $\nu(y) = 0$; damit übergibt er seinem Gegner eine Verlustposition y. Ist ein Gewinnzug möglich, so ist x eine Gewinnposition, kurz $\nu(x) = 1$. Bei optimalem Spiel kann der ziehende Spieler seinen Gewinn erzwingen.

Führt hingegen jeder Zug $x \to y$ zu einer Gewinnposition, mit $\nu(y) = 1$, so ist x eine Verlustposition, kurz $\nu(x) = 0$. In diesem Falle kann also der ziehende Spieler aus eigener Kraft nicht gewinnen, sondern bestenfalls auf einen Fehler seines Gegners hoffen.

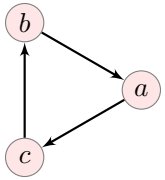
Noch informativer als ν ist die **Sprague–Grundy–Funktion**

$$\gamma : X \to \mathbb{N} : \gamma(x) = \operatorname{mex}\{\, \gamma(y) \mid x \to y \,\}.$$

Motivation und Nutzen von γ erschließen sich erst im weiteren Verlauf der Untersuchung. Es ist bei manchen (mathematischen) Objekten so, dass wir ihre hilfreichen Eigenschaften erst im Gebrauch entdecken.

Die rekursiven Formeln ergeben sich aus den Spielregeln. Bei kritischer Betrachtung stellt sich jedoch die Frage: Warum sind μ, ν, γ dadurch definiert? Können wir für jeden Zustand $x \in X$ garantieren, dass sich der Wert $\mu(x), \nu(x), \gamma(x)$ damit eindeutig berechnen lässt? Anders gefragt: Existiert zu diesen Rekursionsgleichungen eine Lösung? Ist sie eindeutig?

gar keine Lösung
$\mu, \nu, \gamma : X \to \mathbb{N}$



mehrere Lösungen
$\mu, \nu, \gamma : X \to \mathbb{N}$

Satz 1c: Existenz und Eindeutigkeit

Der Graph $G = (X, A, \sigma, \tau)$ sei lokal-endlich und artinsch. Zu den obigen Rekursionsgleichungen für μ, ν, γ existiert jeweils genau eine Lösung.

Beweis: Wir beweisen erst (a) die Eindeutigkeit und dann (b) die Existenz der wie oben rekursiv definierten Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N}$. Der Beweis für die Gewinnfunktionen $\mu, \nu : X \to \{0, 1\}$ verläuft genauso.

Eine **partielle Lösung** $\tilde{\gamma} : X_{\tilde{\gamma}} \to \mathbb{N}$ auf der Teilmenge $X_{\tilde{\gamma}} \subseteq X$ erfüllt: Für alle $x \in X_{\tilde{\gamma}}$ gilt $\{\, y \mid x \to y \,\} \subseteq X_{\tilde{\gamma}}$ und $\tilde{\gamma}(x) = \operatorname{mex}\{\, \tilde{\gamma}(y) \mid x \to y \,\}$.

(a) Eindeutigkeit: Je zwei partielle Lösungen $\tilde{\gamma} : X_{\tilde{\gamma}} \to \mathbb{N}$ und $\hat{\gamma} : X_{\hat{\gamma}} \to \mathbb{N}$ stimmen auf der Schnittmenge $Y = X_{\tilde{\gamma}} \cap X_{\hat{\gamma}}$ überein: Gäbe es $x_0 \in Y$ mit $\tilde{\gamma}(x_0) \neq \hat{\gamma}(x_0)$, so existierte ein Nachfolger $x_0 \to x_1$ mit $\tilde{\gamma}(x_1) \neq \hat{\gamma}(x_1)$. So fortfahrend erhalten wir einen unendlichen Weg $x_0 \to x_1 \to x_2 \to \dots$. Das widerspricht unserer Voraussetzung, dass der Graph G artinsch ist. Also kann es den Ausnahmezustand x_0 nicht geben.

(b) Existenz: Sei $\gamma := \bigcup \tilde{\gamma}$ die Vereinigung aller partiellen Lösungen $\tilde{\gamma}$. Dank (a) erhalten wir eine Funktion $\gamma : X_\gamma \to \mathbb{N}$ mit $X_\gamma = \bigcup_{\tilde{\gamma}} X_{\tilde{\gamma}}$ und $\gamma(x) = \tilde{\gamma}(x)$ falls $x \in X_{\tilde{\gamma}}$; Letzteres ist wohldefiniert dank Eindeutigkeit. Somit ist $\gamma : X_\gamma \to \mathbb{N}$ selbst eine partielle Lösung und zudem maximal.

Wir zeigen $X_\gamma = X$. Angenommen $x \in X$ und $\{\, y \mid x \to y \,\} \subseteq X_\gamma$. Wir können dann $\gamma(x) := \operatorname{mex}\{\, \tilde{\gamma}(y) \mid x \to y \,\}$ berechnen und γ somit fortsetzen, falls $x \notin X_\gamma$. Da γ jedoch maximal ist, gilt bereits $x \in X_\gamma$. Gäbe es ein $x_0 \in X \setminus X_\gamma$, so auch einen Nachfolger $x_0 \to x_1 \in X \setminus X_\gamma$. So fortfahrend erhalten wir einen unendlichen Weg $x_0 \to x_1 \to x_2 \to \dots$. Das widerspricht unserer Voraussetzung, dass der Graph G artinsch ist. Also kann es den Ausnahmezustand x_0 nicht geben.

Die Konstruktion (b) beweist, dass eine Funktion $\gamma : X \to \mathbb{N}$ existiert, die die Rekursion $\gamma(x) = \operatorname{mex}\{\, \gamma(y) \mid x \to y \,\}$ für alle $x \in X$ erfüllt. Je zwei solche Lösungen $\tilde{\gamma}, \hat{\gamma} : X \to \mathbb{N}$ sind gleich dank (a). Die Lösung γ ist also eindeutig. QED

Satz 1d: Aus γ folgt ν.

Vorgelegt sei ein lokal-endlicher artinscher Graph $G = (X, A, \sigma, \tau)$ mit $\nu : X \to \{0, 1\}$ und $\gamma : X \to \mathbb{N}$ wie oben. Für jeden Zustand $x \in X$ gilt

$$\nu(x) = \gamma(x) \wedge 1 := \begin{cases} 0 & \text{falls } \gamma(x) = 0, \\ 1 & \text{falls } \gamma(x) \geq 1. \end{cases}$$

Rückwärtsinduktion: Sei $x \in X$. Für alle Folgezustände y, mit $x \to y$, sei die Aussage bereits bewiesen. Wir unterscheiden zwei Fälle:

(0) *Verlustposition:* Führt jeder Zug $x \to y$ zu $\nu(y) = 1$, so gilt $\nu(x) = 0$. In diesem Falle gilt $\gamma(y) \geq 1$, also $\gamma(x) = 0$, somit $\nu(x) = \gamma(x) \wedge 1$.

(1) *Gewinnposition:* Führt ein Zug $x \to y$ zu $\nu(y) = 0$, so gilt $\nu(x) = 1$. In diesem Falle gilt $\gamma(y) = 0$, also $\gamma(x) \geq 1$, somit $\nu(x) = \gamma(x) \wedge 1$. QED

Induktionsanfang: Für jeden terminalen Zustand $x \in \partial X$ gilt (0).
Per Rückwärtsinduktion arbeiten wir uns dann schrittweise empor.

Aufgabe: Warum können wir diese Induktion auf G anwenden? Warum werden dadurch tatsächlich alle Zustände $x \in X$ erreicht?

Lösung: Wir wollen $\nu(x) = \gamma(x) \wedge 1$ für alle Zustände $x \in X$ zeigen.

Angenommen $\nu(x_0) \neq \gamma(x_0) \wedge 1$ für einen Ausnahmezustand $x_0 \in X$. Gilt $\nu(x_1) = \gamma(x_1) \wedge 1$ für alle Folgezustände x_1, mit $x_0 \to x_1$, so erhalten wir $\nu(x_0) = \gamma(x_0) \wedge 1$ dank (0,1). Also muss $\nu(x_1) \neq \gamma(x_1) \wedge 1$ gelten für mindestens einen Folgezustand x_1, mit $x_0 \to x_1$.

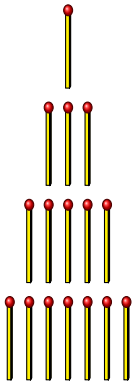
So fortfahrend erhalten wir einen unendlichen Weg $x_0 \to x_1 \to x_2 \to \dots$. Das widerspricht unserer Voraussetzung, dass der Graph G artinsch ist. Also kann es den Ausnahmezustand x_0 nicht geben. QED

⚠ Diese nützliche Beziehung gilt von γ zu ν, doch leider nicht zu μ:

Bemerkung 1e: Aus γ lässt sich μ nicht ablesen.

Im obigen Graphen gilt $\gamma(a) = \gamma(e) = 0$, aber $\mu(a) = 1$ und $\mu(e) = 0$. Dieses Beispiel zeigt: Aus γ lässt sich μ im Allgemeinen nicht ablesen.

Das Spiel Nim: Formalisierung als Graph



$x_4 = 1$

$x_3 = 3$

$x_2 = 5$

$x_1 = 7$

Zustand $x = (x_0, x_1, x_2, \ldots) \in \mathbb{N}^{(\mathbb{N})}$

Aktion $a = (x, r, s, y)$ mit $\begin{cases} (r,s) \in \mathbb{N}^2 \\ 1 \le s \le x_r \\ y = x - s\,e_r \end{cases}$

Zustand $y = (y_0, y_1, y_2, \ldots) \in \mathbb{N}^{(\mathbb{N})}$

Aufgabe: Formulieren Sie dieses Spiel in Worten und als Graph.

Lösung: Nim ist ein Spiel für zwei Spieler; beide ziehen abwechselnd. Die Zustandsmenge ist $X = \mathbb{N}^{(\mathbb{N})}$. Aktionen $x \to y$ sind Paare $(r,s) \in \mathbb{N}^2$ mit $1 \le s \le x_r$ und $y = x - s\,e_r$. Terminal ist $x = 0$, aktiv sind alle $x \neq 0$. Wir betrachten das Normalspiel: Wer nicht mehr ziehen kann, verliert.

Das Spiel Nim: Formalisierung als Graph

Dieses Spiel ist sehr einfach, doch wenn Sie es zum ersten Mal sehen, werden Sie anfangs vermutlich wenig oder keine Struktur erkennen. Erinnern Sie sich an Ihre Erfahrung mit einzeiligem Nim! (Satz 1A)

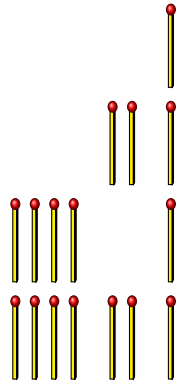
Zur Illustration spielen wir dieses Spiel ausgiebig in der Vorlesung, damit Sie das Problem und seine elegante Lösung wirklich spüren. In hoffe diese experimentelle Spielphase ist gut investierte Zeit.

Manche Spezialfälle entdecken und verstehen Sie leicht im Spiel: Zum Beispiel ist $(x_1, x_2) \in \mathbb{N}^2$ mit $x_1 = x_2$ eine Verlustposition, denn der zweite Spieler kann jeden Zug des ersten spiegeln.

Eine allgemeine Position $(x_1, x_2, \ldots, x_n) \in \mathbb{N}^n$ hingegen ist nicht so einfach zu durchschauen. Das müssen Sie selbst ausprobieren und persönlich erfahren! Dann können Sie die geniale Lösung würdigen.

Das Spiel Nim wurde gelöst von Charles L. Bouton: *Nim, a game with a complete mathematical theory*, Annals of Mathematics 3 (1901) 35–39. Das erklären wir in Satz 1F. Der Satz 2B von Sprague–Grundy überführt allgemein jedes neutrale kombinatorische Spiel in ein Nim-Spiel.

Das Spiel Nim: effiziente Lösung nach Charles Bouton 1901



$x_4 = 001_{\text{bin}}$

$x_3 = 011_{\text{bin}}$

$x_2 = 101_{\text{bin}}$

$x_1 = 111_{\text{bin}}$

Wir entwickeln x_i im Binärsystem

$$x_i = \sum_{j=0}^{m} \langle x_i \rangle_j \, 2^j \quad \text{mit} \quad \langle x_i \rangle_j \in \{0,1\}$$

und bilden die Spaltensumme

$$\langle s \rangle_j = \sum_{i=0}^{n} \langle x_i \rangle_j \operatorname{rem} 2.$$

Binäre Summe ohne Übertrag (XOR):

$$s = \sum_{j=0}^{m} \langle s \rangle_j \, 2^j =: x_0 \oplus x_1 \oplus \ldots \oplus x_n$$

Satz 1F: effiziente Lösung des Nim-Spiels, Bouton 1901

Im Normalspiel sind die Verlustpositionen x genau die Null-Positionen:

(0) Ist $x \in \mathbb{N}^{(\mathbb{N})}$ eine Null-Position, also $x_0 \oplus x_1 \oplus \ldots \oplus x_n = 0$, so führt jeder Zug $x \to y$ in eine Nicht-Null-Position, $y_0 \oplus y_1 \oplus \ldots \oplus y_n \neq 0$.

(1) Ist x eine Nicht-Null-Position, $x_0 \oplus x_1 \oplus \ldots \oplus \ldots x_n \neq 0$, so existiert ein Zug $x \to y$ in eine Null-Position, also $y_0 \oplus y_1 \oplus \ldots \oplus y_n = 0$.

Das Spiel Nim: effiziente Lösung nach Charles Bouton 1901

☺ Den Beweis führen wir allgemein für Satz 1G, noch besser Satz 2B. Die **binäre Summe ohne Übertrag** heißt auch **XOR** oder **Nim-Summe**. Dieser Algorithmus ist ebenso trickreich wie simpel: Er berechnet die Gewinnfunktion $\nu : X \to \{0,1\}$ des Normalspiels, schnell und einfach.

☺ Der Zeitaufwand ist nur linear in der Bitlänge der Eingabe $x \in \mathbb{N}^n$:

$$\begin{aligned} \text{len} &: \mathbb{N} \to \mathbb{N} \quad : x \mapsto \min\{\, \ell \in \mathbb{N} \mid a < 2^\ell \,\} \\ \text{len} &: \mathbb{N}^n \to \mathbb{N} : x \mapsto \text{len}(x_1) + \cdots + \text{len}(x_n) \end{aligned}$$

Linearer Aufwand ist das Beste, was wir hierzu je erhoffen können: Das Problem zu *lösen* ist kaum aufwändiger, als das Problem zu *lesen*!

☺ Boutons Lösung 1F enthüllt Gewinn- und Verlustpositionen. Der folgende Satz 1G zeigt zudem die Sprague–Grundy–Funktion und konstruiert explizit alle Gewinnzüge, also optimale Strategien!

Für das tatsächliche *Spielen* ist dies der entscheidende Schritt. Daher führe ich dies im folgenden Satz gebrauchsfertig aus.

Nim: starke Lösung nach Sprague 1935, Grundy 1939

Satz 1G: die Sprague–Grundy–Funktion des Nim-Spiels

Für Nim ist die Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N}$ gegeben durch

$$\gamma(x) = \bigoplus\nolimits_{i\in\mathbb{N}} x_i = x_0 \oplus x_1 \oplus x_2 \oplus \dots .$$

(0) Für jede Aktion $(r,s) : x \to y$ gilt $x_r \neq y_r$ und somit $\gamma(x) \neq \gamma(y)$.
(1) Zu $0 \leq n < \gamma(x)$ existiert eine Aktion $(r,s) : x \to y$ mit $\gamma(y) = n$.

$1 = 001_{\text{bin}}$	$1 = 001_{\text{bin}}$	$\to$ ~~$2 = 010_{\text{bin}}$~~	$\to$ ~~$3 = 011_{\text{bin}}$~~	$\to$ $0 = 000_{\text{bin}}$
$3 = 011_{\text{bin}}$	$3 = 011_{\text{bin}}$	$\to$ $0 = 000_{\text{bin}}$	$\to$ $1 = 001_{\text{bin}}$	$\to$ $2 = 010_{\text{bin}}$
$5 = 101_{\text{bin}}$	$6 = 110_{\text{bin}}$	$\to$ $5 = 101_{\text{bin}}$	$\to$ $4 = 100_{\text{bin}}$	$\to$ ~~$7 = 111_{\text{bin}}$~~
$7 = 111_{\text{bin}}$	$7 = 111_{\text{bin}}$	$\to$ $4 = 100_{\text{bin}}$	$\to$ $5 = 101_{\text{bin}}$	$\to$ $6 = 110_{\text{bin}}$
$0 = 000_{\text{bin}}$	$3 = 011_{\text{bin}}$	$\to$ $0 = 000_{\text{bin}}$	$\to$ $1 = 001_{\text{bin}}$	$\to$ $2 = 010_{\text{bin}}$

Satz 1G: formale Konstruktion

Sei $z := \gamma(x) \oplus n = 2^\ell + \sum_{k=0}^{\ell-1} z_k 2^k$. Wir wählen $r \in \mathbb{N}$ mit $\langle x_r \rangle_\ell = 1$.
Das garantiert $y_r := x_r \oplus z < x_r$ und ergibt $\gamma(y) = \gamma(x) \oplus z = n$.

Nim: starke Lösung nach Sprague 1935, Grundy 1939

Übung: Experimentieren Sie mit dem genialen Algorithmus des Satzes!
Wenden Sie das Verfahren mehrfach an, dann beweisen sie den Satz.

🙂 Sprague–Grundy 1G beinhaltet Boutons Lösung 1F dank Satz 1D:
Jede Position $x \in X$ mit $\gamma(x) = 0$ ist eine Verlustposition, also $\nu(x) = 0$.
Jede Position $x \in X$ mit $\gamma(x) \geq 1$ ist eine Gewinnposition, also $\nu(x) = 1$.

🙂 Wir werden diese geniale Methode in Satz 2B perfektionieren.
Der Beweis ist jeweils leicht und ich formuliere ihn detailliert aus.

☹ Wenn Sie zuerst den Beweis lesen, sagt er Ihnen noch allzu wenig.
Der Beweis ist einfach genial und genial einfach: Sie können ihn Schritt für Schritt leicht durcharbeiten, doch dabei besteht die Gefahr, dass der zündende Funke nicht überspringt. Das wäre hier besonders schade!

🙂 Zunächst empfehle ich, den Algorithmus in zahlreichen Bei-Spielen zu erproben. Durch eigenständiges Probieren wird Ihnen schnell klar, wie das Verfahren funktioniert und wie ein Beweis aussehen könnte.
So können Sie selbst den Beweis entdecken, entwickeln und verstehen.

Nim: starke Lösung nach Sprague 1935, Grundy 1939

Beweis: Aussage (0) ist klar, da $(\mathbb{N}, \oplus)$ eine Gruppe ist, sogar abelsch.
Abstrakt gesehen nutzen wir nur die Kürzbarkeit der Verknüpfung $\oplus$.

Anschaulich: Jeder Zug $(r,s) : x \to y$ ändert nur die Koordinate x_r zu y_r.
Im Nim-Spiel gilt $y_r < x_r$, wir benötigen nur $y_r \neq x_r$. In der Nim-Summe $\gamma(x) = x_1 \oplus \dots \oplus x_n$ ändert sich mindestens ein Bit, also $\gamma(y) \neq \gamma(x)$.

(1) Vom Start $m = \gamma(x)$ zum gewünschten Ziel $n < m$ steht das höchste zu ändernde Bit an der Stelle ℓ. Dank $m > n$ gilt $\langle m \rangle_\ell = 1$ und $\langle n \rangle_\ell = 0$.
Demnach existiert mindestens eine Koordinate $r \in \mathbb{N}$ mit $\langle x_r \rangle_\ell = 1$.
Genauer existiert eine ungerade Anzahl dieser Wahlmöglichkeiten.

Diese Wahl r mit $\langle x_r \rangle_\ell = 1$ stellt sicher, dass $y_r := x_r \oplus z < x_r$ gilt.
(Im Falle $\langle x_r \rangle_\ell = 0$ wäre $x_r \oplus z > x_r$ im Nim-Spiel kein erlaubter Zug.)
Diese Wahl r und $s = x_r - y_r \geq 1$ definieren die Aktion $(r,s) : x \to y$ mit $\gamma(y) = \bigoplus_{i\in\mathbb{N}} y_i = (\bigoplus_{i\in\mathbb{N}} x_i) \oplus z = \gamma(x) \oplus z = n$, wie gewünscht.

Die beiden Eigenschaften (0–1) garantieren, dass die Funktion $\gamma : X \to \mathbb{N}$ tatsächlich die Sprague–Grundy–Funktion des Nim-Spiels G ist. QED

Nim: starke Lösung nach Sprague 1935, Grundy 1939

Aufgabe: Vorgelegt sei eine Position $x \in X = \mathbb{N}^{(\mathbb{N})}$ im Nim-Spiel.
(1) Wie finden Sie von x aus *alle* möglichen Gewinnzüge $(r,s) : x \to y$?
(2) Wie finden Sie zu $0 \leq n < \gamma(x)$ *alle* Züge $(r,s) : x \to y$ mit $\gamma(y) = n$?
Gehen Sie den Beweis sorgsam durch und zeigen Sie folgenden Zusatz:

Satz 1G: Vollständigkeit

Die genannte Konstruktion liefert *alle* Züge $x \to y$ mit $\gamma(y) < \gamma(x)$.

Lösung: Die Wahl einer Zeile $r \in \mathbb{N}$ mit $\langle x_r \rangle_\ell = 1$ ist nicht nur hinreichend, sondern auch notwendig für $y_r := x_r \oplus z < x_r$.

Satz 1G und sein Beweis beruhen auf der folgenden Beobachtung:

Lemma 1H: Umformulierung der mex–Eigenschaft

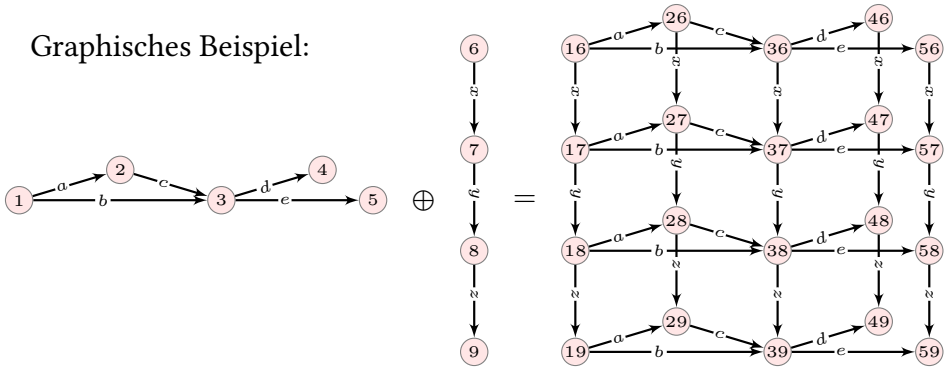
Vorgelegt sei ein Graph $G = (X, A, \sigma, \tau)$ mit einer Funktion $\gamma : X \to \mathbb{N}$.
Dann sind (0,1) äquivalent zu $\gamma(x) = \operatorname{mex}\{\, \gamma(y) \mid x \to y \,\}$ für alle $x \in X$.

Übung: Erklären Sie sorgfältig „⇒“ und „⇐“ zur Wiederholung.

Summe von Spielen

Wir spielen mehrere Spiele G_i parallel, also gleichzeitig nebeneinander. Der ziehende Spieler darf sich aussuchen, in welchem Spiel G_i er zieht. Dies fassen wir zusammen zu einem gemeinsamen Spiel $G = \bigoplus_{i\in I} G_i$.

Graphisches Beispiel:



Graphische Illustration der Summe $G = G_1 \oplus G_2$: Ein Zug in G ist entweder ein Zug in G_1 (horizontal) oder ein Zug in G_2 (vertikal).
Aufgabe: Formulieren Sie explizit die Definition von $G = \bigoplus_{i\in I} G_i$.

Summe von Spielen

⚠ Es gibt mehrere Möglichkeiten, Produkte von Graphen zu definieren. Gegen mögliche Verwirrung hilft wie immer nur eine präzise Definition:

Definition 2A: Produkt und Summe von Spielen

Gegeben sei eine Familie $(G_i)_{i\in I}$ von Graphen $G_i = (X_i, A_i, \sigma_i, \tau_i)$.
Ihr **Produkt** ist der Graph $P = \prod_{i\in I} G_i = (X, A, \sigma, \tau)$ mit

$$X = \prod_{i\in I} X_i = \{\, x : I \to \bigcup_{i\in I} X_i : i \mapsto x_i \mid x_i \in X_i \,\},$$
$$A = \{\, (x, i, a_i, y) \mid x, y \in X,\ i \in I,\ a_i : x_i \to y_i,\ \forall j \neq i : x_j = y_j \,\}$$

sowie den Projektionen $\sigma(x, i, a_i, y) = x$ und $\tau(x, i, a_i, y) = y$.
Die **Summe** $S = \bigoplus_{i\in I} G_i = (X', A', \sigma', \tau')$ ist der volle Teilgraph $S \leq P$ der Zustände $x \in X$ mit endlichem Träger $\operatorname{supp}(x) := \{\, i \in I \mid x_i \in X_i^\circ \,\}$.

Der Träger $\operatorname{supp}(x) \subseteq I$ indiziert aktive Koordinaten $i \in I$ mit $x_i \in X_i^\circ$. Ist I endlich, so ist das Produkt gleich der Summe, doch in unendlichen Summen verlangen wir, dass nur endlich viele Koordinaten aktiv sind.

Erinnerung: Produkt und Summe von Gruppen

Zu jedem Index $i \in I$ sei $G_i = (X_i, +_i, 0_i, -_i)$ eine abelsche Gruppe mit Grundmenge X_i und darauf der Addition $+_i : X_i \times X_i \to X_i$ mit neutralem Element $0_i \in X_i$ und Negation $-_i : X_i \to X_i$.

Das **Produkt** $P = \prod_{i\in I} G_i := (X, +, 0, -)$ hat als Grundmenge

$$X = \prod_{i\in I} X_i = \{\, x : I \to \bigcup_{i\in I} X_i : i \mapsto x_i \mid x_i \in X_i \,\}.$$

Wir definieren koordinatenweise die Addition $x + y = (x_i +_i y_i)_{i\in I}$, das neutrale Element $0 = (0_i)_{i\in I}$ und die Negation $-x = (-_i x_i)_{i\in I}$.

Wir definieren den **Träger** $\operatorname{supp} : P \to \mathfrak{P}I : x \mapsto \{\, i \in I \mid x_i \neq 0_i \,\}$. Einschränkung definiert die **Summe** der Gruppen $(G_i)_{i\in I}$ vermöge

$$S = \bigoplus_{i\in I} G_i := \{\, x \in P \mid \operatorname{supp}(x) \text{ endlich} \,\}.$$

Übung: Mit dieser Konstruktion ist $(P, +, 0, -)$ eine abelsche Gruppe und $S \leq P$ eine Untergruppe. Dasselbe gilt für Ringe; für $\sharp I = \infty$ hat das Einselement $1 = (1_i)_{i\in I}$ keinen endlichen Träger, also $1 \in P$, aber $1 \notin S$.

Erinnerung: Produkt und Summe von Monoiden

Allgemeiner gelingt diese Konstruktion für abelsche Monoide $(X_i, +_i, 0_i)$, noch allgemeiner Tripel aus einer Menge X_i mit innerer Verknüpfung $+_i : X_i \times X_i \to X_i$ und idempotenten Element $0_i \in X_i$, also $0_i +_i 0_i = 0_i$.

Im Falle $G_i = G$ für alle $i \in I$ erhalten wir die (eingeschränkte) Potenz:

$$P = G^I := \{\, x : I \to G \,\}$$
$$S = G^{(I)} := \{\, x : I \to G \mid \operatorname{supp}(x) \text{ endlich} \,\}$$

Nur auf letzterem haben wir die Summenabbildung

$$\sum : G^{(I)} \to G : (g_i)_{i\in I} \mapsto \sum_{i\in I} g_i.$$

Aus dem abelschen Monoid $(\mathbb{N}, +, 0)$ erhalten wir das Monoid $\mathbb{N}^{(\mathbb{N})}$ mit koordinatenweiser Addition $+$ und hierauf $\sum : \mathbb{N}^{(\mathbb{N})} \to \mathbb{N}$.

Beispiel: Für den Sprague–Grundy–Satz nutzen wir insbesondere: Aus der abelschen Gruppe $(\mathbb{N}, \oplus, 0)$ erhalten wir die Gruppe $\mathbb{N}^{(\mathbb{N})}$ mit koordinatenweiser Addition $\oplus$ und hierauf $\bigoplus : \mathbb{N}^{(\mathbb{N})} \to \mathbb{N}$.

Satz 2B: Sprague 1935, Grundy 1939

Gegeben seien lokal-endliche artinsche Graphen $G_i = (X_i, A_i, \sigma_i, \tau_i)$. Dann ist auch ihre Summe $G = \bigoplus_{i\in I} G_i$ lokal-endlich und artinsch, und für ihre Sprague–Grundy–Funktion gilt $\gamma(x) = \bigoplus_{i\in I} \gamma_i(x_i)$.

Genauer: (0) Für jede Aktion $(i, a_i) : x \to y$ in G gilt $a_i : x_i \to y_i$ in G_i. Für die Grundy–Zahlen folgt $\gamma_i(x_i) \neq \gamma_i(y_i)$ und somit $\gamma(x) \neq \gamma(y)$.

(1) Zu $0 \leq n < \gamma(x)$ existiert eine Aktion $(i, a_i) : x \to y$ mit $\gamma(y) = n$. Insbesondere finden wir so Gewinnzüge $x \to y$ mit $\gamma(y) = 0$.

Den Satz beweisen wir durch die Konstruktion solcher Züge:

Satz 2B: formale Konstruktion

(1) Wir lösen $\gamma(x) \oplus z = n$ durch $z = \gamma(x) \oplus n = 2^\ell + \sum_{k=0}^{\ell-1} z_k 2^k$ und wählen $i \in I$ mit $\langle \gamma_i(x_i) \rangle_\ell = 1$. Somit gilt $\gamma_i(x_i) \oplus z < \gamma_i(x_i)$.

In G_i existiert eine Aktion $a_i : x_i \to y_i$ mit $\gamma_i(y_i) = \gamma_i(x_i) \oplus z$.
In G erhalten wir $(i, a_i) : x \to y$ mit $\gamma(y) = \gamma(x) \oplus z = n$.

Erst dieser abschließende Satz motiviert und rechtfertigt die anfangs eingeführte Sprague–Grundy–Funktion: Genau hier liegt ihr Nutzen!

🙂 Als prototypisches Beispiel haben wir oben in Satz 1G das Nim-Spiel eingehend diskutiert; dort ist alles besonders einfach dank $\gamma_i(x_i) = x_i$.

🙂 Der Satz ist konstruktiv als gebrauchsfertiger Algorithmus formuliert. Sie können dieses Verfahren also direkt gewinnbringend anwenden... und sich dann nachfolgend den allgemeinen Beweis erarbeiten.

☹ Wenn Sie zuerst den Beweis lesen, sagt er Ihnen noch allzu wenig. Der Beweis ist einfach genial und genial einfach: Sie können ihn Schritt für Schritt leicht durcharbeiten, doch dabei besteht die Gefahr, dass der zündende Funke nicht überspringt. Das wäre hier besonders schade!

🙂 Zunächst empfehle ich, den Algorithmus in zahlreichen Bei-Spielen zu erproben. Durch eigenständiges Probieren wird Ihnen schnell klar, wie das Verfahren funktioniert und wie ein Beweis aussehen könnte. So können Sie selbst den Beweis entdecken, entwickeln und verstehen.

Beweis: Zunächst ist G lokal-endlich und artinsch: Gäbe es in G einen unendlichen Weg $x^0 \to x^1 \to x^2 \to \dots$, so ist $J = \mathrm{supp}(x^0)$ endlich, und in mindestens einer Koordinate $i \in J$ finden wir einen unendlichen Weg $x_i^{n_0} \to x_i^{n_1} \to x_i^{n_2} \to \dots$, und somit wäre G_i nicht artinsch. Widerspruch!

Für jeden Zustand $x \in X$ ist $\mathrm{supp}(x)$ endlich, also $\gamma(x) := \bigoplus_{i\in I} \gamma_i(x_i)$ wohldefiniert, denn für fast alle $i \in I$ gilt $x_i \in \partial X_i$ und somit $\gamma_i(x_i) = 0$.

Die Aussage (0) ist klar. (Führen Sie dies zur Übung sorgfältig aus!)

(1) Die Frage lautet: Warum können wir $i \in I$ wie angegeben wählen? Vom Start $m = \gamma(x)$ zum Ziel $n < m$ steht das höchste zu ändernde Bit an der angegebenen Stelle ℓ. Dank $m > n$ gilt $\langle m \rangle_\ell = 1$ und $\langle n \rangle_\ell = 0$. Demnach existiert mindestens eine Koordinate $i \in \mathbb{N}$ mit $\langle \gamma_i(x_i) \rangle_\ell = 1$.

Dies garantiert $\gamma_i(x_i) \oplus z < \gamma_i(x_i)$. In G_i existiert demnach eine Aktion $a_i : x_i \to y_i$ mit $\gamma_i(y_i) = \gamma_i(x_i) \oplus z$. In G erhalten wir $(i, a_i) : x \to y$ mit $\gamma(y) = \bigoplus_{i\in\mathbb{N}} \gamma_i(y_i) = (\bigoplus_{i\in\mathbb{N}} \gamma_i(x_i)) \oplus z = \gamma(x) \oplus z = n$, wie gewünscht.

Die beiden Eigenschaften (0,1) garantieren, dass $\gamma : X \to \mathbb{N}$ tatsächlich die Sprague–Grundy–Funktion des Spiels G ist (siehe Lemma 1H). QED

Aufgabe: Vorgelegt sei eine Position $x \in X$ im Spiel $G = (X, A, \sigma, \tau)$.
(1) Wie finden Sie von x aus *alle* möglichen Gewinnzüge (x, i, a_i, y)?
(2) Wie finden Sie zu $0 \leq n < \gamma(x)$ *alle* Züge (x, i, a_i, y) mit $\gamma(y) = n$?

Lösung: Frage (1) ist ein Spezialfall von (2), nämlich für $n = 0$.

Der Zug (x, i, a_i, y) im Spiel G bedeutet $a_i : x_i \to y_i$ im Spiel G_i. Somit gilt $\gamma(y) = \gamma(x) \oplus z$ mit $z = \gamma_i(x_i) \oplus \gamma_i(y_i)$. Wir wollen $\gamma(y) = n$.

Wir berechnen also zunächst $z := \gamma(x) \oplus n = 2^\ell + \sum_{k=0}^{\ell-1} z_k 2^k$ und suchen nun alle Züge $a_i : x_i \to y_i$ mit $\gamma_i(y_i) = \gamma_i(x_i) \oplus z$.

Im Falle $\langle \gamma_i(x_i) \rangle_\ell = 0$ gilt $\gamma_i(x_i) \oplus z > \gamma_i(x_i)$: Geeignete Züge $a_i : x_i \to y_i$ können im Spiel G_i existieren, sie sind möglich, aber nicht zwingend.

Im Falle $\langle \gamma_i(x_i) \rangle_\ell = 1$ gilt $\gamma_i(x_i) \oplus z < \gamma_i(x_i)$: Nach Definition der Grundy–Zahl γ_i existiert im Spiel G_i mindestens ein Zug $a_i : x_i \to y_i$.

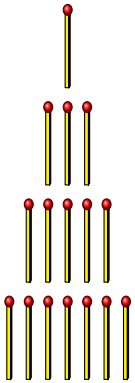
(Der Beweis zeigt, dass es mindestens eine solche Koordinate $i \in I$ gibt, genauer existiert immer eine ungerade Anzahl solcher Koordinaten.)

Wir finden so alle Züge (x, i, a_i, y) mit $\gamma_i(y_i) = \gamma_i(x_i) \oplus z$, also $\gamma(y) = n$.

Nimm eins oder zwei, nun mit mehreren Zeilen

Wir spielen „Nimm eins oder zwei“, nun parallel mit mehreren Zeilen.
Hier können wir den Sprague–Grundy–Satz 2B anwenden und testen.

Aufgabe: Nennen Sie für die gezeigte Position *alle* Gewinnzüge!



$x_4 = 1 \;\Rightarrow\; \gamma_4(x_4) = 01_{\text{bin}} \overset{?}{\to} 11_{\text{bin}} = \gamma_4(y_4)$ unmöglich

$x_3 = 3 \;\Rightarrow\; \gamma_3(x_3) = 00_{\text{bin}} \overset{?}{\to} 10_{\text{bin}} = \gamma_3(y_3) \;\Leftarrow\; y_3 = 2$
zusätzlicher, erhöhender Gewinnzug!

$x_2 = 5 \;\Rightarrow\; \gamma_2(x_2) = 10_{\text{bin}} \overset{?}{\to} 00_{\text{bin}} = \gamma_2(y_2) \;\Leftarrow\; y_2 = 3$
garantierter, reduzierender Gewinnzug

$x_1 = 7 \;\Rightarrow\; \gamma_1(x_1) = 01_{\text{bin}} \overset{?}{\to} 11_{\text{bin}} = \gamma_1(y_1)$ unmöglich

Gewinnposition? $\gamma(x) = 10_{\text{bin}} \overset{!}{\to} 00_{\text{bin}} = \gamma(y)$ Züge?

Nimm eins oder zwei, nun mit mehreren Zeilen

Wie spielen wir optimal? Wir nutzen den Sprague–Grundy–Satz 2B!
Damit finden wir *einen* Gewinnzug, bei genauem Hinsehen sogar *alle*.

Lösung: Wir untersuchen den gezeigten Spielzustand $x = (7, 5, 3, 1)$.
Dank Satz 1A kennen wir die Grundy–Zahlen $\gamma_i(x_i) = x_i \text{ rem } 3$.
Dank Satz 2B berechnen wir damit $\gamma(x) = 1 \oplus 2 \oplus 0 \oplus 1 = 2$.
Der vorgegebene Zustand x ist also ein Gewinnzustand.

Der Sprague–Grundy–Satz 2B konstruiert dazu einen Gewinnzug:
Wir wollen $\gamma(x) = 2$ reduzieren auf $0 = \gamma(y) = \gamma(x) \oplus z$, hier also $z = 2$.
Dazu reduzieren wir $\gamma_2(x_2) = 2$ auf $\gamma_2(y_2) = 0$ vermöge $x_2 = 5 \to y_2 = 3$.
Satz 2B liefert so *einen* Gewinnzug, wie versprochen, wunderbar explizit.

⚠ Satz 2B behauptet jedoch nicht, *alle* Gewinnzüge zu konstruieren.
In unserem Beispiel gibt es einen weiteren, versteckten Gewinnzug:
Wir erhöhen $\gamma_3(x_3) = 0$ auf $\gamma_3(y_3) = 2$ vermöge $x_3 = 3 \to y_3 = 2$.
(In Nim ist die Erhöhung $0 \to 2$ kein legaler Zug, hier ist es möglich.)

🙂 Erst damit sind wirklich alle Gewinnzüge gefunden.

Vergleich von Nim-Spiel und Sprague–Grundy–Satz

Der Sprague–Grundy–Satz 2B berechnet effizient die Gewinn- und Verlustpositionen. Zudem liefert er eine konkrete Konstruktion von Gewinnzügen, allgemein von Zügen $(i, a_i) : x \to y$ mit $\gamma(y) < \gamma(x)$.
Die Konstruktion liefert alle Züge, die zudem $\gamma_i(y_i) < \gamma_i(x_i)$ erfüllen.

Für Nim liefert der Sprague–Grundy–Satz 1G eine stärkere Aussage:
Die genannte Konstruktion liefert *alle* Züge $x \to y$ mit $\gamma(y) < \gamma(x)$,
denn hier gilt $\gamma_i(x_i) = x_i$, und jeder Zug $x_i \to y_i$ ist reduzierend.
Insbesondere ist die Anzahl solcher Züge in Nim immer ungerade.

In einer allgemeinen Summe $G = \bigoplus_{i \in I} G_i$ sind neben den reduzierenden Zügen $(i, a_i) : x \to y$ mit $\gamma_i(x_i) > \gamma_i(y_i)$ manchmal auch erhöhende Züge mit $\gamma_i(x_i) < \gamma_i(y_i)$ möglich. Hierüber macht Satz 2B keine Aussagen, daher auch nicht über Anzahl oder Parität der gesuchten Züge.

Wenn es nur darum geht, *irgendeinen* Gewinnzug zu finden, dann genügt die allgemeine Konstruktion des Satzes 2B. Die ambitioniertere Frage nach *allen* Gewinnzügen erfordert eine etwas genauere Untersuchung.

Vergleich von Nim-Spiel und Sprague–Grundy–Satz

Slogan: *Der Sprague–Grundy–Satz überführt jedes Spiel in ein Nim-Spiel.*
Manche sagen sogar: *in ein äquivalentes Nim-Spiel.* Was heißt das genau?

Der Sprague–Grundy–Satz konstruiert und nutzt den „Morphismus“

$$\tilde{\gamma} : G = \bigoplus_{i \in I} G_i \to \mathbb{N}^{(I)} = \bigoplus_{i \in I} \mathbb{N} : (x_i)_{i \in I} \mapsto (\gamma_i(x_i))_{i \in I}.$$

von einer beliebigen Summe von Spielen zum entsprechenden Nim-Spiel.

0 Verlustpositionen in G werden zu Verlustpositionen in Nim.

1 Gewinnpositionen in G werden zu Gewinnpositionen in Nim.

2 Gewinnzüge in Nim übersetzen sich zurück zu Gewinnzügen in G.

Allgemeiner: Die Sprague–Grundy–Zahl wird erhalten, und reduzierende Züge in Nim übersetzen sich zurück zu reduzierenden Zügen in G.

Das genügt meist. Die Rückübersetzung ist allerdings nicht surjektiv:
Es kann Gewinnzüge in G geben, die nicht von Nim-Zügen herkommen.

Der Sprague–Grundy–Morphismus ist so gesehen kein Isomorphismus;
er vereinfacht etwas und lässt dazu einige unwesentliche Details weg.

Lasker–Nim: Berechnung dank Sprague–Grundy

Eine interessante Variation entsteht, wenn man die Spielregel wie folgt festsetzt: Der am Zuge Befindliche darf irgendeinen der Haufen in zwei Haufen zerteilen oder aber, nach seinem freien Ermessen, verkleinern.
Emanuel Lasker: *Brettspiele der Völker.* Scherl Verlag, Berlin 1931.

Aufgabe: Lösen Sie dieses Spiel, **Lasker–Nim**, möglichst effizient.
Berechnen Sie zu jeder Position $x = (x_1, \dots, x_n)$ die Grundy–Zahl $\gamma(x)$.

Lösung: Das Spiel ist eine Summe, dank Sprague–Grundy gilt also

$$\gamma : \mathbb{N}^n \to \mathbb{N} : (x_1, \dots, x_n) \mapsto \gamma(x_1) \oplus \cdots \oplus \gamma(x_n).$$

Wir benötigen demnach nur noch $\gamma : \mathbb{N} \to \mathbb{N}$. Laut Spielregel gilt:

$$\gamma(x) = \operatorname{mex}\Big[\{\, \gamma(y) \mid 0 \le y < x \,\} \cup \{\, \gamma(y) \oplus \gamma(z) \mid x = y + z,\ 1 \le y \le z \,\}\Big]$$

Wir berechnen folgende Tabelle (bottom-up / memoisierte Rekursion):

$x=$	0	1	2	3	4	5	6	7	8	9	10	11	12
$\gamma=$	0	1	2	4	3	5	6	8	7	9	10	12	11

Lasker–Nim: Berechnung dank Sprague–Grundy

Übung: Rechnen Sie die obigen Fälle $x = 0, 1, 2, \dots, 12$ sorgfältig nach.
Formulieren Sie die allgemeine Regel. Beweisen Sie diese per Induktion.

Übung: Welchen Zeitaufwand $T(x)$ haben beide Methoden für $\gamma(x)$?
(0) Die naive Rekursion, ohne Speicherung von Zwischenergebnissen.
(1) Die memoisierte Rekursion, wie in obiger Tabelle illustriert.
(2) Die allgemeine Regel, die Sie soeben bewiesen haben.
Vergleichen Sie dies mit unserer Diskussion zu einzeiligem Nim.

Übung: Denken Sie sich „zufällig“ einige Spielpositionen $x \in \mathbb{N}^{(\mathbb{N})}$ aus.
Ist dies eine Gewinn- oder Verlustposition in Lasker–Nim? Wie nutzen Sie Ihre Vorarbeit? Nennen Sie jeweils alle Gewinnzüge in dieser Position.

Ähnliche Formeln gelten für alle **Take-and-Break-Spiele**: Der ziehende Spieler darf Objekte entfernen und/oder einen Haufen teilen, wozu viele verschiedene Regeln denkbar sind. In *Winning Ways*, Kapitel 4 „Taking and Breaking“ wird für die Familie der sogenannten **Octalen Spiele** eine konzise Notation eingeführt, siehe `en.wikipedia.org/wiki/Octal_game`.

Grundys Spiel: eine ungelöste Vermutung

Das Spiel beginnt mit einem Haufen von $x \in \mathbb{N}$ Objekten. Der ziehende Spieler teilt einen Haufen seiner Wahl in zwei Haufen ungleicher Größe.
Wir vereinbaren Normalspiel: Wer nicht mehr ziehen kann, verliert.

Aufgabe: Dieses Spiel heißt **Grundys Spiel**. Lösen Sie es für kleine x!
Berechnen Sie zur Position x die Grundy–Zahl $\gamma(x)$, soweit möglich.

Lösung: Die Sprague–Grundy–Funktion $\gamma : \mathbb{N} \to \mathbb{N}$ ist gegeben durch:

$$\gamma(x) = \operatorname{mex}\Big[\{\, \gamma(y) \oplus \gamma(z) \mid x = y + z,\ 1 \le y < z \,\}\Big]$$

Wir berechnen folgende Tabelle (bottom-up / memoisierte Rekursion):

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
γ	0	0	0	1	0	2	1	0	2	1	0	2	1	3	2	1	3	2	4	3	0

⚠ Anders als bei Lasker–Nim erkennen wir hier kein einfaches Muster, das unsere Rekursion zu einer effizienten Lösung abkürzen könnte.

Offene Frage: Ist diese Sprague–Grundy–Funktion $\gamma : \mathbb{N} \to \mathbb{N}$ periodisch?

Grundys Spiel: eine ungelöste Vermutung

Elwyn Berlekamp, John Horton Conway und Richard Guy vermuten in ihrem Buch *Winning Ways* (1982), dass γ tatsächlich periodisch ist.
📖 Richard Guy: *Unsolved Problems in Combinatorial Games* (1996), online verfügbar unter `library.msri.org/books/Book29/files/unsolved.pdf`

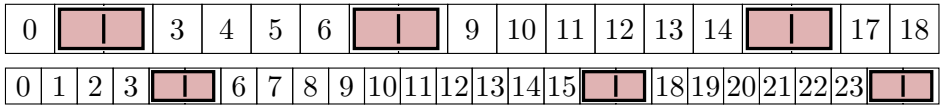
Sie können selbst weitere Werte berechnen, leichter mit Computerhilfe. So hat Achim Flammenkamp die ersten $2^{38} \approx 2.74 \cdot 10^{11}$ Werte berechnet, summarisch dargestellt unter `wwwhomes.uni-bielefeld.de/achim/grundy.html`. Ein periodisches Verhalten konnte allerdings noch niemand entdecken. Eine simple Rekursion erzeugt eine erstaunlich komplexe Folge.

Übung: Denken Sie sich „zufällig“ einige Spielpositionen $x \in \mathbb{N}^{(\mathbb{N})}$ aus.
Ist dies eine Gewinn- oder Verlustposition in Grundys Spiel? Wie nutzen Sie hier Ihre Vorarbeit? Nennen Sie alle Gewinnzüge in dieser Position.

Übung: Spielpositionen $x \in \mathbb{N}^{(\mathbb{N})}$ beschreiben wir am besten sortiert, etwa $(x_1 \ge x_2 \ge \dots \ge x_n)$ wie für Partitionen üblich. Schreiben Sie den Spielgraphen für einige kleine Startwerte (x_1) möglichst explizit aus.

Alice und Bob spielen auf einem horizontalen Spielbrett von n Feldern. Sie legen abwechselnd je einen Dominostein, jeder Stein bedeckt zwei benachbarte, noch freie Felder. Wer nicht mehr ziehen kann, verliert.

Für $n = 19$ oder $n = 26$ könnte der Zustand nach drei Zügen so aussehen:



Aufgabe: (1) Berechnen Sie die Grundy–Zahl $\gamma(n)$ für $n \leq 10$.
(2) Berechnen Sie $\gamma(n)$ für $n \leq 100$ mit einem Python–Skript.
(3) Welche der obigen Zustände sind Gewinnpositionen?
(4) Nennen Sie hierzu alle Gewinnzüge.

In der Klausur haben wir Hilfestellungen gegeben, um die Rechnungen etwas abzukürzen und die knappe Bearbeitungszeit effizient zu nutzen. Sie haben es jetzt viel besser und können den Knobelspaß genießen! Mit Frage (2) erproben Sie Ihre Python–Skills und die Memoisation.

Lösung: (1) Es gilt $\gamma(n) = \mathrm{mex}\{\, \gamma(k) \oplus \gamma(n-2-k) \mid 0 \leq k \leq n-2 \,\}$. Anfangs finden wir $\gamma(0) = \gamma(1) = \mathrm{mex}\{\,\} = 0$, und dann rekursiv

$$\begin{aligned}
\gamma(2) &= \mathrm{mex}\{\, 0 \,\} &&= 1,\\
\gamma(3) &= \mathrm{mex}\{\, 0,0 \,\} &&= 1,\\
\gamma(4) &= \mathrm{mex}\{\, 1,0,1 \,\} &&= 2,\\
\gamma(5) &= \mathrm{mex}\{\, 1,1,1,1 \,\} &&= 0,\\
\gamma(6) &= \mathrm{mex}\{\, 2,1,0,1,2 \,\} &&= 3,\\
\gamma(7) &= \mathrm{mex}\{\, 0,2,0,0,2,0 \,\} &&= 1,\\
\gamma(8) &= \mathrm{mex}\{\, 3,0,3,0,3,0,3 \,\} &&= 1,\\
\gamma(9) &= \mathrm{mex}\{\, 1,3,1,3,3,1,3,1 \,\} &&= 0,\\
\gamma(10) &= \mathrm{mex}\{\, 1,1,2,1,0,1,2,1,1 \,\} &&= 3.
\end{aligned}$$

$n=$	0	1	2	3	4	5	6	7	8	9	10
$\gamma=$	0	0	1	1	2	0	3	1	1	0	3

(2) Wir implementieren zunächst die Funktion `mex` wie auf Seite 115. Eine erste Berechnung mit Python, hastig und naiv, sieht dann so aus:

```
def gamma(n):
    if n <= 1: return 0;
    return mex({ gamma(k)^gamma(n-2-k) for k in range(0,n-1) })
```

☹ Diese naive Implementierung hat exponentiellen Aufwand. (Warum?) Ab 20 wird es langsam, schon 30 oder 40 dauert eine gefühlte Ewigkeit.

☺ Memoisation reduziert dies auf quadratischen Aufwand. (Warum?) Damit ist selbst die Rechnung bis 1000 blitzschnell. Probieren Sie es!

```
Gamma = { 0: 0, 1: 0 }
for n in range(2, 1001):
    Gamma[n] = mex({ Gamma[k]^Gamma[n-2-k] for k in range(0,n-1) })
```

☺ Die Funktion $\gamma : \mathbb{N} \to \mathbb{N}$ ist periodisch ab $x = 53$ mit Periode 34. Dieses Spiel heißt auch *Dawson Kayles*, siehe Siegel: CGT, Seite 185f. Periodizität folgt dort aus dem allgemeinen Satz 2.7 für oktale Spiele.

(3a) Dieser Zustand ist die Summe der Spiele mit 1, 4, 6, 2 freien Feldern. Die Grundy–Zahl ist $0 \oplus 2 \oplus 3 \oplus 1 = 0$. Dies ist eine Verlustposition.

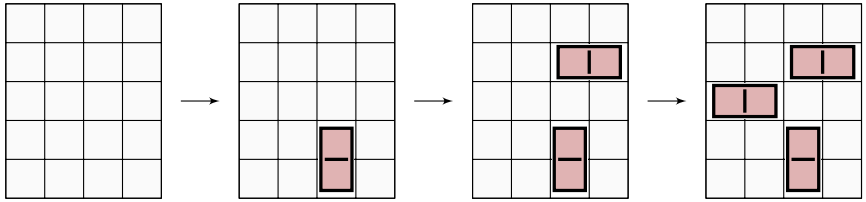
(3b) Dieser Zustand ist die Summe der Spiele mit 4, 10, 6 freien Feldern. Die Grundy–Zahl ist $2 \oplus 3 \oplus 3 = 2 \neq 0$. Dies ist eine Gewinnposition.

(4) Für den Zustand in (3b) gibt es genau die folgenden neun Gewinnzüge:

$$\{1,2\}, \{6,7\}, \{7,8\}, \{9,10\}, \{11,12\}, \{13,14\}, \{14,15\}, \{19,20\}, \{21,22\}$$

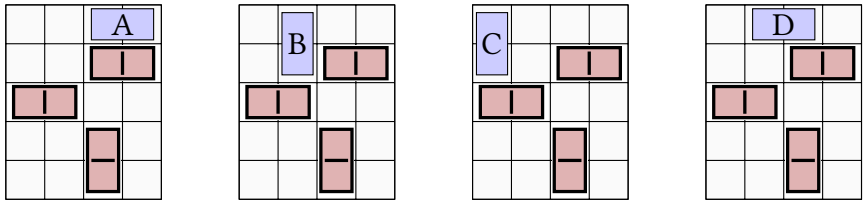
Ausführlich: Wir reduzieren die Grundy–Zahl von $2 \oplus 3 \oplus 3 = 2$ auf 0. Erster Summand von 2 auf 0: nur ein möglicher Zug. Zweiter Summand von 3 auf 1: weitere 6 Züge. Dritter Summand von 3 auf 1: weitere 2 Züge.

☺ Hier zahlt sich aus, dass wir in (1) die Rechnung dokumentiert haben. Die Grundy–Zahl $\gamma(n) = \mathrm{mex}\{\, \gamma(m) \mid n \to m \,\}$ entscheidet über Gewinn und Verlust. Für die systematische Konstruktion aller Gewinnzüge jedoch benötigen wir darüber hinaus alle Folgezustände $n \to m$ mit einem vorgegebenen Grundy–Wert $\gamma(m)$. Damit gelingt es leicht.



Das Spielfeld besteht aus Quadraten, zum Beispiel rechteckig 4×5.
Beide Spieler ziehen abwechselnd, der Ziehende legt ein Domino auf zwei benachbarte freie Quadrate. Wer nicht mehr ziehen kann, verliert.

Aufgabe: Wie lässt sich hier der Sprague–Grundy–Satz anwenden?
Im oben skizzierten konkreten Beispiel? Allgemein als Algorithmus?
Welcher der folgenden vier Züge A–D führt zum Gewinn?

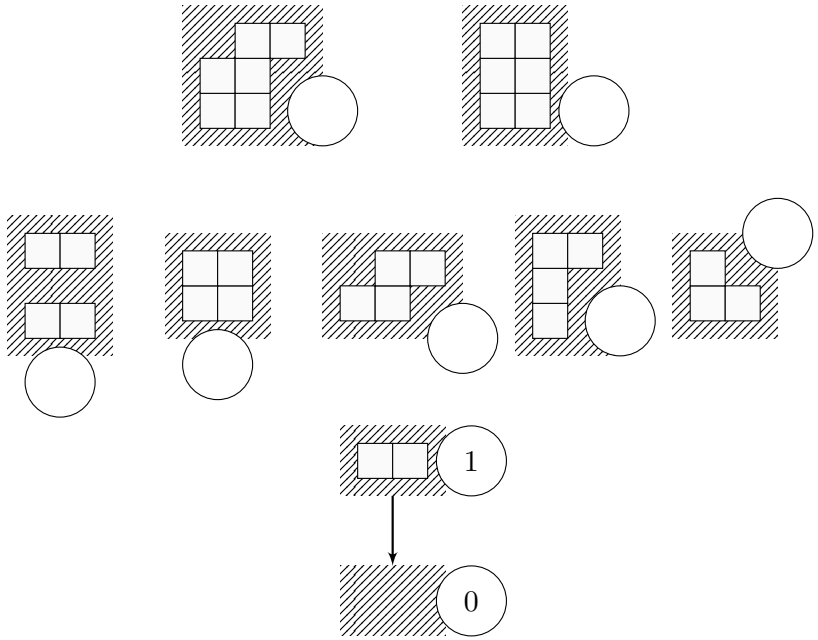


Wir parkettieren hier mit Dominos: Das ist Tetris für Fliesenleger!
Der hier entdeckte Trick gilt ganz allgemein für **Positionsspiele**:
Die Spieler erobern Positionen mit Spielsteinen und behalten diese.
Im Verlauf entstehen Inseln, also Zusammenhangskomponenten,
die sich nicht mehr gegenseitig beeinflussen. Das nutzen wir gerne:
Das Spiel zerfällt nachfolgend in die Summe seiner Komponenten!

Es genügt daher, jede **Komponente** zu analysieren, den Graphen G_i zu erstellen und seine Sprague–Grundy–Funktion γ_i zu berechnen.
Für das gesamte (End-)Spiel gilt dann $G = \bigoplus_{i \in I} G_i$ und $\gamma = \bigoplus_{i \in I} \gamma_i$.
Die Berechnung von γ gelingt auf diesem Wege wesentlich effizienter.
Anschließend lesen wir aus $x \mapsto \gamma(x)$ alle Gewinnzüge ab. Voilà!

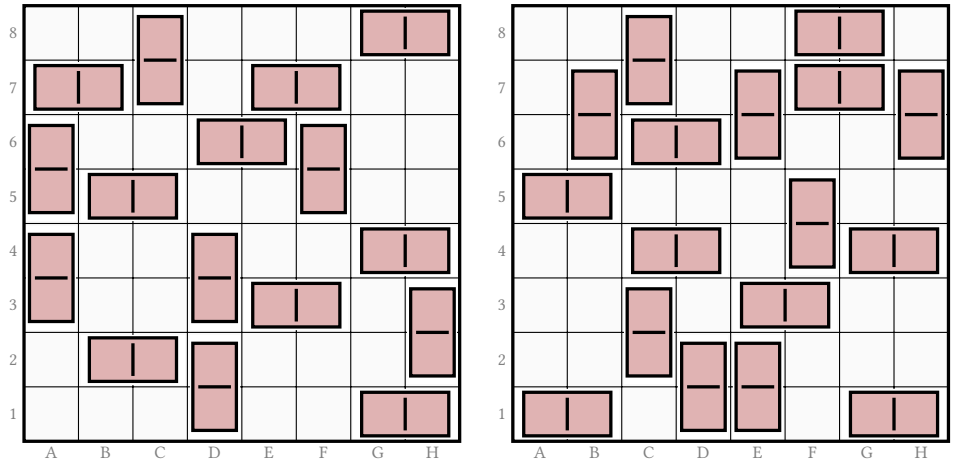
🙂 Der Sprague–Grundy–Satz lässt sich überall bei Summen einsetzen.
Diese bilden wir willkürlich, indem wir beliebige Spiele parallel spielen.
Manchmal entstehen Summen auch von ganz alleine, ohne unser Zutun.

🙂 Das entspricht übrigens der **externen** und der **internen** Summe,
wie Sie dies von Vektorräumen und ähnlichen Strukturen kennen.

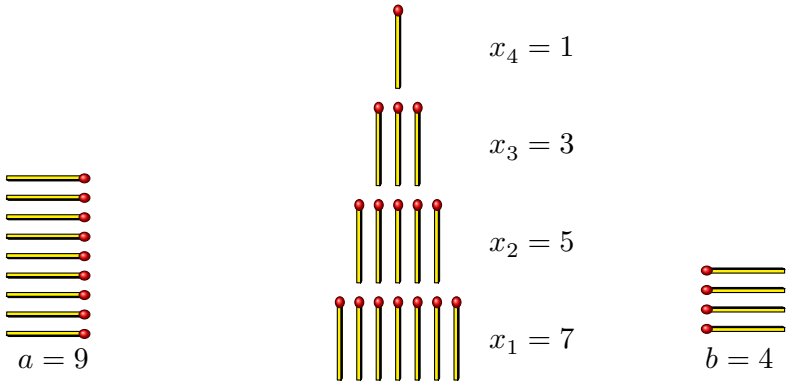


Aufgabe: (1) Bestimmen Sie den Spielgraphen und die Grundy–Zahlen.
Als Hilfestellung zeigt die vorige Folie die kleinsten möglichen Inseln.

(2) Sind die folgenden Spielstände Gewinn- oder Verlustpositionen?
Nennen Sie alle Gewinnzüge! Wie viele gibt es jeweils?



Poker-Nim: Geben ist seliger als Nehmen?



Das Spiel **Poker-Nim** wird gespielt wie Nim mit Spielständen $x \in \mathbb{N}^{(\mathbb{N})}$: Der ziehende Spieler A nimmt $s \geq 1$ Streichhölzer eines Haufens x_r zu seinem Haufen a, oder legt umgekehrt $s \geq 1$ Streichhölzer von a zu x_r. Entsprechend für Spieler B und seinen Haufen b.

Aufgabe: (1) Formalisieren Sie Poker-Nim als ein neutrales Spiel (G, v).
(2) Ist die oben gezeigte Position eine Gewinn- oder Verlustposition?
Allgemein: Wie erkennen Sie Gewinnpositionen und Gewinnzüge?

Poker-Nim: Geben ist seliger als Nehmen?

Bislang betrachteten wir nur Spiele (G, v) auf artinschen Graphen G, also ohne unendliche Wege $x_0 \to x_1 \to x_2 \to \dots$, somit ohne Schleifen. Egal wie gespielt wird, das Spiel endet nach endlich vielen Zügen.

Ausgehend von der terminalen Auszahlung $v : \partial G \to \{0, 1\}$ konstruieren wir per Rückwärtsinduktion (1c) die **Gewinnfunktion** $u : X \to \{0, 1\}$:

(T) Für jeden terminalen Zustand $x \in \partial X$ gilt $u(x) = v(x)$.
(A) Für jeden aktiven Zustand $x \in X^\circ$ gilt $u(x) = \sup_{x \to y}[1 - u(y)]$.

Das bedeutet ausführlich als Fallunterscheidung:
(A0) Falls $u(x) = 0$: Für jeden Zug $x \to y$ gilt $u(y) = 1$.
(A1) Falls $u(x) = 1$: Es existiert ein Zug $x \to y$ mit $u(y) = 0$.

Poker-Nim und Northcotts Spiel (siehe unten) sind erste Illustrationen zu **Schleifenspielen** (engl. *loopy games* nach John H. Conway 1978).

A priori ist es hier möglich, unendlich lange zu spielen. Bei optimalem Spiel kann jedoch einer der beiden Spieler seinen Gewinn erzwingen!

🙂 Zur Lösung beschränken wir explizit die Zeit bis zum Spielende.

Poker-Nim: Geben ist seliger als Nehmen?

Definition 2c: erweiterte Gewinnfunktion mit Zeitschranke

Sei $G = (X, A, \sigma, \tau)$ ein Graph mit Auszahlung $v : \partial G \to \{0, 1\}$.

Eine **erweiterte Gewinnfunktion** $\tilde{u} = (u, w) : X \to \{0, 1\} \times \mathbb{N}$ besteht aus einer Gewinnfunktion $u : X \to \{0, 1\}$ mit Zeitschranke $w : X \to \mathbb{N}$.

(T) Für jeden terminalen Zustand $x \in \partial X$ gilt $w(x) = 0$ und $u(x) = v(x)$.
(A) Für jeden aktiven Zustand $x \in X^\circ$ gilt $w(x) \geq 1$ und zudem:
(A0) Falls $u(x) = 0$: Für jeden Zug $x \to y$ gilt $u(y) = 1$ und $w(y) \leq w(x)$.
(A1) Falls $u(x) = 1$: Es gibt $x \to y$ mit $u(y) = 0$ und $w(y) < w(x)$.

Beispiel: Mit dieser genial-einfachen Technik lösen wir Poker-Nim: Zustand $(x, a, b) \in X := \mathbb{N}^{(\mathbb{N})} \times \mathbb{N}^2$, Zug $(r, s) : (x, a, b) \to (x', a', b')$ mit $r \in \mathbb{N}, s \in \mathbb{Z}, 1 \leq s \leq x_r$ oder $1 \leq -s \leq a, x' = x - s\,e_r, a' = b, b' = a + s$.

Die Gewinnfunktion ist $u : X \to \{0, 1\} : (x, a, b) \mapsto 1 \wedge \bigoplus_{i \in \mathbb{N}} x_i$, genau wie beim klassischen Nim-Spiel, nun erweitert durch die explizite Zeitschranke $w : X \to \mathbb{N} : (x, a, b) \mapsto [1 - u(x)]a + u(x)b + \sum_{i \in \mathbb{N}} x_i$.

Übung: Dieses Paar (u, w) erfüllt alle Forderungen aus Definition 2c.

Poker-Nim: Geben ist seliger als Nehmen?

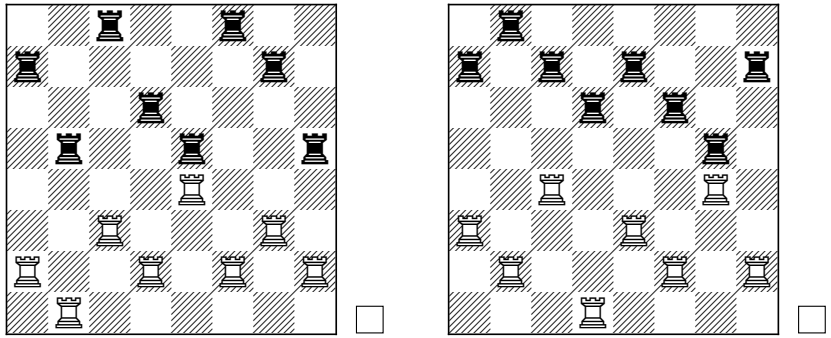
Interpretation: Beim Spielstand $x \in X$ mit $u(x) = 1$ kann der ziehende Spieler seinen Gewinn erzwingen, indem er stets reduzierend zieht (A1); er benötigt dann höchstens $w(x)$ Züge bis zu seinem sicheren Gewinn. (Wir setzen wie immer optimale Spielweise voraus.)

Im Falle $u(x) = 0$ wird der ziehende Spieler verlieren – wie immer bei optimalem Spiel seines Gegners. Der ziehende Spieler kann dies selbst nicht verhindern, sondern höchstens auf $w(x)$ Züge hinauszögern. (Verzögern lohnt sich, wenn der Gegner gelegentlich Fehler macht.)

🙂 Erlaubt unser Spiel (G, v) eine erweiterte Gewinnfunktion (u, w), so ist damit das Spiel gelöst: Jeder optimale Spielverlauf ist endlich, wir erkennen die Gewinnpositionen an der Eigenschaft $u(x) = 1$ und die (reduzierenden) Gewinnzüge $x \to y$ an $u(y) = 0$ und $w(y) < w(x)$.

Übung: Wir spielen um Gewinn $+1$ oder Verlust -1, diesmal zusätzlich mit Diskont $\delta \in]0, 1[$, etwa $\delta = 1/2$. Existiert zu jedem Graphen (G, v) eine Gewinnfunktion $\bar{u}$? Wie interpretieren Sie $\bar{u}$? Welche Beziehung besteht zur obigen Gewinnfunktion u mit Zeitschranke w?

Northcotts Spiel: Türme ziehen ohne Ende?



Gezogen wird abwechselnd weiß und schwarz (Markierung am Rand).
Der Ziehende bewegt einen Turm nur vertikal und soweit Platz ist.
Wir vereinbaren Normalspiel: Wer nicht mehr ziehen kann, verliert.

Aufgabe: (1) Formalisieren Sie dies als ein neutrales Spiel (G, v).
Ist der Spielgraph G hier endlich? Wie viele Zustände hat er?
Ist der Spielgraph artinsch? Endet jeder (optimale) Spielverlauf?
(2) Sind die beiden obigen Positionen Gewinn- oder Verlustposition?
Allgemein: Wie erkennen Sie Gewinnpositionen und Gewinnzüge?

Northcotts Spiel: Türme ziehen ohne Ende?

Dieses bekannte Beispiel heißt auch **Northcotts Spiel**.
Es ist insofern ungewöhnlich (und interessant!), als die Regeln allein nicht garantieren, dass jeder Spielverlauf wirklich endet.
Es ist hier durchaus möglich, unendlich lange zu spielen!

Überraschend zeigt sich jedoch, dass bei optimalem Spiel einer der beiden Spieler seinen Gewinn erzwingen kann. Sehen Sie wer und wie?
Dahinter versteckt sich eine weitere Variante des obigen Poker-Nim.
Das ist genau das Ziel der obigen Aufgabe. Probieren Sie es!

Übung: Ist dieses Spiel die Summe seiner Spalten? Falls ja, so lässt sich der Sprague–Grundy–Satz 2B direkt anwenden. Falls nein, so lässt sich das Spiel vielleicht geschickt umformulieren in eine äquivalente Summe. Das vereinfacht die Analyse und ermöglicht eine effiziente Lösung!

Übung: Untersuchen Sie folgende Variante: Über bzw. unter jeder Zeile markiert eine Münze, welcher der beiden Türme als nächstes gezogen wird. Nach jedem Zug in dieser Spalte wird die Münze nach unten bzw. oben umgelegt. Ist dieses variierte Spiel die Summe seiner Spalten?

Northcotts Spiel: Türme ziehen ohne Ende?

Wir lösen Northcotts Spiel durch eine geeignete Zeitschranke:

Definition 2D: erweiterte Grundy–Funktion mit Zeitschranke

Sei $G = (X, A, \sigma, \tau)$ ein Graph. Eine **erweiterte Grundy–Funktion** $\tilde{\gamma} = (\gamma, w) : X \to \mathbb{N} \times \mathbb{N}$ besteht aus einer Grundy–Funktion $\gamma : X \to \mathbb{N}$ und einer Zeitschranke $w : X \to \mathbb{N}$ mit folgenden Eigenschaften:

(T) Für jeden terminalen Zustand $x \in \partial X$ gilt $w(x) = 0$ und $\gamma(x) = 0$.
(A) Für jeden aktiven Zustand $x \in X^\circ$ gilt $w(x) \geq 1$ und zudem:
(A0) Für jeden Zug $x \to y$ gilt $\gamma(y) \neq \gamma(x)$ und $w(y) \leq w(x)$.
(A1) Zu $0 \leq n < \gamma(x)$ existiert $x \to y$ mit $\gamma(y) = n$ und $w(y) < w(x)$.

Interpretation: Bei $\gamma(x) > 0$ gewinnt der ziehende Spieler das Spiel in $\leq w(x)$ Zügen, wenn er immer reduzierend zieht wie in (A1) erklärt.

🙂 Erlaubt unser Spiel $(G, 0)$ eine erweiterte Grundy–Funktion (γ, w), so ist damit das Spiel gelöst: Jeder optimale Spielverlauf ist endlich, wir erkennen die Gewinnpositionen an der Eigenschaft $\gamma(x) \geq 1$ und die (reduzierenden) Gewinnzüge $x \to y$ an $\gamma(y) = 0$ und $w(y) < w(x)$.

Northcotts Spiel: Türme ziehen ohne Ende?

Lemma 2E: Eindeutigkeit und Existenz

Sind (u, w) und (u', w') erw. Gewinnfunktionen zu (G, v), so gilt $u = u'$.
Sind (γ, w) und (γ', w') erw. Grundy–Funktionen zu $(G, 0)$, so gilt $\gamma = \gamma'$.

Ist G artinsch, so existiert zu (G, v) eine erw. Gewinnfunktion (u, w).
Ist G artinsch und lokal-endlich, so existiert zum Normalspiel $(G, 0)$ eine erw. Grundy–Funktion (γ, w). Wie zuvor gilt $\nu = \gamma \wedge 1$.

🙂 Für lösbare Schleifenspiele gilt der Satz von Sprague–Grundy 2B:

Satz 2F: Sprague–Grundy für Schleifenspiele

Gegeben sei eine Familie von Graphen G_i indiziert durch $i \in I$, jeder mit einer erweiterten Grundy–Funktion $(\gamma_i, w_i) : X_i \to \mathbb{N} \times \mathbb{N}$.

Dann erlaubt ihre Summe $G = \bigoplus_{i \in I} G_i$ die erweiterte Grundy–Funktion $(\gamma, w) : X \to \mathbb{N} \times \mathbb{N}$ mit $\gamma(x) = \bigoplus_{i \in I} \gamma_i(x_i)$ und $w(x) = \sum_{i \in I} w_i(x_i)$.

Übung: Rechnen Sie Satz und Lemma sorgfältig nach! Damit lösen Sie Summen von Schleifenspielen. Wie lösen Sie damit Northcotts Spiel?

Die Sprague–Grundy–Theorie untersucht das Normalspiel: Wer nicht mehr ziehen kann, verliert. Beim Misèrespiel gilt umgekehrt: Wer nicht mehr ziehen kann, gewinnt. Das klingt zunächst symmetrisch, manches lässt sich übertragen, doch die Rechnungen verändern sich erheblich.

Ohne Beschränkung der Allgemeinheit darf angenommen werden, dass der Sieg dem Partner zufällt, der die Partie beendet. Man braucht nur alle Endstellungen zu verbieten, in denen der zuletzt Ziehende verliert.
Roland Sprague (1935)

Aufgabe: Wir betrachten Nim $G = (X, A, \sigma, \tau)$, doch nun als Misèrespiel. Beschreiben Sie dies als Normalspiel auf einem geeigneten Teilgraphen.

Lösung: Für Nim haben wir die Zustandsmenge $X = \mathbb{N}^{(\mathbb{N})}$. Aktionen $(r, s) : x \to y$ sind Paare $(r, s) \in \mathbb{N}^2$ mit $1 \leq s \leq x_r$ und $y = x - s\, e_r$. Der einzige terminale Zustand ist demnach $x = 0$, aktiv ist $x \neq 0$. Wir betrachten demnach den vollen Teilgraph $G' = (X', A', \sigma', \tau')$ auf $X' = X \setminus \{0\}$. Das Misèrespiel auf G ist dann das Normalspiel auf G'.

The game [of Nim in normal play] may be modified by agreeing that the player who takes the last counter from the table loses. [...] The safe combinations are the same as before, except that an odd number of piles, each containing one, is now safe, while an even number of ones is not safe.
Charles Bouton (1901)

Aufgabe: Explizieren und beweisen Sie diese Formel der Gewinnfunktion.

Satz 3A: Boutons Lösung des Nim-Spiels, normal vs misère

Für das Normalspiel von Nim haben wir die vertraute Gewinnfunktion

$$\nu : X \to \{0,1\} : \nu(x) = \begin{cases} 0 & \text{falls } \bigoplus_{i\in\mathbb{N}} x_i = 0, \\ 1 & \text{falls } \bigoplus_{i\in\mathbb{N}} x_i \geq 1. \end{cases}$$

Im Misèrespiel von Nim hingegen gilt die abgewandelte Formel

$$\mu : X \to \{0,1\} : \mu(x) = \begin{cases} 1 - \nu(x) & \text{falls } \max(x) \leq 1, \\ \nu(x) & \text{falls } \max(x) \geq 2. \end{cases}$$

Als unterhaltsames *pub game* von Scam School, youtu.be/mR6mVm4SuRw, schön einfach, aber auch etwas unpräzise. Klarheit schafft der Beweis!

Beweis: (1) Für den Endzustand $x = 0$ gilt $\mu(x) = 1 = 1 - \nu(x)$. Für $\max(x) \leq 1$ besteht x aus n Haufen der Größe 1. Jeder Zug $x \to y$ führt zu $n - 1$ Haufen der Größe 1, also $\mu(x) = 1 - (n \operatorname{rem} 2) = 1 - \nu(x)$.

Kurzum: Im Misèrespiel ist also eine un/gerade Anzahl von Einserhaufen eine Verlust/Gewinnposition, im Normalspiel jedoch genau umgekehrt.

Sei nun $\max(x) \geq 2$. Wir unterscheiden dabei zwei Fälle:

(2) Existiert nur noch ein Haufen $i \in \mathbb{N}$ mit $x_i \geq 2$, so gilt $\nu(x) = 1$ und ebenso $\mu(x) = 1$: Der ziehende Spieler kann eine gerade Anzahl von Einserhaufen übergeben, also eine Misère-Verlustposition gemäß (1).

(3) Angenommen, es existieren mehrere Haufen $i \in \mathbb{N}$ mit $x_i \geq 2$. Jeder Zug $x \to y$ übergibt dann mindestens einen Haufen mit $y_i \geq 2$, wir gelangen also immer wieder zum Fall (3) und schließlich zu (2).

Somit gilt für $\max(x) \geq 2$ immer $\mu(x) = \nu(x)$. QED

🙂 Im Nim-Spiel können wir die Gewinnfunktion $\nu : X \to \{0,1\}$ leicht berechnen dank Boutons Lösung 1F. Die Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N} : \gamma(x) = \bigoplus_{i\in\mathbb{N}} x_i$ ist ebenso leicht dank 1G. Allgemein für Summen von Spielen haben wir den Satz 2B von Sprague–Grundy.

🙂 Das Misèrespiel auf G entspricht dem Normalspiel auf G', in dem alle terminalen Zustände von G gelöscht wurden. Für die Theorie ist das eine elegante Formulierung, doch die konkreten Rechnungen verändern sich dramatisch. Oft erweist sich das Misèrespiel als wesentlich schwieriger.

🙂 In Misère-Nim können wir die Gewinnfunktion $\mu : X \to \{0,1\}$ und entsprechend die normale Gewinnfunktion $\nu' : X' \to \{0,1\}$ erfreulich leicht berechnen durch den Kniff 3A. Das ist ein glücklicher Zufall.

☹ Für die Sprague–Grundy–Funktion $\gamma' : X' \to \mathbb{N}$ von Misère-Nim ist jedoch keine geschlossene Formel oder effiziente Berechnung bekannt: Der Sprague–Grundy–Satz funktioniert wunderbar für Normalspiele, doch für Misèrespiele scheint er keine Entsprechung zu erlauben.

Charles Bouton 1901: Wie alles begann...

NIM, A GAME WITH A COMPLETE MATHEMATICAL THEORY.

By Charles L. Bouton.

The game here discussed has interested the writer on account of its seeming complexity, and its extremely simple and complete mathematical theory.* The writer has not been able to discover much concerning its history, although certain forms of it seem to be played at a number of American colleges, and at some of the American fairs. It has been called Fan-Tan, but as it is not the Chinese game of that name, the name in the title is proposed for it.

1. Description of the Game. The game is played by two players, A and B. Upon a table are placed three piles of objects of any kind, let us say counters. The number in each pile is quite arbitrary, except that it is well to agree that no two piles shall be equal at the beginning. A play is made as follows:—The player selects one of the piles, and from it takes as many counters as he chooses; one, two, . . ., or the whole pile. The only essential things about a play are that the counters shall be taken from a single pile, and that at least one shall be taken. The players play alternately, and the player who takes up the last counter or counters from the table wins.

It is the writer's purpose to prove that if one of the players, say A, can leave one of a certain set of numbers upon the table, and after that plays without mistake, the other player, B, *cannot* win. Such a set of numbers will be called a *safe combination*. In outline the proof consists in showing that if A leaves a safe combination on the table, B at his next move cannot leave a safe combination, and whatever B may draw, A at his next move can again leave a safe combination. The piles are then reduced, A always leaving a safe combination, and B never doing so, and A must eventually take the last counter (or counters).

2. Its Theory. A *safe combination* is determined as follows: Write the number of the counters in each pile in the binary scale of notation,† and place these numbers in three horizontal lines so that the units are in the same vertical column. If then the sum of *each* column is 2 or 0 (*i. e.* congruent to 0, mod. 2), the set of numbers forms a safe combination.

Literatur zur kombinatorischen Spieltheorie

Die Originalartikel sind online erhältlich und noch immer schön zu lesen:

📖 Charles L. Bouton: *Nim, a game with a complete mathematical theory.* Annals of Mathematics 3 (1901) 35–39. doi.org/10.2307/1967631

Roland P. Sprague: *Über mathematische Kampfspiele.* Tôhoku Math. 41 (1935) 438–444. www.jstage.jst.go.jp/article/tmj1911/41/0/41_0_438/_pdf

Roland P. Sprague: *Über zwei Abarten von Nim.* Tôhoku Math. 43 (1937) 451–454. www.jstage.jst.go.jp/article/tmj1911/43/0/43_0_351/_pdf

Patrick M. Grundy: *Mathematics and Games.* Eureka 2 (1939) 6–8

Kombinatorische Spieltheorie ist heute ein riesiges Forschungsgebiet. Die monumentalen Klassiker WW und ONAG und das Lehrbuch CGT:

📖 Elwyn R. Berlekamp, John H. Conway, Richard K. Guy:
Winning Ways for Your Mathematical Plays. A K Peters 2001-2004
Gewinnen: Strategien für mathematische Spiele. Vieweg 1985-1986

John H. Conway: *On Numbers and Games.* A K Peters 2000

Aaron N. Siegel: *Combinatorial Game Theory.* AMS 2013

Roland Sprague 1935: Einleitung

Über mathematische Kampfspiele,

von

R. Sprague in Berlin-Charlottenburg.

1. Gegenstand dieser Arbeit sind Spiele mit folgendem Partieverlauf: eine Anfangsstellung wird nach einer beschränkten Anzahl von abwechselnden Zügen zweier Personen in eine Endstellung übergeführt, die keinen Zug mehr zulässt und den Sieg einer der beiden Personen ergibt.

Ohne Beschränkung der Allgemeinheit darf angenommen werden, dass der Sieg dem Partner zufällt, der die Partie beendet. Man braucht nur alle Endstellungen zu verbieten, in denen der zuletzt Ziehende verliert.

Diese Spiele mögen kurz als "Kampfspiele" bezeichnet werden.

2. E. Lasker(1), von dem der Name "mathematische Kampfspiele" herrührt, hat diese Spiele methodisch untersucht und bemerkt, dass ihre Stellungen in zwei Klassen zerfallen, nämlich "Gewinnstellungen" und "Verluststellungen". In Gewinnstellungen, kurz: "G", kann *der* Spieler den Sieg erzwingen, der am Zuge ist, in Verluststellungen, "V", der andere. Das Gewinnproblem eines Kampfspiels ist mit der Kenntnis seiner V erledigt: wer eine V herbeiführt, gewinnt, indem er seinem Gegner stets wieder eine V hinterlässt.

3. Zur Erläuterung diene der Hinweis auf ein altbekanntes Kampfspiel. Von einem Haufen von Dingen werden in jedem Zuge eine beschränkte Anzahl fortgenommen, nämlich mindestens 1, höchstens m. Jede Stellung ist dann durch die Anzahl der vorhandenen Dinge gekennzeichnet. Als V erweisen sich die ganzzahligen Vielfachen von $m+1$; denn wer eine Stellung $k(m+1)$ erreicht, gelangt bei beliebigen Zügen des Gegners schrittweise zu $(k-1)\cdot(m+1)$, $(k-2)\cdot(m+1)$ und so weiter bis zur Endstellung 0 und gewinnt. Jede andere Stellung ist G, weil sie in einem Zuge zu einer V gemacht werden kann.

Roland Sprague 1935: Hauptsatz

Satz III: Schreibt man die Zahlen $a, b, c, \ldots$ dyadisch untereinander, in einem Schema wie zur Addition dekadischer Zahlen, so lässt sich eine dyadische Zahl R bilden, deren Ziffern 0 oder 1 heissen, je nachdem die entsprechende Kolonne des Schemas eine gerade oder eine ungerade Anzahl von Einsen aufweist. Der Wert von R ist der Rang der Stellung $a, b, c, \ldots$ im Nim.

Beweis: Wegen der Eindeutigkeit der Zuordnung in Satz I genügt es zu zeigen, dass die Zahlen R die Bedingungen A) und B) erfüllen.

Zu A). Jeder Zug verändert eine der Zahlen $a, b, c, \ldots$, also eine Zeile des Schemas, und damit eine oder mehrere Kolonnen in je einer Ziffer. Demnach ändert sich auch R.

Zu B). Ist $R>0$ und P eine dyadische Zahl, für die $0 \leqq P < R$ gilt, so unterscheiden sich R und P, von links her verglichen, zum ersten Mal an einer Stelle, wo P eine 0, R eine 1 aufweist. In der entsprechenden Kolonne des Schemas steht dann eine ungerade Anzahl von Einsen, also mindestens eine. Irgendeine dieser Einsen, oder die einzige, befinde sich in der Zeile, die zum Haufen x gehört. Durch Verkleinerung von x lässt sich erreichen, dass diese 1 zu 0 wird und überdies die weiter nach rechts liegenden Ziffern der Zeile sich ändern oder nicht, je nachdem P in den entsprechenden Stellen von R abweicht oder nicht. Nach dieser Verkleinerung von x ist P die dem Schema zugeordnete dyadische Zahl.

Hiermit ist Satz III bewiesen.