Niklas Hofer

Algorithmen zum Aufzählen und Abzählen bei Günter Rote

(Stand: 8. Februar 2005)

Inhaltsverzeichnis

1 Einleitung

1.1 Begriffsklärung

- Aufzählen/Erzeugen:
  - generating/enumeration
  - alle Objekte werden durchlaufen
- Abzählen:
  - counting
  - nur Anzahl interessant
- Durchsuchen: Suchen des besten Objekts.

1.2 Motivation

Eine 10-elementige Menge besitzt $2^{10} = 1024$ Teilmengen. Bei 100 Elementen sind es schon 2^{100}. Ist es trotz dieser enormen Anzahl von Teilmengen möglich, Aussagen über einzelne Mengen zu machen?

1.3 Anwendungen

- kombinatorische Zählaufgaben (Statistik, mathematische Physik)
- Optimierungsaufgaben (heuristische Verfahren)
- Analyse von Algorithmen in Form von Frage-und-Antwort-Spielen, Entscheidungsbäumen, Gesellschaftsspielen

Beispiele

- Wir haben 5 Punkte in der Ebene gegeben, gesucht ist die komplexe Hülle, also das größtmögliche Polygon.
- Optimierung von Sortieralgorithmen, so dass sie im Worst Case möglichst wenig Kosten verursachen.

2 Aufzählen aller Teilmengen

Die Grundmenge $S = \{0, 1, \cdots, n-1\}$ sei vorausgesetzt.

Definition 2.1 (Charakteristischer Vektor). Jede Teilmenge $T \subseteq S$ entspricht einer Folge $(a_{n-1}, a_{n-2}, \cdots, a_1, a_0)$ von n Bits $a_i \in \{0, 1\}$. Dabei ist $a_i = 1$ gdw. $i \in T$. Diese Folge wird *charakteristischer Vektor* genannt.

Somit gibt es 2^n Möglichkeiten bzw. verschiedene Teilmengen von S.

Beispiel 2.1. Für $n = 6$, die Teilmenge $T = \{0, 4\}$ entspricht der Folge $(0, 1, 0, 0, 0, 1) \equiv 010001_2 = 17$.

2.1 Grundaufgaben beim Aufzählen

Sei S eine endliche Menge mit M Objekten.

Definition 2.2 (Reihenfolge). Die *Reihenfolge* definiert eine Bijektion zwischen den Objekten einer Menge und den Zahlen $(0, 1, \ldots, M-1)$.

Definition 2.3 (Rang). Die einem Objekt $x \in S$ entsprechende Zahl $f(x) \in \{0, 1, \ldots M-1\}$ heißt der *Rang* von x.

Grundaufgaben beim Aufzählen in einer bestimmten Reihenfolge (z.B. $\leqslant_{lex}$, s. 2.2) sind:

Rangbestimmung: Die Rangbestimmung fordert das Bestimmen von $f(x)$ (en: *ranking*). (Die wievielte Teilmenfge ist T?)

Inverse Rangbestimmung: Die inverse Funktion wird mit $f^{-1}(i)$ (en: *unranking*) bezeichnet. (Was ist die Teilmenge $T_k,\ 0 \leqslant k < M$?)

Nachfolgerbestimmung: Bestimmen des Nachfolgers: $next(x) = f^{-1}(f(x) + 1)$, falls $f(x) < M - 1$.

2.2 Lexikographische Reihenfolge

Seien $\alpha = \alpha_1\alpha_2 \ldots \alpha_n,\ \beta = \beta_1\beta_2 \ldots \beta_m$ zwei Folgen mit vergleichbaren Elementen.

Definition 2.4 (Lexikographische Reihenfolge). Die Folge α ist *lexikographisch kleiner* als die Folge β: $\alpha <_{lex} \beta \Leftrightarrow$
$[\exists i, 1 \leqslant i \leqslant min(m,n), (\forall j : 1 \leqslant j \leqslant i-1, \alpha_j = \beta_j) \wedge \alpha_i < \beta_i]$
$\vee\ [n < m \wedge \forall j : 1 \leqslant j \leqslant n \Rightarrow \alpha_j = \beta_j]$

Bemerkung 2.5. Lexikographische Reihenfolge von Bitketten entspricht numerischer Reihenfolge der entsprechenden Binärzahlen, wenn die Bitketten die gleiche Länge haben.

$$(\alpha_{n-1} \ldots, \alpha_n) \rightarrow \sum_{i=0}^{n-1} \alpha_i 2^i$$

Rangbestimmung: Um den Rang einer Teilmenge bezüglich lexikographischen Reihenfolge zu bestimmen, betrachten wir deren charakteristischen Vektor als Binärzahl.

Dann gilt die Abbildung $f : T \mapsto \sum_{i \in T} 2^i$, wobei T als Menge betrachtet wird.

Inverse Rangbestimmung: Wir interpretieren die Binärdarstellung des Rangs als Bitfolge.

Nachfolgeroperation: Die Nachfolgeroperation entspricht Addition von 1 zum Rang der Teilmenge.

Algorithm 1 Nachfolger in lexikographischen Reihenfolge

```
1: j = n − 1
2: while a_j = 1 do
3:    if j = 0 then
4:       letztes Element
5:    else
6:       j = j − 1
7:    end if
8: end while
9: a_j = 1
10: for k = j + 1 to n − 1 do
11:    a_k = 0
12: end for
```

2.3 Gray-Codes

Betrachten wir ein Messrad mit 16 Sektoren. Die Sektoren können wir mit 4-bitigen Binärzahlen in lexikographische Reihenfolge kodieren, wobei die weißen Streifen eine 0 bedeuten und die schwarzen eine 1. Falls nun der Abtaster genau am Übergang zwischen zwei Sektoren stehen bleibt, z.B. zwischen 1111 und 0000, kann das Ergebnis 1000 oder auch 0111 lauten. Um solche Ablesefehler zu vermeiden, wäre es gut, eine Kodierung oder eine Reihenfolge zu haben, bei der *von einer Menge zur nächsten sich möglichst wenig ändert.*

Definition 2.6 (Gray-Code). Die Reihenfolge auf 2^n Binärzahlen, bei der zwei benachbarte Zahlen sich nur in einem Bit unterscheiden heißt *Gray-Code.*

Betrachten wir einen n-dimensionalen (Hyper)Würfel als Graphen. Der Hyperwürfel hat einen Knoten für jede Bitfolge der Länge n und eine Kante zwischen zwei Folgen, wenn sie sich an genau einer Stelle unterscheiden.

Bemerkung 2.7. Ein Gray-Code entspricht einem Hamilton'schen Pfad (bzw. Kreis) im Hyperwürfel.

Um einen Gray-Code systematisch zu erzeugen, geht man induktiv vor, indem man sich am Aufbau des Hyperwürfels orientiert.

Der n-dimensionale Hyperwürfel C_n entsteht aus dem n-1-dimensionalen Hyperwürfel C_{n-1}, indem man zwei Kopien von C_{n-1} nimmt und die entsprechenden Knoten der beiden Kopien miteinander verbindet.

Sei nun ein Hamiltonischer-Pfad in C_{n-1} gegeben. Durch Verbinden der Hamilton-Pfade in den beiden Kopien erhält man einen Hamilton-Pfad, der ganz C_n abdeckt. Dabei muss einer der Pfade rückwärts durchgangen werden (ausgehend von der Symmetrie der beiden Pfade)

Definition 2.8 (Reflektierter binärer Gray-Code). Der *reflektierte binärer Gray-Code* G_n läßt sich berechnen durch:

$$G_0 = (\varepsilon), \qquad G_n = (0G_{n-1}, (1G_{n-1})^R),$$

wobei

- G_n bezeichnet eine Folge von Bitketten der Länge n,

- xG – an jedes Element von G wird vorne ein x angehängt,
- G^R – die Reihenfolge der Elemente von G wird umgekehrt.

Beispiel 2.2. $G_1 = (0, 0)$, $G_2 = (00, 01, 11, 10)$,
$G_3 = (000, 001, 011, 010, 110, 111, 101, 100)$

Satz 2.9. G_n ist ein Gray-Code für Bitfolgen der Länge n, der mit $00\ldots0$ begint und mit $100\ldots0$ aufhört.

Beweis. Durch Induktion zeigen wir, dass:

- G_n hat Länge 2^n,
- Jedes Wort kommt genau einmal vor,
- Aufeinanderfolgende Wörter sind benachbart.

Rätsel: Betrachten Sie die k-elementigen Teilmengen T_k von S, mit $|S| = n$. Wir wissen dass $|T_k| = \binom{n}{k}$ und $|T_k| = |T_{n-k}|$. Das wird üblicherweise gezeigt durch die Bijektion $f : T_k \to T_{n-k},\ f(T) = S \setminus T$.

Für $k \leqslant \frac{n}{2}$, gesucht ist eine Bijektion f mit $T \subseteq f(T)$.

2.3.1 Nachfolgerbestimmung beim reflektieren Gray-Code

Aus dem Satz 2.9 wissen wir, dass G_n mit $000\ldots0$ beginnt und mit $100\ldots0$ endet. Wenn die Anzahl der gesetzten Bits gerade ist, dann muss das letzte Bit gekippt werden. Ansonsten muss von rechts beginnend die erste 1 gesucht werden, die dann invertiert wird.

Prüfbit / Parität: $a_\infty = a_{n-1} \underbrace{\oplus}_{\text{Summe modulo 2}} a_{n-2} \oplus \ldots \oplus a_0$.

Algorithm 2 Nachfolger im reflektierten Gray-Code

```
1: a_∞ = a_{n-1} ⊕ a_{n-2} ⊕ ··· ⊕ a_0
2: if a_∞ = 0 then
3:    a_0 = 1 − a_0; a_∞ = 1
4: else
5:    bestimme das kleinste j mit a_j = 1
6:    if j = n − 1 then
7:       Halt
8:    end if
9:    a_{j+1} = 1 − a_{j+1}; a_∞ = 0
10: end if
```

Laufzeitanalyse: Die Schleife zum Suchen der ersten 1 hat im schlimmster Fall $O(n)$ Laufzeit. In Summe (zum Erzeugen aller 2^n Wörter) = Summe der δ-Werte.

$$O(\underbrace{(\delta_0+1)+(\delta_0+1)+\cdots+(\delta_{2^m-2}+1)+n)}_{L}$$

$$L = 2^n + \sum_{i=0}^{2^n-2} \delta_i + (n-1)$$

$$= 2^m + \sum_{k=0}^{n-1} \frac{2^n}{2^{k+1}} k + (n-1) = O(2^n + 2^n + n) = O(2^n)$$

Anzahl	δ-Wert
$\frac{2^n}{2}$	0
$\frac{2^n}{4}$	1
$\vdots$	$\vdots$
$\frac{2^n}{2^k}$	$(k-1)$
$\vdots$	$\vdots$
2	$(n-2)$
1	$(n-1)$

, $\quad \sum_{k=0}^{\infty} \frac{k}{2^{k_1}} = 1$

Im Mittel ist also die Laufzeit einer Nachfolgeroperation konstant.

2.3.2 Rangbestimung und inverse Rangbestimmung

$rang(a_{n-1}, a_{n-2}, \ldots, a_0) = (b_{n-1}, b_{n-2}, \ldots, b_0)$ als Binärzahl.

Rangbestimung:

$$\begin{aligned} b_{n-1} &= a_{n-1} \\ b_{n-2} &= a_{n-2} \oplus b_{n-1} \\ b_{n-3} &= a_{n-3} \oplus b_{n-2} \\ &\vdots \\ b_1 &= a_1 \oplus b_2 \\ b_0 &= a_0 \oplus b_1 \end{aligned}$$

bzw:

$$\begin{aligned} b_{n-1} &= a_{n-1} \\ b_{n-2} &= a_{n-1} \oplus a_{n-2} \\ b_{n-3} &= a_{n-1} \oplus a_{n-1} \oplus a_{n-3} \\ &\vdots \\ b_0 &= a_{n-1} \oplus \ldots \oplus a_1 \oplus a_0 \end{aligned}$$

Inverse Rangbestimung:

$$\begin{aligned} a_{n-1} &= b_{n-1} \\ a_{n-2} &= b_{n-2} \oplus b_{n-1} \\ &\vdots \\ a_0 &= b_1 \oplus b_0 \end{aligned}$$

Beispiel 2.3. $rang(0101) = 0110_2 = 6$

Bemerkung 2.10. Folge der $(b_{n-1}, b_{n-2}, \ldots, b_0)$ ist in lexikographischer Reihenfolge.

Inverse Rangbestimmung mit logischen Operationen auf Bitketten: Sei Rang $(b_{n-1}, b_{n-2}, \ldots, b_0) = i$ als Binärzahl gegeben. Gesucht wird $a = unrank(b)$. i und a müssen mindestens n Bits enthalten.

```
int i;
a = ( i >> 1 ) ^ i;
```

2.3.3 δ-Folge

Definition 2.11 (δ-Folge). Die δ-Folge eines Gray-Codes ist die Folge der Positionen $\delta_0, \delta_1, \ldots, \delta_{2^n-2}, \delta_{2^n-1} \in \{0, \ldots, n\}$, wo sich jeweils das Bit beim Übergang zum nächsten Wort ändert.

Der Übergang vom Wort i zum Wort $(i+1)$ wird mit $a_{\delta_i} := 1 - a_{\delta_i}$ gebildet.

Die δ-Folge des reflektierten Gray-Codes wird wie folgt gebildet:

$$\delta(G) = \delta(G_{n-1}), n-1, (\delta(G_{n-1}))^R$$

Beispiel 2.4.

$$\begin{aligned}
\delta(G_1) &= 0 \\
\delta(G_2) &= 0, 1, 0 \\
\delta(G_3) &= 0, 1, 0, 2, 0, 1, 0 \\
\delta(G_4) &= 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0
\end{aligned}$$

Wenn man an $\delta(G_n)$ das Element $n-1$ anhängt, erhält man einen Gray-*Zyklus*, der wieder zum Anfang zurückspringt. Zudem ist die δ-Folge symmetrisch $\left(\delta(G_n) = (\delta(G_n))^R\right)$. Diese δ-Folge beschreibt ebenfalls die Anzahl der Überträge beim Hochzählen von $00\ldots0$ bis $11\ldots1$.

3 Verallgemeinerung auf n-Tupel zur Basis m

Wir betrachten n-Tupel $a = (a_1, \cdots, a_n)$, mit $0 \leqslant a_i < m_i$. Wir bilden nun einen (Such-) Baum, so dass Knoten auf Tiefe k den Teilfolgen $(a_1, \cdots, a_k)$ entsprechen. Alle Blätter befinden sich auf Tiefe n. Knoten auf der Tiefe k haben m_{k+1} Kinder, die mit $0, 1, \cdots, m_{k+1} - 1$ bezeichnet sind, s. Abbildung 1.

Wenn wir die Blätter von links nach rechts durchlaufen, bekommen wir die Tupel in der **lexikographischen Reihenfolge**.

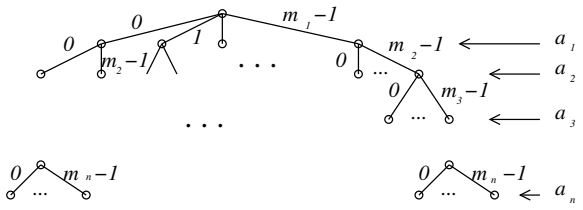


Abbildung 1: Suchbaum für n-Tupel zur (variablen) Basis m.

Konstruktion des reflektierten m-ärer Gray-Codes: In dem oben definierten Baum für $d = 1, 2, \ldots, n-1$: (in dieser Reihenfolge) Wähle jeden zweiten Knoten auf Tiefe d und kehre die Reihenfolge seiner Kinder um.

Wenn wir jetzt die Blätter von links nach rechts durchlaufen, bekommen wir die n-Tupel in der Reihenfolge, bei der zwischen zwei benachbarten Elementen nur eine Stelle sich ändert und zwar um höchstens 1.

Beispiel 3.1. $n = 2, m = 10$

	020	190	⋮
	021		109
000	⋮	⋮	
001		199	
⋮	029	189	⋮
	039		
009	038	⋮	100
019			200
018	⋮	180	⋮
⋮	030099	170	
			209
010	⋮	⋮	
	090	179	⋮

4 t-Teilmengen

Wir betrachten nun t-Teilmengen von $S = \{0, 1, \cdots, n-1\}$, d.h. Kombinationen von t Elementen aus n. Anzahl dieser Teilmengen ist $\binom{n}{t}$.

Wir werden folgende Darstellungen einer Teilmenge $T = \{c_1, c_2, \ldots, c_t\} \subset S$ mit $0 \leqslant c_1 < \cdots < c_t \leqslant n$ verwenden:

1. als Bitfolge: $(a_0, \ldots, a_{n-1})$ mit $a_i = 1 \Leftrightarrow i \in T$, wobei die Bitfolge genau t Einsen enthält;
2. als sortierte Liste oder Tupel $(c_1, c_2, \ldots, c_t)$,
3. als absteigend sortierte Liste: $(c_t, c_{t-1}, \ldots, c_1)$.

Beispiel 4.1. Für $n = 6, t = 3$ sei $T = \{2, 4, 5\}$. Dann können wir T als Bitfolge: (001011), sortierte Liste: 245, oder absteigend sortierte Liste: 542, darstellen.

Definition 4.1 (Lexikographische Reihenfolge). Wenn wir die Teilmengen als sortierte Listen betrachten und die Listen lexikographisch sortieren, bekommen wir die **lexikographische Reihenfolge auf den Teilmengen**.

Definition 4.2 (Kolexikographische Reihenfolge (colex)). Die Elemente der Teilmengen sind absteigend geordnet, und die resultierenden Folgen werden lexikographisch sortiert.

4.1 Lexikographische Reihenfolge

Nachfolgerbestimmung bei lexikographischen Erzeugung Sei $T = \{c_1, c_2, \ldots, c_t\}$ als sortierte Folge $(c_1, c_2, \ldots, c_t)$ mit $0 \leqslant c_1 < \cdots < c_t \leqslant n-1$ gegeben. Bestimme den Nachfolger von T in lexikographischen Reihenfolge.

Idee: Suche von rechts nach links die erste Stelle, die erhöht werden kann, erhöhe sie und setze die nachfolgenden Stellen auf ihren kleinstmöglichen Wert.

Beispiel 4.2. Für $n = 6$ und $t = 3$:

012	123
013	124
014	125
015	134
023	135
024	145
025	234
034	235
035	245
045	345

Höchste Werte, die an der Stelle j angenommen werden können: $c_j \leqslant j+n-t-1$

Die kleinstmöglichen Werte nachdem c_j gesetzt wurde sind für nachfolgenden Stellen dann $c_{j+1} := c_j + 1$.

Das erste Element, d.h. die kleinste Teilmenge: $c_j := j - 1$ für $j = 1, \cdots, t$.

Algorithm 3 Nachfolger in lexikographischen Reihenfolge für t-Teilmengen

```
1: j = t
2: while c_j = j + n − t − 1 do
3:    if j = 1 then
4:       HALT
5:    end if
6:    j=j-1
7: end while
8: c_j = c_j + 1
9: while j < t do
10:   c_{j+1} = c_j + 1
11:   j = j + 1
12: end while
```

Rangbestimmung

Beispiel 4.3. Wir wählen 134. Vor 134 kamen die Elemente:

$0*$: 2-Teilmengen von $\{1, 2, 3, 4, 5\}$
$12*$: 1-Teilmengen von $\{3, 4, 5\}$

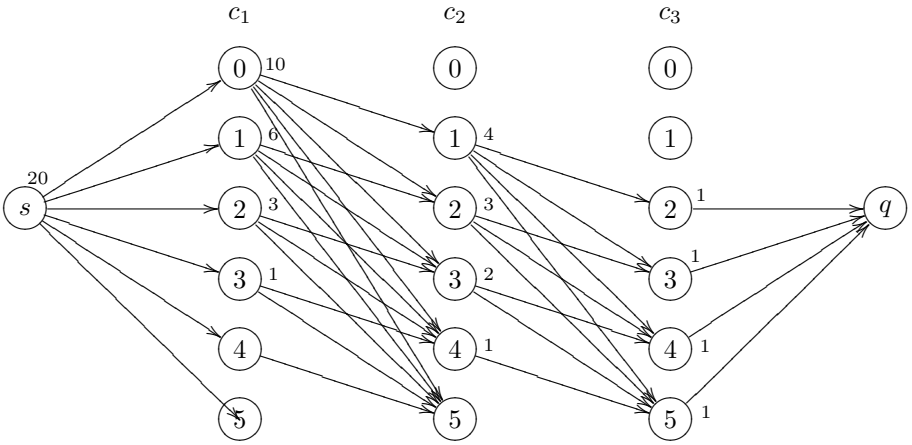
Also $\binom{5}{2} = 10$ und $\binom{3}{1} = 3$ Elemente.

Beispiel 4.4. Wählen $257, n = 10$, die kleinerren Teilmengen sind dann die folgenden:

$0*$	$\binom{9}{2}$
$1*$	$\binom{8}{2}$
$23*$	$\binom{6}{1}$
$24*$	$\binom{5}{1}$
256	$\binom{3}{0}$

Alternative Vorgehensweise: Wir erstellen einen Graphen. Die Knoten sind in Schichten angeordnet, die den Stellen entsprechen. Kanten führen von einer Schicht zur nächsten. „Lösungen“ entsprechen Wegen im Graphen.

Beispiel 4.5. $n = 6$, $t = 3$



Für jeden Knoten kann man die Anzahl der Wege bestimmen, die zum Zielknoten q führen. Dabei geht man Schicht für Schicht von hinten nach vorne vor. Der Wert eines Knotens ist die Summe der Werte seiner Nachbarn in Vorwärtsrichtung.

Benötigte Zeit = O(Anzahl der Kanten), im Beispiel: $O(t \cdot n^2)$ Kanten, die Abarbeitung kann aber in $= (t \cdot n)$ Zeit erfolgen.

Die Wege, die lexikographisch kleiner sind als ein gegebener Weg W, sind die Wege, die von W nach oben abtrennen.

Rangbestimmung für einen Weg W: Für jede Kante (u, v) von W

- betrachte alle Kanten (u, v'), die „höher“ als (u, v) liegen, d.h. $(v' < v)$ und
- summiere die Anzahl der Wege nach q, die in v' beginnnen.

Somit ist der Rang gleich der Summe der Werte auf den Kanten des Weges.

In einer Vorbereitungsphase kann man für jede Kante des Graphen die oben erwähnte Summe speichern.

Inverse Randbestimmung: Wir suchen einen geeigneten Wegvon links nach rechts unter Zuhilfenahme der gespeicherten Zahlen. Statt der induktiven Erzeugung können die Zahlen auch durch geschachtelte Schleifen erzeugt werden.

Algorithm 4 Inverse Rangbestimmung

```
1: for c_1 = 0 TO 3 do
2:    for c_2 = c_1 + 1 TO 4 do
3:       for c_3 = c_2 + 1 TO 5 do
4:          verarbeite(c1, c2, c3)
5:       end for
6:    end for
7: end for
```

Es müssen also t ineinander verschachtelte Schleifen implementiert werden. Wenn t eine Variable ist, kann man Rekursion verwenden, wobei die Reikursionstiefe ein Parameter ist.

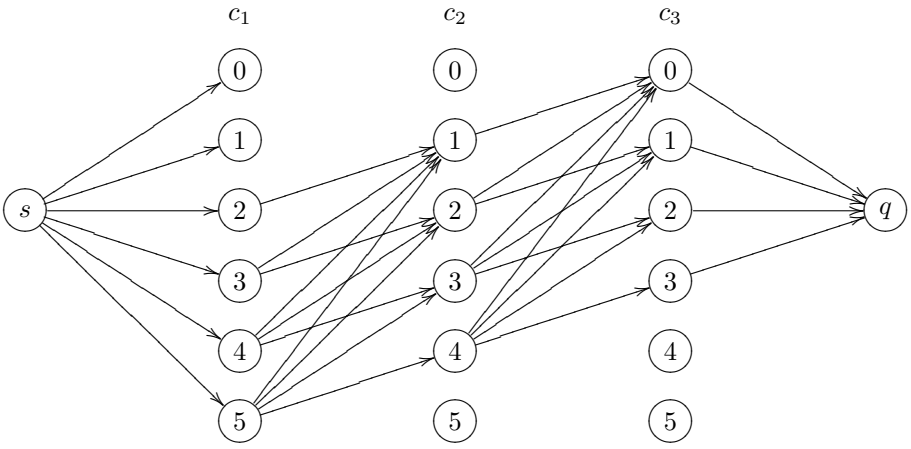
Algorithm 5 erzeuge

```
1: erzeuge(j)
2: global c1, ··· , ct, t, n
3: if j = t then
4:    besuche(c1, ··· , ct)
5: else
6:    for c_j + 1 TO j − n + t do
7:       erzeuge(j + 1)
8:    end for
9: end if
10: —
11: c_{−1} = −1
12: erzeuge(0)
```

4.2 Kolexikographische(colex-) Reihenfolge

Betrachten wir wieder die Graphen-Darstellung anhand eines Beispiels:

Beispiel 4.6. $n = 6$, $t = 3$



Hierbei ergibt sich der Wert eines Knotens c_j in Spalte j durch Anzahl der

$(j-1)$-Teilmengen von $\{0,1,\cdots,c_j-1\} = \binom{c_j}{j-1}$.

Die ersten 20 3-Elementigen Teilmengen in kolexikographischen Reihenfolge:

210	510
310	520
320	521
321	530
410	531
420	532
421	540
430	541
431	542
432	543

Beispiel 4.7. Bestimmen wir den Rang von 431: kleiner sind die Elemente

$\left.\begin{matrix}1*\\2*\\3*\end{matrix}\right\}$ 3-Teilmengen von $\{0,\cdots,3\}$,

$\left.\begin{matrix}41*\\42*\end{matrix}\right\}$ 2-Teilmengen von $\{0,1,2\}$

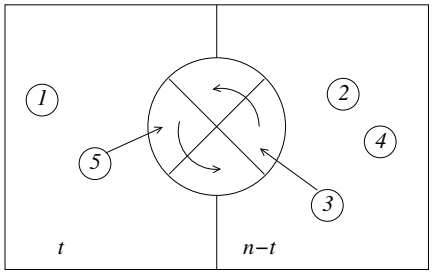
$\left.\begin{matrix}420\end{matrix}\right\}$ 1-Teilmengen von $\{0\}$

Der Rang einer Menge $\{c_t, c_{t-1}, \cdots, c_1\} = \sum_{j=1}^{t} \binom{c_j}{j}$. Der Rang ist unabghängig von n.

Satz 4.3. Sei t fest. Jede natürliche Zahl $N \geqslant 0$ läßt sich eindeutig in der Form $\sum_{j=1}^{t} \binom{c_j}{j}$ mit $c_t > c_{t-1} > \ldots > c_1 \geqslant 0$ darstellen. Dann nennen wir dies das kombinationsche Zahlsystem vom Grad t.

4.3 Drehtür-Reihenfolge (alternierendes kombinatorisches Zahlsystem)

Wir betrachten wieder die t-Teilmengen aus n Elementen. Bei einer Drehtür-Reihenfolge wird zwischen den zei benachbarten Teilmengen immer nur *ein* Element gegen ein anderes ausgetauscht. Dies stellt die kleinstmögliche Änderung dar.



Die t-elementigen Teilmengen im Standard-Gray-Code (binärer reflektierter) bilden eine Drehtür-Reihenfolge!

Der binärer reflektierte Gray-Code als Bit-Folge: $G_n = 0G_{n-1}, (1G_{n-1})^R$, nehmen wir davon die Teilfolge der Wörter mit t Einsen und bezeichnen sie mit D_n^t.

Satz 4.4. $D_n^t = 0D_{n-1}^t, (1D_{n-1}^{t-1})^R$ ist eine Drehtür-Reihenfolge für die t-elementigen Teilmengen von $\{0, \cdots, n-1\}$, die mit $0^{n-t}1^t$ beginnt und mit $10^{n-t}1^{t-1}$ aufhört. (Beweis durch Induktion).

Alternative Darstellung als sortierte Folge: $c_t > c_{t-1} > \cdots > c_1, 0 \leqslant c_i \leqslant n-1$

$$D_n^t = D_{n-1}^t, \left((n-1)D_{n-1}^{t-1}\right)^R$$

Beispiel 4.8. $t = 1$: $\underbrace{0, 1, 2, 3}_{D_4^1}, 4, \cdots$

$t = 2$: $\underbrace{\underbrace{\underbrace{10}_{D_2^2}}_{D_3^2}, 21, 20, 32, 31, 30, 43, 42, 41, 40, 54}_{D_4^2}$

$t = 3$: $\underbrace{\underbrace{\underbrace{210}_{D_3^3}}_{D_4^3}, 320, 321, 310, 430, 431, 432, 420, 421, 410}_{D_5^3}$

Das erste Element (D_n^t) in jeder Folge sind aufsteigend sortiert, das zweite absteigend, das dritte aufsteigend etc.

In D_n^t ist c_t aufsteigend sortiert. Durch Induktion ergibt sich: Die Elemente von D_n^t mit festem $c_t, c_{t-1}, \cdots, c_{t-i+1}$ kommen in aufsteigender / absteigender Reihenfolge von c_{t-i} vor wenn i gerade / ungerade ist.

Algorithm 6 Erzeugung mit geschachtelten Schleifen

```
1: for c_t = t − 1 TO n − 1 do
2:   for c_{t−1} = c_t − 1 DOWNTO t − 2 do
3:     for c_{t−2} = t − 3 TO c_{t−1} do
4:       ...
5:     end for
6:   end for
7: end for
```

Rangbestimmung Die Rangbestimmung ist nur abhängig von t, aber unabhängig von n.

$$\begin{aligned} Rang(c_1) &= c_1 = \binom{c_1+1}{1} - 1 \\ Rang(c_2c_1) &= \binom{c_2+1}{2} - Rang(c_1) - 1 \\ Rang(c_3c_2c_1) &= \binom{c_3+1}{3} - Rang(c_2c_1) - 1 \end{aligned}$$

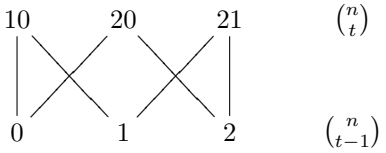
allgemein:

$$\begin{aligned} Rang(c_t c_{t-1} \cdots c_1) &= \binom{c_t+1}{t} - 1 - Rang(c_{t-1} \cdots c_1) \\ &= \binom{c_t+1}{t} - \binom{c_{t-1}+1}{t-1} \pm \ldots \begin{cases} +\binom{c_1+1}{1} - 1, & t \text{ unger.} \\ -\binom{c_1+1}{1}, & t \text{ gerade} \end{cases} \\ &= \sum_{i=1}^{t} (-1)^{t-i} \binom{c_i+1}{i} - 1 \cdot (t \mod 2) \end{aligned}$$

Definition 4.5 (Das alternierende kombinatorische Zahlsystem der Ordnung t). (t fest). Jede natürliche Zahl $n \geqslant 0$ läßt sich eindeutig in der Form $N = \binom{f_t}{t} - \binom{f_{t-1}}{t-1} \pm \ldots \begin{cases} +\binom{f_1}{1} - 1, & t \text{ unger.} \\ -\binom{f_1}{1}, & t \text{ gerade} \end{cases}$, mit $f_t > f_{t-1} > \cdots > f_1 \geqslant 1$, schreiben.

Offenes Problem: kann man die t- und die $(t-1)$-elementigen Teilmengen einer $n = (2t-1)$-elementigen Menge so aufzählen, dass in jedem Schritt nur $\underline{1}$ Element hinzugefügt oder weggenommen wird?

Beispiel 4.9. $t = 2$:



Bis $t \leqslant 15$ ist eine Lösung bekannt.

5 de-Bruijn-Zyklus

Definition 5.1 (de-Bruijn-Zyklus). Ein de-Bruijn-Zyklus der Ordnung n über einem m-elementigen Alphabet ist eine Folge $a_0 a_1 \ldots a_{m^n-1}$ mit $0 \leqslant a_i \leqslant m-1$, so dass jedes n-stellige Wort $b_1 b_2 \cdots b_n$ genau einmal als $a_{i+1} a_{i+2} \cdots a_{i+n}$ vorkommt (Indizies $i+k$ modulo m^n).

Beispiel 5.1. $m = 4, n = 2$: 0011022120331323

Satz 5.2. Für jedes m und n existiert ein de-Bruijn-Zyklus. (Beweis: Eulerscher Kreis im Wortgraphen)

Erzeugung durch Schieberegisterfolgen

$$a_k = \begin{cases} c_1 a_{k-n} + c_2 a_{k-n+1} + \ldots + c_n a_{k-1} (\mod m) \\ c_1 \text{ , wenn } (a_{k-n}, a_{k-n+1}, \ldots, a_{k-1}) = (0, 0, \ldots, 0) \end{cases}$$

für feste Koeffizienten $c_1, c_2, \ldots, c_n$. Beginne mit $0^n = 000 \ldots 0$.

Beispiel 5.2. $n = 4,\ m = 2,\ (c_1, c_2, c_3, c_4) = (1, 1, 0, 0) : a_k = a_{k-4} + a_{k-3}$ $\Rightarrow$ 0000100110101111.

Betrachte Übergang von $(a_{k-1}, a_{k-2}, \ldots, a_{k-1})$ zu $(a_k, a_{k-1}, a_{k-2}, \ldots, a_{k-n+1})$ als Multiplikation des Polynoms $P(x) = x_n - c_n x^{n_1} - \ldots - c_2 x - c_1$ mit dem Polynom $a_{k-1} x^{n-1} + a_{k-2} x^{n-2} + \cdots + a_{k-n}$ modulo x^n.

Definition 5.3 (irreduziebles Polynom). Ein *irreduziebles Polynom* ist ein Polynom, das sich nicht als Produkt von zwei Polynomen mit kleinerem Grad ($\geqslant 1$) darstellen lässt.

Für die meisten n gibt es ein irreduzibles Polynom in der Form $x^n + x^t + 1$ oder $x^n + x^s + x^t + x^{s+t} + 1$.

Definition 5.4 (primitives Polynom modulo m). $P(x)$ ist ein *primitives Polynom* modulo m wenn $x^i \mod P(x) \neq 1$ für $i = 1, 2, \cdots, m^n - 2 \Leftrightarrow$ wenn $x^i \mod P(x)$ alle von 0 verschiedenen Polynome vom Grad $\leqslant n-1 \mod m$ durchläuft.

Satz 5.5. Sei $P(x) = x^n - c_n x^{n-1} - \cdots - c_2 x_1 - c_1$ ein primitives Polynom mod m, und m eine Primzahl, dann liefert der obige Algorithmus einen m-ären de-Bruijn-Zyklus der Länge m^n wenn mit $\underbrace{(0, 0, \cdots, 0)}_{n}$ begonnen wird.

Betrachte den Ring der Polynome in einer Variablen mit Koeffizienten in $\mathbb{Z}_m$, $R = \mathbb{Z}_m[x]$.

Man rechnet modulo $P(x)$: Polynome, die sich nur durch ein Vielfaches von $P(x)$ unterscheiden, werden als gleich betrachtet.

Beispiel 5.3. $P(x) = x^5 + x^2 + 1,\ Q(x) = x^4 + x^2$

- Berechne $Q(x) \cdot x \mod (P(x))$. $Q(x) \cdot x = x^5 + x^3$
  eliminiere x^5: $\underset{x^3+x^2+1}{=x^5+x^2+1}$
- Berechne $Q(x) \cdot Q(x) (\mod P(x))$
  $(x^4 + x^2)(x^4 + x^2) = x^8 + x^6 + x^6 + x^4 = x^8 + x^4$
  eliminiere x^8: addiere $P(x)x^3$ $\underset{x^5+x^4+x^3}{=x^8+x^5+x^3}$
  eliminiere x^5: addiere $P(x)$ $\underset{x^4+x^3+x^2+1}{=x^5+x^2+1}$

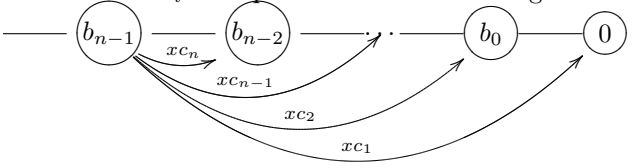
(Polynomdivision durch $P(x)$, behalte den Rest.)

Allgemein:

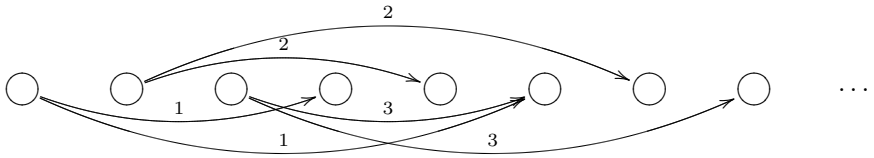
$$\begin{aligned} \text{Sei} \qquad x^i &= b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_1 x + b_0 \qquad (1) \\ x^{i+1} &= b_{n-1}x^n + b_{n-2}x^{n-1} + \cdots + b_1 x^1 + b_0 x \qquad (2) \end{aligned}$$

- Fall 1: $b_{n-1} = 0$: fertig
- Fall 2: $b_{n-1} \neq 0$: substituiere $b_{n-1}P(x)$, also $x^{i+1} - b_{n-1}x^n + b_{n-1}c_n x^{n-1} + \cdots + b_{n-1}c_2 x + b_{n-1}c_1 = (b_{n-2} + b_{n-1}c_n)x^{n-1} + \cdots + (b_0 + b_{n-1}c_2)x + (b_0 + b_{n-1}c_1) = b'_{n-1}x^{n-1} + \cdots + b'_1 x + b'_0$

Zuerst werden die Koeffizienten zyklisch verschoben. Dann wird b_{n-1} mit dem Koeffizienten c_i multipliziert und zu den übrigen Koeffizienten addiert.



Beispiel 5.4. $c_1 = 1, c_3 = 1$, Rest $= 0$



Die Folge der Ziffern b_{n-1}, die jeweils nach jeder Multiplikation mit x ausgespuckt werden ist die de-Bruijn-Folge, die von dem rekursiven Algorithmus berechnet wird.

Algorithm 7 Erzeigen einer de-Bruijn-Folge

1: Beginne mit $Q(x) = 1$, $i = 0$
2: **while** $i < m^n - 1$ **do**
3: $Q(x) = b_{n-1}x^{n-1} + \ldots + b_0$
4: print b_{n-1}
5: $Q(x) = Q(x) \cdot x \mod P(x)$
6: $i++$
7: **end while**

Bemerkung 5.6. Auf $\mathbb{C}$ sind nur lineare Polynome $(x - c)$ irreduziebel. Irreduzieble Polynome auf $\mathbb{R}$ sind lineare Polynome: $(x - c)$ und quadratische Polynome mit zwei komplexen Lösungen: $ax^2 + bx + c$ mit $b^2 - 4ac < 0$.

Auf $\mathbb{Z} \mod 2$ irreduzible Polynome vom

- Grad 1
  - $\underline{x}$
  - $\underline{x + 1}$
- Grad 2
  - $x^2 = x \cdot x$
  - $x^2 + x = x(x + 1)$
  - $x^2 + 1 = (x + 1)(x + 1)$
  - $\underline{x^1 + x + 1}$
- Grad 3
  - $(x^2 + x + 1)(x + 1) = x^3 + 1$
  - $(x + 1)^3 = x^3 + x^2 + x + 1$
  - $(\cdots)x = x^3 + ?x^? + ?x^? + 0$
  - $\underline{x^3 + x^2 + 1}$
  - $\underline{x^3 + x + 1}$

Es gibt $\frac{\varphi(m^n-1)}{n}$ irreduzible Polynome vom Grad $n \mod m$, wobei

$$\varphi(x) = \#\{i | 1 \leqslant i \leqslant m^n - 1 | ggT(i, m^n - 1) = 1\}$$

ist Euler'sche Funktion, Anzahl der Teiler.

Satz 5.7. Wenn m Primzahl und $P(x)$ ein primitives Polynom, dann bilden $\mathbb{Z}_m[x] \mod P(x)$ einen Körper mit m^n Elementen. Diese Körper sind die einzigen endlichen Körper.

5.1 de-Bruijn-Zyklen aus Primwörtern

Definition 5.8 (Primwort). Ein *Primwort* ist ein Wort, das lexikographisch echt kleiner ist als alle seine zyklischen Verschiebungen. $\Leftrightarrow$ das lexikographisch kleiner ist als alle seine echten Suffixe.

Lemma 5.9. periodische Wörter sind keine Primwörter.

Wenn ein Wort w der Länge n nicht periodisch ist, sind alle n zyklischem Verschiebungen verschieden, und es gibt unter ihnen genau ein Primwort.

Ein Wort sei periodisch mit Periode $d : x = a^{\frac{n}{d}}$ mit $|a| = |d|$, $d|n$. Bei den zyklischen Verschiebungen erhält man d verschiedene Wörter. Bei genau einem dieser Wörter bilden die ersten d Symbole ein Primwort.

Satz 5.10. Sei L_k die Anzahl der Primwörter der Länge k (über einem m-elemntigem Alphabet). Dann gilt: $\sum_{d|n} L_d \cdot d = m^n$

Beweis. m^n ist Anzahl der Wörter x der Länge n. Die Periode von x sei d. Die zyklischen Verschiebungen von x bilden eine Klasse mit d Wörtern. Jede solche Klasse entspricht einem Primwort der Länge d.

Satz 5.11 (Kleiner Farmat'scher Satz). Sei n prim $\Rightarrow n|(m^n - m)$

Beweis.

$$\begin{aligned} L_n \cdot 1 - L_n \cdot n &= m^n \\ L_n \cdot n &= m^n - m \quad \Rightarrow \quad n|m^n - m \end{aligned}$$

Erzeugen alle Primwörter der Länge $\leqslant n$.

Beispiel 5.5. $m = 3$, $n = 4$

0|000	0102|	0212|	2|222
0001|	011|0	022|0	1222|
0002|	0111|	0221|	
001|0	0112|	0222|	
0011|	012|0	1|111	
0012|	0121|	1112|	
002|0	0122|	112|1	
0021|	02|02	1122|	
0022|	021|0	12|12	
01|01	0211|	122|1	

Alles, was vor „|“ steht ist ein Primwort der entsprechenden Länge. Nun schreiben wir alle Primwörter der Länge $d|n$ hintereinander: $0, 0001, 0002, 0011 \ldots$. Das bildet eine de-Bruijn-Folge der Länge m^n. (ohne Beweis)

Definition 5.12 (Ordnung). Betrachten wir x^i. Das kleinste $k > 0$ mit $x^k \mod P(x) = 1$ heißt die *Ordnung* von $x \mod P(x)$.

Wenn k die Ordnung von $P(x)$ ist, dann $x^i \mod P(x) \equiv 1 \Leftrightarrow k|i$.

$P(x)$ ist ein primitives Polynome.
$\Leftrightarrow$ Ordnung von $x \mod P(x)$ ist $m^n - 1$
$\Leftrightarrow$ $x^{m^n-1} \mod P(x) = 1$ und für alle primen Teiler $d|m^{n-1}$ gilt $x^{(m^n-1)/d} \mod P(x) \neq 1$

Ein primitives Polynom muss irreduzibel sein, Die Anzahl der primitiven Polynome ist $\varphi(m^n - 1)/n$.

$\sum_{d/n} L_d \cdot d = m^n$, $n = p$ Primzahl.

$\underbrace{L_1}_{=m} + L_p \cdot p = m^p$

6 Kontingenztafeln

Kontingenztafeln finden Anwendung in der Statistik

25	1	26
18	4	22
43	5	28

Verfahren zum Testen der Nullhypothese:

- Zähle alle Kontingenztafeln $T_1, T_2, \cdots$ mit den vorgegebenen Zeilen- und Spaltensummen auf.
- Summiere die Wahrscheinlichkeiten aller Tafeln, die kleine sind als die Wahrscheinlichkeit der beobachteten Tagel plus $\frac{1}{2}$ (Die Wahrscheinlichkeiten, die gleich der beobachteten Tafel sind.
- Wenn die berechnete Zahl kleiner als eine vorgegebene Schranke ist (zb $p = 0.05$ oder $p = 0.01$)

verwirf die Nullhypothese.

Definition 6.1 (primitives Polynom). Grad n, gerechnet wird $\mod m, m = 2$

$P(x) \in \mathbb{Z}_n, \forall i = 1 \cdots 2^n - 1, x^i \mod P(x) \not\equiv 1$

Beispiel 6.1. $m = 2, n = 4$

$x^4 + 1 : (x + 1)|(x^4 + 1)$

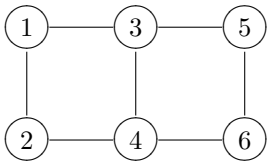
$P(x) = x^4 + x + 1; x[v](x + 1)[v]x^2 + x + 1[v]$

7 Abzählen mit dynamischer Programmierung bzw. Übergangsmatrizen

Gegeben sei ein $(2 \times n)$-Gitter. Gesucht sei die Anzahl der Spannbäume (oder die Anzahl der zusammenhängenden Teilgraphen, Anzahl der hamiltonschen Kreise, Anzahl der vollständigen Paarungen etc.).

Definition 7.1 (Spannbaum). Ein Spannbaum ist ein Baum über alle Knoten eines Graphen .

Beispiel 7.1. (2×3):



Teilprobleme:

- $(2 \times k)$-Gitter für alle kleineren k
- alle kreisfreien Graphen (auch unzusammenhängende), die zu einem Spannbaum ergänzt werden können: Jede Komponente muss einen von den beiden rechtesten Knoten enthalten.

7.1 Aufbau

Die Leitern lassen sich rekursiv aufbauen: Ein $(2 \times n)$-Gitter entsteht aus einem $\big(2 \times (n-1)\big)$-Gitter durch Hinzufügen von 2 Knoten und 3 Kanten auf der rechten Seite.

Der „Zustand“ einer Teillösung umfasst alle Informationen, die notwendig sind, um festzustellen, auf welche Art die Teillösung nach rechts zu einer vollständigen Lösung erweitert werden kann.

In diesem Beispiel gibt es zwei Zustände:

- A: die beiden rechten Knoten sind in einer Komponente
- B: die beiden rechten Knoten sind in verschiedenen Komponenten

$x_A^n, x_B^n :=$ # aufspannenden kreisfreien Teilgraphen des $(2 \times n)$-Gitters im Zustand A bzw. B, wo jede Komponente einen der beiden rechten Knoten enthält.

7.1.1 Beispiel

$x_A^1 = 1, x_b^1 = 1$ $x_A^2 = 4, x_b^2 = 3$

Den rekursiven Aufbau der Leiter kann an nun in einer Übergangsmatrix (Transformationsmatrix) dargestellt werden. $T = \begin{pmatrix} + & A & B \\ A & 3 & 2 \\ b & 1 & 1 \end{pmatrix}$ $(x_A^{n+1}, x_B^{n+1}) = (x_A^n, x_B^n) \cdot$

$T = (\underbrace{3x_A^n + 1x_B^n}_{x_A^n}, \underbrace{2x_A^n + 1x_B^n}_{x_B^{n-1}})$

Lösung des eigentlichen Problems $= x_A^n = (1,1)T^{n-1}\binom{1}{0}$

7.2 Variante: das $(2 \times n)$-Band

$V = \{1,2\} \times \{1,2,\cdots,n\}$

$E = ((1,i),(2,i))\,((1,i),(1,(i \mod n)+1))\,,((2,i),(2,(i \mod n)+1))$

Wir lösen das Problem des $(2 \times n)$-Bandes mit Hilfe der $(2 \times n)$-Leiter

Teillösungen: kreisfreie spannende Untergraphen des Gitters, wo jede Komponente mindestens einen der „Randknoten“ $P = (1,1), Q = (2,1), R = (1,n)$ oder $S = (2,n)$ enthält.

Zustand: Welche der vier Randknoten gehören zu einer Menge?

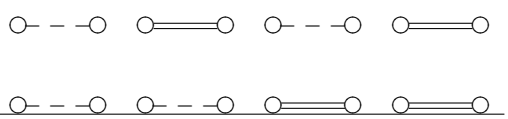
$$\left\{P|Q|R|S\right\} \quad \overline{\left\{PS|RQ\right\}}$$
$$\left\{PQ|R|S\right\} \quad \left\{PQR|S\right\}$$
$$\left\{PR|Q|S\right\} \quad \left\{PQS|R\right\}$$
$$\left\{PS|R|Q\right\} \quad \left\{PRS|Q\right\}$$
$$\left\{QR|P|S\right\} \quad \left\{QRS|P\right\}$$
$$\left\{QS|R|P\right\} \quad \left\{PQRS\right\}$$
$$\left\{RS|P|Q\right\}$$
$$\left\{PQ|RS\right\}$$
$$\left\{PR|QS\right\}$$

Für $\left\{PS|RQ\right\}$ gibt es keine Lösungen, weil sich die beiden Komponenten „kreuzen“ müssten. Damit bleiben 14 Zustände.

$x^1_{\{PR|QS\}} = 1, x_{\{PQRS\}} = 1, x^1_Z = 0$ sonst $, x^{n+1} = x^n \cdot T$

7.3 Gesamtlösung

Der Übergang von der $(2 \times n)$-Leiter zum $(2 \times n)$-Band lässt sich durch folgende Tabelle darstellen:

	
$\{PQRS\}$	1 - - -
$\{P\|RSQ\}$	- 1
$\{PQ\|RS\}$	- 1 1
$\{PSQ\|R\}$	- 1 -
$\{Q\|PRS\}$	1
$\{S\|PQR\}$	1
$\{RS\|P\|Q\}$	1
$\{RQ\|P\|S\}$	1
$\{PS\|R\|Q\}$	1
$\{PQ\|R\|S\}$	1

Ergebis = $\sum_{\text{Zustände}}$ Anzahl der möglichen Kombinationen.

Statt Zählproblemen kann man auf diese Art auch Optimierungsproblme lösen.

Man merkt sich für jeden Zustand die optimale Teillösung x_A^n, x_B^n

7.4 Verallgemeinerung auf das $(m \times n)$-Gitter

Die Anzahl der Zustände wächst im allgemeinen exponentiell zu m, die Anzahl der Schritte hängt von n nur linear ab. $O(2^{O(m)} \cdot n)$ Um die Zustände zu bestimmen, muss man alle Möglichkeiten mit einer bestimmten kombinatorischen Struktur aufzählen (z.B. alle Partitionen aeiner Menge $\{P, Q, R, S\}$).

8 Polyominos

Definition 8.1 (Polyominos). Polyominos bestehen aus einzelnen Zellen, die sich an den Kanten berühren. Berührungen an den Ecken reichen nicht aus. Ein mögliche Darstellung sind Punkte in einem Gitter, wobei die Mittelpunkte der Zellen auf einem Schnittpunkt im Raster liegen. Also entsprechen Polyominos zusammenhängenden endlichen Knotenmengen des ebenen Gitters $\mathbb{Z} \times \mathbb{Z}$ mod Translation.

Ein anderer Name für Polyominos ist *animals*.

Bemerkung 8.2. Manchmal möchte man man Polyominos ausschließen, die Löcher enthalten.

A_n = #Polyominos mit n Zellen. Wir betrachten nur „feste Polyominos“ ohne Drehung und Spiegelung.

$n = 2$ Dominos
$n = 3$ Triominos
$n = 4$ Tetrominos

A_n ist bis $n = 56$ bekannt. Dabei ist $\frac{A_{n+1}}{A_n} \approx 4, \lim_{n\to\infty} \frac{A_{n+1}}{A_n} = \lambda^*$ existiert und wird Kla-Konstante. Der Bereich konnte auf $3,98 \leqslant \lambda^* \leqslant 4,6$ eingeschränkt werden.

8.1 Bestimmung der Anzahl (nach Conway)

Es existieren diverse Verfahren zur Bestimmung der Abzahl, entwickelt von Conway, Jensen, Guttmann. Wir betrachten hier Ersteren.

Wir berechnen Polyominos in einem Rechteck der Breite B. Dieses Rechteckt wird dabei von links nach rechts aufgebaut, indem einzelne Zellen von oben nach unten am rechten Rand hinzugefügt werden, bis eine ganze Spalte voll ist.

Beim Hinzufügen muss der Zustand gemerkt werden, ob es sich schon um ein Polyomino handelt. Fügt man eine Zelle hinzu, beachtet man, ob das betreffende Feld schon belegt ist.

Definition 8.3 (Teilpolyomino). Ein Teilpolyomino ist ein Teil eines Polyominos links von der Trennlinie, die den rechten Rand bestimmt. Es besitzt folgende Eigenschaften:

- Zusammenhang ist nicht gefordert
- jede Komponente muss eine Zelle enthalten, die am rechten Rand anstößt, damit es zu einem Polyomino vervollständigt werden kann.

Die *Randzellen* seien von 1 bis B durchnummeriert. Der Knick in der Trennlinie sei unter Zelle i. Also läßt sich der Zustand beschreiben: Welche der Randzellen sind belegt und welche dieser Zellen gehören zu einer Komponente? (eine Partition der Randzellen) und wo ist der Knick? Jeder Zustand hat bis zu 2 Nachfolgezustände:

- die neue Zelle ist belegt
- die neue Zelle ist frei (ist verboten, wenn die Zelle links davon nur alleine in einer Komponente enthalten ist und somit von der neuen Zelle verdeckt werden würde)

$X_{j,n,i,s}$ = Anzahl der Teilpolyominos in einem Bereich mit j vollen Spalten und i Zellen in der letzten Spalte, mit n Zellen im Zustand s (Partition der Randzellen).

```
************ ^
*           * | i
*           * |
*          ** V
*          *#
*          *
************
<---- j --->
```

Im obigen Algorithmus fehlen:

- X auf 0 inititialisieren.
- Behandlung der 1. Spalte

Algorithm 8 Abzählen von Teilpolyominos

```
1: for j = 0 TO ... do
2:   for i = 0 TO B − 1 do
3:     Bestimme s_1 = Nachf_1(s) und s_0 = Nachf_0(s)
4:     Addiere X_{j,n,i,s} zu X_{j,n+1,i+1,s_1}
5:     if s_0 existiert then
6:       Addiere X_{j,n,i,s} zu X_{j,n,i+1,s_0}
7:     end if
8:   end for
9: end for
```

- Bestimmen des Endergebisses: zähle nur Polyominos $X_{j,n,B,s}$ mit einer Komponente → Polyominos im Rechteck $B \times (j+1)$, die am linken und rechten Rand anstoßen.
- Übergang zur nächsten Spalte: $X_{j,n,B,s} \to X_{j+1,n,0,s}$

Bemerkung 8.4. Ein Polyomino, das b Zeilen belegt wird $(B - b + 1)$-mal gezählt.

$Y_{b,j}$ sei die Anzahl der Polyominos mit Breite b (und Länge j), Ergebnis $= Y_B + 2Y_{B-1} + 3Y_{B-2} + \cdots BY_1$.

Wenn man $Y_1, \cdots, Y_{B-1}$ kennt, dann kann man Y_B bestimmen. Dazu läßt man den Algorithmus für $B = 1, 2, 3, \cdots$ laufen und bestimmt $Y_1, Y_2, Y_3, \cdots$

Ein Polyomino mit n Zellen hat Breite b und Länge l mit $n \geqslant l + b - 1$. Für $b > l$ drehen wir das Polyomino ($Y_{b,j} = Y_{j,b}$). Es genügt $b \leqslant \frac{n+1}{2}$ zu betrachten.

Beispiel 8.1.

$$\begin{aligned} A_7 &= Y_{7,1} \\ &+ Y_{6_1} + Y_{6,2} \\ &+ Y_{5_1} + Y_{5,2} + Y_{5,3} \\ &\vdots \\ &+ Y_{1,1} + Y_{1,2} + \cdots Y_{1,7} \end{aligned}$$

Algorithm 9 Abzählen von Teilpolyominos (Variante)

```
1: for i = do
2:   for j = 1 TO B do
3:     for ALL n, s do
4:       Bestimme die Menge Vorg_z(s) aller Zustände s', für die s =
         Nachf_z(s') ist. (z = 0, 1)
5:     end for
6:   end for
7: end for
```

Variante $X_{j,n,i,s} := \sum_{s' \in Vorg_0(s)} X_{j,n,i-1,s'} + \sum_{s' \in Vorg_1} X_{j,n-1,i-1,s'}$

$X_{j,*,i,x}$ ist der „alte“ Vektor, $X_{j,*,i+1,*}$ ist der „neue“ Vektor. Mann merkt sich nur jeweils einen alten und neuen Vektor, z.B. als Hash-Tabelle, indiziert durch den Zustand s. Für jedes s hat man ein Array, das durch n indiziert wird.

Definition 8.5 (pruning). Verbesserung des Speicherbedarfs durch Ausschließen von Zuständen.

Sei ein Zustand gegeben. Wieviel zusätzliche Zellen n' werden mindestens benötigt, um aus dem Teilpolyomino s ein gültiges Polyomino zu bilden. Dabei muss der Zusammenhang und mindestens die Länge L erreicht werden. Wenn $n + n' > N$ (A_N soll berechnet werden), dann brauch dieser Zustand nicht betrachtet zu werden.

Weiterhin brauchen nur die Polyominos gezählt zu werden, welche die volle Breite B haben. Dabei muss im Zustand gespeichert werden, ob das Teilpolyomino jeweils den oberen/unteren Rand schon berührt hat.

$x'_n = 3x_{n-1} + 5x_{n-2} + x_{n-3}$

$x'^{(n)} = Ax^{(n-1)}$ mit Anfangsvektor $x^{(0)}$

$x^{(n)} \approx v_1 \Lambda_1^n \cdot c_0$

Bemerkung 8.6. Vektor $x = \binom{x_1}{x_n} \geqslant 0 \Leftrightarrow \forall i : x_i \leqslant 0$

Matrix $A \leqslant 0$ bzw. $A > 0$

$||x||_\infty = \max_{i=1,\cdots,n} |x_i|$

$||x||_2 = \sqrt{\sum_i |x_i|^2}$

$||x||_1 = \sum_i |x_i|$

Satz 8.7 (Satz von Perron-Frobenius). Sei $A \in \mathbb{R}^{k \times k}, A \leqslant 0, A^{n_0} > 0$. Dann gilt für jeden beliebigen Startvektor $x_0 \in \mathbb{R}^k$ mit $x_0 \geqslant 0, x_0 \neq 0$

$\lim_{n\to\infty} \sqrt[n]{||A^n x_0||} = \lim_{n\to\infty} \frac{(A^{n+1}x_0)_i}{(A^n x_0)_i} = \lambda_1$ für $i = 1, \cdots, k$, λ_1 hängt nicht von x_0 ab.

$\lim_{n\to\infty} \frac{A^n x_0}{\lambda_1^n} = v_1 \cdot \text{Konst}$

$\lim_{n\to\infty} \frac{A^n x_0}{||A^n x_0||} = \frac{v_1}{||v_1||}$

v_1 hängt nicht von x_0 ab. $||\cdot||$ kann jede beliebige Norm sein.

Es gilt:

1. λ_1 ist der eindeutige Eigenvektor von A mit dem größten Betrag.
2. λ_1 hat Vielfachheit 1.
3. v_1 ist Eigenvektor zum Eigenwert λ_1.
4. v_1 ist der einzige Eigenvektor mit $v_1 \geqslant 0$.

5. λ_1 ist der PF-Eigenwert, v_1 ist der PF-Eigenvektor

Außerdem gilt: Es sei $\underline{\lambda}x \leqslant Ax \leqslant \overline{\lambda}x$ für einen beliebigen Vektor $x \geqslant 0$
$\Rightarrow \underline{\lambda} \leqslant \lambda_1 \leqslant \overline{\lambda}$

Bemerkung 8.8. Es gibt ein n_0 mit $A^{n_0} > 0$ (die Matrix heißt „primitiv“) $\Leftrightarrow$

1. Der Graph von A ist stark zusammenhängend (A irreversibel)
2. Der größte gemeinsame Teiler der Längen aller Kreise in G ist 1 (A aperiodisch)

Wenn G stark zusammenhängend ist und eine Schleife enthält ($a_{ii} > 0$) $\Rightarrow A$ ist primitiv.

Beweis $\lambda_1 := \sup\{\lambda | \exists u \geqslant 0, u \neq 0 : Au \geqslant \lambda u\}$

Das sup kann durch max ersetzt werden. Dabei genügt es, u mit $||u||_\infty = 1$ zu betrachten.

$\lambda_{\max}(u) = \max\{\lambda : Au \geqslant \lambda u\} = \min\limits_{i:u_i \neq 0} \frac{(Au)_i}{u_i}$ ist eine stetige Funktion auf $u \geqslant 0$ mit $||u||_\infty = 1$

$\lambda_1 = \max\{\lambda_{\max}(u) | u \geqslant 0, ||u|| = 1\}$

Wähle v mit $Av \geqslant \lambda_1 v$. Annahme: Es sei $Av = \lambda_1 v + v_{\text{Rest}}, v_{\text{Rest}} > 0$

$A^{n_0}Av = A^{n_0}\lambda_1 v + \underbrace{A^{n_0}v_{\text{Rest}}}_{>0}$

$A\underbrace{(A^{n_0}v)}_{u} > \lambda_1 \underbrace{(A^{n_0}v)}_{u}$

Mann kann λ_1 ein bisschen vergrößern, so dass $Au > \lambda_1 u$ immernoch gilt. Widerspruch.

$v > 0\ \underbrace{A^{n_0}}_{>0} v = \lambda^{n_0} v \Rightarrow v > 0$

Analog: $\lambda_1 \inf\{\lambda | \exists \geqslant 0 : Au \leqslant \lambda u\}$

$v = \operatorname{diag}(v_1, \cdots, v_k) = \begin{pmatrix} v_1 & \cdots & 0 \\ 0 & & v_k \end{pmatrix}$

Übergang $A \to \frac{1}{\lambda} V^{-1} A V = B$

Beispiel 8.2. $\underbrace{\begin{pmatrix} 6 & \frac{6}{5} & 2 \\ 20 & 4 & 0 \\ 6 & 3 & 3 \end{pmatrix}}_{A} \cdot \begin{pmatrix} 2 \\ 5 \\ 3 \end{pmatrix} = \begin{pmatrix} 24 \\ 60 \\ 36 \end{pmatrix} = 12 \begin{pmatrix} 2 \\ 5 \\ 3 \end{pmatrix}, \lambda_1 = 12$

$B = \frac{1}{12} \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{5} & 0 \\ 0 & 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} 6 & \frac{6}{5} & 2 \\ 20 & 4 & 0 \\ 6 & 3 & 3 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{2}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{5}{12} & \frac{1}{4} \end{pmatrix}$

$B \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

- $A = \lambda V B V^{-1}$

- $A^2 = \lambda^2 V B^2 V^{-1}$
- $A^n = \lambda^n V B^n V^{-1}$

(μ, u) ist (EW, EV) für A $(\frac{\mu}{\lambda_1}, V^{-1}m)$ ist (EW, EV) für B

$A^n x_0 = \lambda^n V B^n \underbrace{(V^{-1}x_0)}_{y_0}$

Behauptung

$\lim\limits_{n\to\infty} B^n y = \overline{y} = \binom{1}{1} \cdot \frac{\sum y_i}{n} \Rightarrow \lim \frac{A^n x_0}{\lambda^n} = V(\lim B^n y) = V\binom{1}{1}\text{Konst} = v_1 \cdot \text{Konst}$

Lemma 8.9. $C \in \mathbb{R}^{k\times k}$ sei eine Matrix $\geqslant 0$ mit Zeilensummen $= 1$ („stochastische Matrix“).

Weiterhin $\forall i, j : c_{ij} \geqslant \frac{c_0}{k} (0 \leqslant c_0 \leqslant 1$

v sei ein Vektor ein Zeilensumme 0. ($v \in \mathbb{C}^k$ ist erlaubt.)

$\Rightarrow ||Cv||_\infty \leqslant (1 - c_0) \cdot ||v||_\infty$

Beweis Kompontente i von $Cv = \sum\limits_{j=1}^{k} c_{ij} v_j \sum(c_{ij} - \frac{c_0}{k}) v_j + \underbrace{\sum \frac{c_0}{k} v_j}_{=0} \leqslant \sum(c_{ij} - \frac{c_0}{k})|v_j| \leqslant \sum(c_{ij} - \frac{c_0}{k})||v||_\infty = ||v||_\infty(1 - c_0)$ „fertig“

Beweis $y = \overline{y} + y^{\text{Rest}}$

$\sum y_i^{\text{Rest}} = 0$

$B^n y = B^n \overline{y} + B^n y^{\text{Rest}}$

$||B^n y^{\text{Rest}}||_\infty \leqslant \underbrace{(1 - b_0)^n}_{<1} ||y^{\text{Rest}}||_\infty \to 0.$

$\lim B^n y = \overline{y}$

Folgerung 8.10. B hat keinen anderen Eigenvektor $\geqslant 0$. Jeder andere EV von B hat $|EW| < 1$.

9 Gray-Codes ohne Schleife

9.1 Hochzählen ohne Schleife

Dabei sind die Bitfolgen $(b_{n-1}, b_{n-2}, \ldots, b_1, b_0) \in \{0, 1\}^n$ in lexikographischer Reihenfolge „ohne Schleife“

```
00101110011
 \/\--/ \-/
```

$(f_{10}, \cdots, f_0) = (0, 0, 0, 9, 0, 0, 0, 4, 0, 0, 0, 2)$

Definition 9.1. $f_i = \begin{cases} \text{Position der ersten 0 links von } b_i & \text{falls } b_i \text{ die rechteste Position in einem Einsenbloc} \\ 0 & \text{sonst} \end{cases}$

Algorithm 10 Hochzählen ohne Schleife

```
1: b_i = 0; f_i = 0
2: j = f_0
3: for i = 0 TO j − 1 do
4:     b_i = 0; b_j = 1
5:     Ausgabe/Verarbeitung (b_{n−1}, ··· , b_0)
6: end for
7: f_0 = 0
8: if f_{j+1} = 0 then
9:     f_j = j + 1
10: else
11:     f_j = f_{j+1}; f_{j+1} = 0
12: end if
```

Algorithm 11 Hochzählen ohne Schleife (optimiert)

```
1: b_i = 0; f_i = 0
2: j = f_0
3: for i = 0 TO j − 1 do
4:     b_i = 0; b_j = 1
5:     Ausgabe/Verarbeitung (b_{n−1}, ··· , b_0)
6: end for
7: f_0 = 0
8: f_j = f_{j+1}
9: f_{j+1} = j + 1
```

Beispiel 9.1.

a_3	a_2	a_1	a_0	f_4	f_3	f_2	f_1	f_0
0	0	0	0	4	3	2	1	$\underline{0}$
0	0	0	1	4	3	2	$\underline{1}$	0
0	0	1	1	4	3	2	2	$\underline{0}$
0	0	1	0	4	3	2	1	2
0	1	1	0	4	3	3	1	0

10 Gray-Code über Alphabet $(0, 1, \cdots, m-1)$

$000 \cdots 009, 019 \cdots 010, 020 \cdots 029, 039 \cdots 030$

$090, 190 \cdots 199 \cdots 198 \cdots 180$

Man speichert sich zu jeder Stelle eine Richtung $r_i = \pm 1$ und einen Endwert $e_i \in \{0, \cdots, n-1\}$

Eine Position ist <u>aktiv</u>, solange sie ihren Endwert noch nicht erreicht hat. a_i wird geändert, <u>bevor</u> irgendeine Stelle links von a_i geändert wird. Zeigen f_i dienen zum überspringen von passiven Stellen.

Algorithm 12 Erzeugung des Gray-Codes $(a_{n-1}, a_{n-2}, \cdots, a_0$

Initialisierung
$a_i = 0$
for $i = 0, 1, \cdots, n$ **do**
    $f_i = i$
end for
Nachfolger
$j = f_0$
if $j = n$ **then**
    Abbruch
end if
$a_j = 1 - a_j$
$f_0 = 0; f_j = f_{j+1}; f_{j+1} = j + 1$

Algorithm 13 Erzeugung des Gray-Codes $(a_{n-1}, a_{n-2}, \cdots, a_0$ (optimiert)

Initialisierung
$a_{n-1}, \cdots, a_0 = (0 \cdots 0), f_j = j(0 \leqslant j \leqslant n), r_i = +1(i = 0, \cdots, n-1)$
Nachfolger
$j = f_0$
if $j = n$ **then**
    Abbruch
end if
$a_j = a_j + r_j$
$f_0 = 0$
if $a_j = m - 1$ or $a_j = 0$ **then**
    $r_k = -r_j$
    $f_j = f_{j+1}$
    $f_{j+1} = j + 1$
end if

Eine symmetrische Kette in den Teilmengen von $0, 1, \cdots, n-1$ ist eine Folge $A_k \subseteq A_{k+1} \subseteq A_{k+1} \subseteq \cdots \subseteq A_{n-k} \subseteq \{0, 1, \cdots, n-1\}$ mit $|A_i| = i$

Gesucht ist eine Zerlegung von $2^{\{0,\cdots,n-1\}}$ in symmetrische Ketten.

Sei $\sigma_1\sigma_2\cdots\sigma_S$ eine Kette im n-Würfel.

Bilde daraus $\sigma_2 0\sigma_3 1\cdots\sigma_S 0$

$\sigma_1 0\sigma_1 1\sigma_2 1\cdots\sigma_S$ Zwei neue Ketten i $(n+1)$-Würfel.Für $s = 1$ fällt die erste Zeile weg.

```
n = 1
0 1

n = 2
   10
00 01 11

n = 3
    100 101
    010 110
000 001 011 111

n = 4
          1010
     1000 1001 1011
          1100
     0100 0101 1101
     0010 0110 1110
0000 0001 0011 0111 1111
```

Folgerung 10.1. Folgerung (Satz von Spermer) $B_1, B_2, \cdots, B_k$ sei eine Menge von teilmengen von $\{0, \cdots, n-1\}$ bei denen keine ein einer anderen enthalten ist (eine Antikette) $\Rightarrow k = \binom{n}{\frac{n}{2}}$

Beweis: $B_1, B_2, \cdots, B_k$ enthält höchstens ein Element aus jeder Zeile.

Bestimmung des Nachfolgerelementes Man übersetzt $0 \to (+1, -1)$ (Abstieg) und $1 \to (1, 1)$ (Aufstieg)

Kippe die tiefste Strecje $(1, -1)$, die von $(-\infty, 0)$ aus sichtbar ist zu $(1, 1)$, sofern sie unter der x-Achse liegt.

Bijektive Abbilgung von $\binom{n}{k}$ nach $\binom{n}{n-k}, k \leqslant \frac{n}{2}, s \leqslant f(s)$

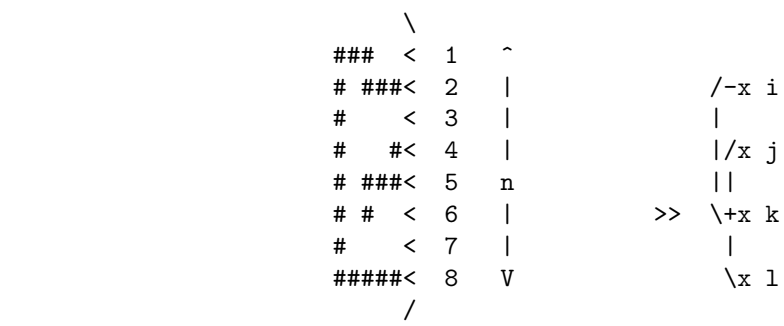
$S \subseteq \{0, \cdots, n-1\}$

1. Bestimme $i \notin S, |\{i+1, \cdots, k\} \cap S| \geqslant |\{i+1, \cdots, k\} \setminus S|$ für $i+1 \leqslant k \leqslant n-1$

   sonst $\{k, \cdots, i-1\} \cap S| \leqslant |\{k, \cdots, i-1\} \setminus S|$ für $0 \leqslant k \leqslant i-1$

2. Bestimme $f(s) = S \cup \Big\{ j \in S : \{k, \cdots, j-1\} \cap S| \leqslant \{k, \cdots, j-1\} \setminus S|$
   für $0 \leqslant k \leqslant j-1$
   sonst $|\{j, j+1, \cdots, i\} \setminus S| - |\{j, j+1, \cdots, i\} \cap S| \leqslant n - 2k \Big\}$

„Zustände“ beim Abzählen von Polyominos



$\{\{2, 8\}, \{4, 5\}\}$

Ein Zustand ist beschrieben durch

1. eine Teilmenge von $\{1, \cdots, n\}$: belegte Zellen
2. eine Partition dieser Teilmenge mit
   (a) zwei benachbarte Zellen gehören zur gleichen Klasse
   (b) kreuzungsfrei: $i, k \in K_1, j, l \in K_2 \neq K_1, i < j < k < l$ ist ausgeschlossen

Definition 10.2 (Motzkin-Weg). Ein Motzkin-Weg ist ein Weg $(a_1, a_2, \cdots, a_n)$ mit $a_i \in \{0, +1, -1\}$.

Es gilt $\sum_{i=1}^{k} a_i \geqslant 0 \forall k = 1, \cdots, n-1$ und $\sum_{i=1}^{n} a_i = 0$.

Dabei wird a_i als Schritt $(1, a_i)$ im Gitter interpretiert.

Weg von $(0, 0)$ zu $(n, 0)$ mit n Schritten der Art $(1, 0), (1, 1), (1, -1)$, der über der x-Achse bleibt.

Definition 10.3 (Motzkin-Zahl). Die Motzkin-Zahl beträgt die Anzahl der Motzkin-Wege der Länge n.

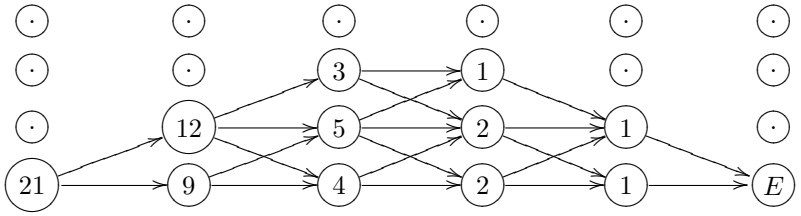
$M_n = n -$ te Motzkin-Zahl $\approx 3^n \frac{1}{n^{\frac{3}{2}}} \sqrt{\frac{27}{4\pi}}$

Rangbestimmung Anzahl der Wege, welche unterhalb des gewählten Weges liegen.

inverse Rangbestimmung Am Anfang beginnen, dann wie ein Weg in einem Netzwerk.

Beispiel 10.1. $M_i = 1, 1, 2, 4, 9, 21, 51, \cdots$ für $i \geqslant 0$

Beispiel 10.2. Graph für $n = 5$. Die Zahlen in den Knoten geben die Anzahl der Wege bis zum Endpunkt E an.



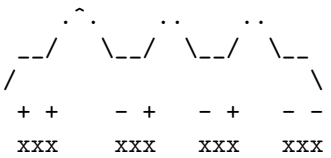
Satz 10.4. Es gibt eine Bijektion zwischen den Zuständen für n Zellen und den Motzkin-Wegen der Länge $n - 1$.

Beweis. $(a_1, a_2, \cdots, a_{n+1})$

a_i entspricht dem Übergang von Zelle $i - 1$ zur Zelle i.

Betrachte Blöcke von aufeinanderfolgenden belegten Zellen.

$$a_i = \begin{cases} +1 & \text{Anfang des ersten Blockes einer Komponente} \\ -1 & \text{Anfang eines anderen Blockes} \\ -1 & \text{Ende des letzen Blockes einer Komponente} \\ +1 & \text{Ende eines anderen Blockes} \\ 0 & \text{sonst (Übergang zwischen zwei belegten oder zwei freien Zellen)} \end{cases}$$



Bemerkung 10.5. Punkte auf gerader Höhe entsprechen leeren Zellen, Punkte auf ungerader Höhe entsprechen belegten Zellen. Alle Zelle einer Komponente sind auf gleicher Höhe.

Der Algorithmus 14 verwaltet einen Stapel $K_0, K_1, K_2, \cdots$ von Mengen.

$A(s, n) = A(\text{Nachf}_0(s), n) + A(\text{Nachf}_1(s), n - 1)$

s ist eine Zahl zwischen 0 und $M_{n+1} - 1$, Index in einem Feld.

Bestimmung des Nachfolgers s $\overrightarrow{\text{Rang}}^{-1}$ MW $\overrightarrow{\text{Dekod}}$ Zustand $\overrightarrow{\text{Nachf}_0, \text{Nachf}_1}$ Zustand $\longrightarrow MW \longrightarrow s$

Die Dekodierung kann übersprungen werden, indem direkt im Motzkin-Weg zwei Komponenten verschmolzen werden, indem die betrffenden $a_i = 1$ gesetzt und die dazwischenliegenden Knoten entsprechend angehoben werden.

Definition 10.6 (Catalan-Zahlen). Anzahl der Wege von $(0, 0)$ nach $(2n, 0)$
$a_i \in \{+1, -1, \emptyset\}$

$C_n = \frac{1}{n+1}\binom{2n}{n}$

Algorithm 14 Dekodierung eines Motzkin-Weges

```
1: Ebene := 0
2: for i := 1 TO n + 1 do
3:   if a_i = 1 then
4:     Ebene++
5:     if Ebene ungerade: then
6:       K_Ebene := 0 # beginne neue Komponente
7:     end if
8:   end if
9:   if a_i = −1 then
10:    if Ebene ungerade then
11:      Ausgabe K_Ebene # Komponente abgeschlossen
12:    end if
13:    Ebene–
14:  end if
15:  if Ebene ungerade then
16:    K_Ebene = K_Ebene ∪ {i}
17:  end if
18: end for
```

11 Polyominos auf dem geschraubten Zylinder

nach: Barequet, Moffie, Rib'o, Rote

Definition 11.1 (geschraubter Zylinder). Gegeben sei ein Gitter in der Ebene. Beginn der Schraubung sei die Zelle $(0, 0)$ Wir bezeichnen die Schraubenbreite mit w , Beim Aufrollen wird die Zeile (i, j) mit der Zelle $(i + 1, j + w)$ identifiziert.

Lemma 11.2. In der Ebene gibt es mindestens so viele n-ominos wie auf dem geschraubten Zylinder.

Jedes n-omino in $\mathbb{Z}^2$ kann man auf den Zylinder aufwickeln. Dabei entsteht ein Polyomino mit $\leqslant n$ Zellen (Felder auf dem Zylinder können mehrfach belegt werden.)

Ein Polyomino mit n Zellen auf dem Zylinder läßt sich sich in die Ebene zu einem n-omino auseinanderfalten, aber im allgemeinen nicht eindeutig.

Wähle einen aufspannenden Baum im Gittergraphen auf dem Zylinder. Dieser Baum kann dann eindeutig in die Ebene entfaltet werden und liefert ein n-omino. Diese Abbildung ist injektiv.

Bemerkung 11.3 (Vorteil des geschraubten Zylinders). Der Aufbau der Polyominos erfolgt durch Hinzufügen einer einzelnen Zelle. Dies ist die einzige Art von Operation, die nötig ist.

Zähle Polyominos mit n Zellen abhängig vom Zustand der W Randzellen. Ein Zustand ist eine nichtkreizende Partition der Randzellen. Dies entspricht einem Motzkin-Weg der Länge $w + 1$.

Nach dem Hinzufügen einer neuen freien oder belegten Zelle „0“ und dem Berechnen des Nachfolgerzustandes muss man die Zellen $0, 1, \cdots, w - 1$ auf

$1, \cdots, w$ umbenennen.

Vorwärtsiteration Man berechnet also $X_s^{(n)} = \#$ Plyominus mit n Zellen im Zustand s.

$$x^{(n)} = \begin{pmatrix} x_1^{(n)} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{pmatrix} M_{W-1} - 1 \text{ Komponenten}$$

$x^{(1)} \to x^{(2)} \to x^{(3)} \to \cdots$

$$y^{(n)} = \begin{pmatrix} y_1^{(n)} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{pmatrix}$$

Rückwärtsitereation $Y_s^{(k)} = Y^{(k)}_{\text{Nachf}_0(s)} + Y^{(k+1)}_{\text{Nachf}_1(s)}$

$y^{(0)} \to y^{(-1)} \to y^{(-2)} \to \cdots$

Dabei enthält der Vektor y^{-n} Informationen, wieviele n-ominos es gibt.

$y^{(k)} = T \cdot y^{(k+1)}$ für eine bestimmte Matrix $T \geqslant 0$

$y^{(-k)} \approx \lambda^n v_0$

$\underline{\lambda} y^{(-k)} \leqslant y^{(-k-1)} \leqslant \overline{\lambda} y^{(-k)} \to \underline{\lambda} \leqslant \lambda \leqslant \overline{\lambda}$

$\underline{\lambda} = \min_s \frac{y_s^{(-k-1)}}{y_s^{(k)}}$

$\overline{\lambda} = \max_s \frac{y_s^{(-k-1)}}{y_s^{(k)}}$

$\underline{\lambda}, \overline{\lambda}$ sind die Schranken für die „Wachstumskonstante“ der Anzahl der Polyominos

Für $w = 22$ über 140 Iterationen ergaben $underline\lambda = 3.980157, \overline{\lambda} = 3.180142, M_2 3 \approx 10^9$

Bemerkung 11.4 (Klarners Konstante). Die Zahl 3.9801 ist untere Schrank für die Wachstumskonstante von Polyominos in der Ebene

Beispiel 11.1. $w = 6$, $\{\{1,3\},\{5,6\}\}$

```
 X3
 .2
xX1     Zustand ohne Vorgänger
X6
X5
.4
```

```
111      2          2
  1  +  22  =  11122
                  1
```

$A \oplus B$: Die Anfangszelle von B unmittelbar rechts von der Endzelle von A.

$\rightarrow$ Eindeutige Zerlegun in unzerlegbare Polyominos.

$A_n = G_1 \oplus A_{n-1} \cup G_2 \oplus A_{n-2} \cup \cdots \cup (G_{n-1} \oplus A_1 \cup G_n$

G_i ist die Menge der unzerlegbaren Polyominos mit i Zellen, A_n die Menge aller n-ominos.

$a_n = g_1 a_{n-1} + \cdots + g_{n-1} a_1 + g_n$

$a_1 \cdots a_{56}$ sind bekannt.

$a'_n = g_1 a'_{n-1} + g_2 a'_{n-2} + \cdots + g_{56} a'_{n-56}, a'_n \leqslant a_n$ für $n \geqslant 57$

$\underbrace{\text{Wachstumskonstante von } a'_n}_{\sqrt{\lambda^{56}} = \sqrt{g_1 \lambda^{56} + \cdots + g_{55}\lambda + g_{56}}} \leqslant$ Klarnersche Konstante. $\lambda = 3.8746, g_{n+1} \geqslant g_n$

12 Aufzählen von schwer strukturierbaren Daten

- Triangulierungen eines konvexen Polygons
- alle Topologischen Sortierungen in einem azyklisch gerichteten Graphen.
- alle Zellen, die beim Schnitt von $n \geqslant 3$ Geraden in der Ebene entstehen

Wir betrachten das Problem abstrakt. Gegeben sei eine Menge von Objekten, die man aufzählen will. Diese Objekte werden als Knoten eines Graphen G betrachtet.

Zudem definieren wir eine Nachbarschaftsrelation zwischen Objekten, wodurch wir die Kanten des Graphen G erhalten.

Die Relation soll symmetrisch sein ($\rightarrow$ G ungerichtet). G soll zusammenhängend sein. (ist jeweils zu beweisen!)

Der Graph ist nur durch die Nachbarschaftsrelation gegeben. Für einen gegebenen Knoten v kann man die Menge $\Gamma(v)$ der Nachbarn berechnen.

Ziel: Alle Knoten aufzählen bzw. besuchen. Bei der Tiefen- bzw. Breitensuche (Beginn mit einem beliebigen Startknoten v_0) wird viel Speicher benötigt, um die besuchten Knoten zu markieren.

Definition 12.1 (Reverse Search (umgekehrte Suche)(Avis, K. Fukuda)). Definiere eine Nachfolgerfunktion $f(v) \in \Gamma(v), v \in V \backslash \{v_0\}$ mit Ausnahme eines einzigen Knotens $v_0, f(v_0) = f(v_0)$, so dass für jeden Knoten v die Folge $v, f(v), f(f(v)), \cdots$ zu v_0 führt. (Äquivalent: die Folge kehrt nicht zu v zurück

Diese Kanten $\{(v, f(v))|v \in V \setminus \{v_0\}\}$ bilden einen gerichteten aufspannenden Baum mit Wurzel v_0.

Unter diesen Annahmen kann man eine Tiefensuche auf dem Baum ohne zusätzlichen Speicherbedarf durchführen.

Algorithm 15 Nachfolger in lexikographischen Reihenfolge

v ist der aktuelle Knoten
bereits die ersten i Nachfahren von v in der Liste $\Gamma(v)$ abgearbeitet.
Sei w der $(i+1)$te Knoten in $\Gamma(v)$
if $f(w) = v$ **then**
  w ist Kind von v, gehe zu w
else
  $i++$ (untersuche den nächsten Nachbarn)
end if
if $f(v) = v$ **then**
  STOP
end if
if alle Nachbarn in $\Gamma(v)$ untersucht **then**
  gehe zurück zu $f(v)$
end if
v sei der i-te Knoten in $\Gamma(f(v))$. Setze $v = f(v)$

Bemerkung 12.2. Die Berechnung von $\Gamma(v)$ muss die Nachbarn immer in der selben Reihenfolge erzeugen

Speicherbedarf Speicher von $v, w, \Gamma(v)$

Zeitaufwand

- Berechnen von $f : s(t)$
- Berechnen von $\Gamma : 1 + 2(M - 1)$

12.1 Abschätzen der Anzahl der Blätter eines Baumes

Der Baum sei nur durch ein Orakel gegeben, dass zu jedem Knoten v die Liste der Kinder liefert.

Wähle in jedem Schritt ein Kind zufällig gleichverteilt aus der Liste der Kinder, bis man ein Blatt erhält.

B = Produkt der Knotengrade auf diesem Weg.

Satz 12.3. Der Erwartungswert von B ist die Anzahl der Blätter

$$\begin{aligned} E(B) &= \sum_{\text{Blätter } v} \mathbb{P}(\text{der Weg zu } v \text{ wird ausgewählt} \cdot B_v \\ &= \sum \prod_{u \text{ auf dem Weg zu } v} \frac{1}{Grad(u)} \\ &= \prod_{u \cdots} Grad(u) \\ &= \sum 1 = \#\text{ Blätter} \end{aligned}$$

13 Unique Sink Orientations (USOs) des Würfels

Alle Bitketten der Länge k bilden die Ecken des k-dimensionalen Hyperwürfels. Kanten zwischen Ecken, die sich an genau einer Stelle unterscheiden. Eine Seite des Hyperwürfels besteht aus allaen Ecken, die an einer Teilmenge der Stellen einen festen Wert haben. $010 * *1 * 1$ ist eine Seite des 8-Würfels, ein 3-dimensionaler Würfel. Eindimensionale Seiten sind Kanten.

Definition 13.1 (Senke). Eine Senke ist eine Ecke, aus der keine gerichtete Kante hinausführt.

Eine Orientierung des Würfels gibt jeder Kante des Würfels eine der zwei möglichen Richtungen.

Definition 13.2 (USO). Eine Orientierung mit eindeutigen Senken ist eine Orientierung, bei der jede Seite des Würfels eine eindeutige Senke besitzt

Eine USI sei durch folgendes Orakel gegeben:

Eingabe: eine Ecke v Ausgabe: Orientierung aller Kanten, die mit v gesucht: Senke (Der Algorithmus muss die Senke abfragen, auch wenn er sie schon weiß) Wieviele Anfragen sind nötig, um die Senke zu bestimmen?

$\approx \sqrt{3}^k$

Lemma 13.3. Wenn man einen Algorithmus für den k_1-Würfel (k_2) mit a_1 (a_2) Anfragen hat, dann kann man den $k_1 + k_2$-Würfel mit $a_1 \cdot a_2$ Anfragen lösen.

14 Entscheidungsbäume für Suchprobleme

Definition 14.1 (Suchproblem). Man muss aus einer endlichen Anzahl von möglichen Lösungungen eine herausfinden.

Dabei darf man nur bestimmte Fragen stellen.

Definition 14.2 (Entscheidungsbaum). innere Knoten: Fragen des Algorithmus'

Kanten zu den Kindern: mögliche Antworten

Blätter: Ausgabe der Lösung

Jedem Knoten im Baum ist die Menge der Möglichkeiten zugeordnet, die von dem Wurzelknoten zu diesem Knoten führen.

Die Möglichkeiten (USOs) bei denen der abgefragte Knoten die Senke ist, werden nicht an die Kinder weitergereicht. Bei diesen Möglichkeiten terminiert das USO-Spiel.

Eine Fragestraegie läßt sich als Baum darstellen. Jeder Knoten hat $\leqslant 7$ Kinder (hängt von den Fragen auf dem Weg dothin ab).

Tiefe $\leqslant 8$ (doppelte Fragen sind unsinnig)

Endlich viele mögliche Strategien. Länge des Algorithmus (im schlimmsten Fall) = Höhe des Baumes +1.

14.1 Reduzierung des Baumes durch Symmetrieüberlegung

Satz 14.3. o.B.d.A kann an als ersten den Knoten 000 abfragen.

14.1.1 Symmetrien des Würfels

1. *Spiegelung an den Koordinatenebenen.* An einer festen Position werden 0 und 1 vertauscht. Dadurch kann man jede beliebige Ecke an die Position 000 bringen. (2^n)
2. *Vertauschungen der Koordinaten* z.B: $b_1b_2b_3 \to b_3b_2b_1$. Die Ecken 000 und 111 bleiben dabei fest. ($n!$)

Insgesamt hat der Würfel $2^n n!$ Symmetrien.

14.2 randomisierte Strategien

Man kann zu einen gegebenen Zeitpunkt zufällig (gemäß irgendeiner Verteilung) entscheiden, welchen Knoten man als nächsten abfragt. → gemischte Strategie.

Bemerkung 14.4. Jede gemischte Strategie ist eine Mischung von reinen Strategien. Zufallsentscheidungen lassen sich an den Anfang verlegen, danach kommen dann nur noch reine Strategien.

14.3 Auszählungsmatrix

A: reine Strategien, x: USOs („Zustände“)

$C \in \mathbb{M}_{(A\times x)}, c_{ij} = \#$ Fragen bei Strategie A_i und USO x_j

gemischte Strategie: $\lambda_1 A_1 + \lambda_2 A_2 \cdots$ mit Wahrscheinlichkeiten $\lambda_i geq 0, \sum \lambda_i = 1$.

Erwartete Anzahl von Fragen (im schlimmsten Fall über alle x_j) → MIN.

Wenn x_j vorliegt: $\max_j(\sum_j \lambda_i c_{ij}) \to$ MIN.

minimale z unter:

$\forall j : \sum_i \lambda_i c_{ij} \leqslant z, \sum \lambda_j = 1, \lambda_j \geqslant 0$

ist ein lineares Optimierungsproblem (lineares Programm).

man kann die Senken im $k-$dimensionalen Würfel mit einem USO mit $O(2.975^{\frac{k}{3}}$ Anfragen im Mittel bestimen mit Hilfe eines randomisierten Algorithmus'.