Übungen zu Diskrete Mathematik und Theoretische Informatik

Markus Fulmek

Sommersemester 2022

Zur Verfügung gestellt von:
Markus Fulmek
UE Diskrete Mathematik und Theoretische Informatik, SoSe 2022
Fakultät für Mathematik, Universität Wien
Danke!

Zu den Programmieraufgaben in diesem Übungsskriptum steht auf Moodle ein Jupyter–Notebook `uebungen.ipynb` mit Hinweisen und Hilfsfunktionen zur Verfügung: Bitte benutzen Sie dieses Notebook und verwenden Sie insbesondere für die auszuarbeitenden Funktionen und Programmteile das dort vorgegebene "standardisierte Format".

Graph(ik)en, Permutationen und erzeugende Funktionen

Graph(ik)en modellieren und visualisieren Sachverhalte

Aufgabe 1 (⋆)**:** *Zeigen Sie: Jede Zusammenhangskomponente eines einfachen (endlichen) Graphen mit* Maximalgrad 2 *(d.h., für alle* $v \in V(G)$ *gilt* $\deg(v) \leq 2$*) ist entweder ein Weg oder ein Kreis.*

Aufgabe 2 (⋆ ⋆)**:** *Die Kantenmenge* $E(G)$ *eines* einfachen *(endlichen) Graphen* G *auf der Knotenmenge* $V = V(G)$ *ist (definitionsgemäß) eine Teilmenge der Familie der zweielementigen Teilmengen von* G*:*

$$E(G) \subseteq \binom{V}{2}.$$

Das Komplement *von* G *sei der Graph* $\overline{G}$ *auf derselben Knotenmenge wie* G *(also* $V\left(\overline{G}\right) = V(G)$*), aber mit der komplementären Kantenmenge*

$$E\left(\overline{G}\right) = \binom{V}{2} \setminus E(G).$$

Zeigen Sie: Für alle endlichen einfachen Graphen G *ist* G *oder* $\overline{G}$ zusammenhängend*.*

Aufgabe 3 (⋆)**:** *Zeichnen Sie das Hasse–Diagramm der Potenzmenge von* $[4]$*, geordnet durch* Mengeninklusion*:*

$$A \preceq B \overset{\text{def}}{\iff} A \subseteq B.$$

Aufgabe 4 (⋆)**:** *Sei* $f : X \to Y$ *eine* bijektive *Funktion: Formulieren Sie das in der "Sprache der bipartiten gerichteten Graphen".*

Zeichnen Sie alle verschiedenen bijektiven Funktionen $[3] \to [3]$

- *als* bipartite *gerichtete Graphen (mit Knotenmenge ist disjunkte Vereinigung von zwei "Kopien"* $[3]$ *und* $[3]$*)*
- *und als gerichtete Graphen mit Knotenmenge* $[3]$*.*

(Hinweis: Es gibt genau 6 *solche Funktionen!)*

Permutationen

Aufgabe 5 (⋆)**:** *Schreiben Sie jeweils eine kleine Python–Funktion, die die folgenden Zahlen rekursiv berechnet, und geben Sie die entsprechenden Zahlenfolgen für* $n = 0, 1, 2, \ldots, 9$ *aus.*

(1) *Die Doppelfaktorielle* $n!!$ *wird definiert durch* $0!! = 1!! = 1$ *und*

$$n!! = n(n-2)(n-4)\cdots 1 \text{ für } n \text{ ungerade und}$$
$$n!! = n(n-2)(n-4)\cdots 2 \text{ für } n \text{ gerade.}$$

(2) *Die Folge der Fibonacci–Zahlen* F_n *ist definiert durch* $F_0 = 0$, $F_1 = 1$ *und* $F_n = F_{n-1} + F_{n-2}$.

(3) *Die Fibonacci–Faktorielle* $n!_F$ *(auf englisch auch kurz: "Fibonorial") ist gleich dem Produkt*

$$n!_F = \prod_{i=1}^{n} F_i$$

und $0!_F = 1$.

Aufgabe 6 (⋆ ⋆)**:** *Eine Permutation* $\pi \in \mathfrak{S}_n$ *entspricht eindeutig der* Permutationsmatrix $M(\pi)$*, das ist eine* $n \times n$*–Matrix, deren Eintragungen wie folgt gegeben sind:*

$$(M(\pi))_{i,j} = \delta_{j,\pi(i)}.$$

(Hier bezeichnet $\delta_{i,j}$ *das* Kronecker–Delta*:* $\delta_{i,j} = 1$*, wenn* $i = j$*, für* $i \neq j$ *ist* $\delta_{i,j} = 0$*. Anders gesagt: Die Eintragungen von* $M(\pi)$ *sind alle* 0*, bis auf die Eintragungen* $(M(\pi))_{i,\pi(i)}$ *— die sind alle* 1*.)*

Zeigen Sie: Die Hintereinanderausführung von Permutationen $\pi \circ \tau$ *entspricht genau der Multiplikation der entsprechenden Matrizen "in umgekehrter Reihenfolge", also*

$$M(\tau) \cdot M(\pi) = M(\pi \circ \tau).$$

Aufgabe 7 (⋆ ⋆)**:** *Wenn man sich vor Augen führt, daß eine Permutationsmatrix* $M(\pi)$ *einer "Darstellung des* Funktionsgraphen*" von* $\pi\colon [n] \to [n]$ *entspricht (nur daß die* x*–Achse nach unten und die* y*–Achse nach rechts zeigt statt — wie in der Schule gewohnt — nach rechts und nach oben), wird sofort klar:*

$$M\left(\pi^{-1}\right) = (M(\pi))^T;$$

denn die transponierte *Matrix* $(M(\pi))^T$ *entspricht der "an der Hauptdiagonale* gespiegelten*" Matrix* $M(\pi)$*.*

Eine quadratische Matrix kann man aber nicht nur an der Hauptdiagonale spiegeln, sondern auch an ihrer horizontalen *Symmetrieachse. Sei* $\pi \in \mathfrak{S}_n$ *mit Permutationsmatrix* $M(\pi)$*:*

Drücken Sie die Permutation, die der an der horizontalen Symmetrieachse gespiegelten Matrix $M(\pi)$ *entspricht, durch* π *und* n *aus.*

Drücken Sie die Permutation, die der um $90° = \pi/2$ gedrehten *Matrix* $M(\pi)$ *entspricht, durch* π^{-1} *und* n *aus. (Hinweis: Betrachten Sie die Sache "rein geometrisch" — zwei Spiegelungen ergeben eine Drehung . . .)*

Aufgabe 8 (⋆)**:** *Eine quadratische* $n \times n$*–Matrix* M *mit reellen Eintragungen nennt man* doppelt stochastisch*, wenn*

- *für alle* $1 \leq i, j \leq n$ $M_{i,j} \geq 0$ *gilt,*
- *für alle* $1 \leq i \leq n$ $\sum_{k=1}^{n} M_{i,k} = \sum_{k=1}^{n} M_{k,i} = 1$ *gilt*

(also: alle Zeilensummen und alle Spaltensummen von M *sind gleich* 1*).*

Klarerweise ist jede Permutationsmatrix $M(\pi)$ *(für* $\pi \in \mathfrak{S}_n$*) eine doppelt stochastische Matrix.*

Eine Konvexkombination *von Elementen* $v_1, v_2, \ldots, v_k$ *eines reellen Vektorraums ist eine* Linearkombination

$$\lambda_1 \cdot v_1 + \lambda_2 \cdot v_2 + \cdots + \lambda_k \cdot v_k$$

mit folgenden Bedingungen an die Skalare λ_i*:*

- *für* $1 \leq i \leq k$ *gilt* $\lambda_i \geq 0$*,*
- $\lambda_1 + \lambda_2 + \cdots + \lambda_k = 1$*.*

Die reellen $n \times n$*–Matrizen bilden einen Vektorraum (der Dimension* n^2*) über* $\mathbb{R}$*. Zeigen Sie: Jede Konvexkombination von Permutationsmatrizen* $M(\pi)$ *(*$\pi \in \mathfrak{S}_n$ *für ein festes* n*) ergibt eine doppelt stochastische Matrix.*

Aufgabe 9 (⋆ ⋆)**:** *Eine Permutation* $\pi \in \mathfrak{S}_n$ *können wir als Punkt im* $\mathbb{R}^n$ *auffassen, wenn wir sie in Einzeilen–Notation schreiben. Die Menge aller* Konvexkombinationen *dieser "Permutations–Punkte"* π_k*,* $1 \leq k \leq n!$*, also*

$$\left\{ \sum_{k=1}^{n!} \lambda_k \pi_k \colon 0 \leq \lambda_k \leq 1 \text{ und } \sum_{k=1}^{n!} \lambda_k = 1 \right\},$$

nennt man Permutaeder *der Ordnung* n*.*

Zeigen Sie, daß der Permutaeder *der Ordnung* n *in einer* Hyperebene *(also in einem affinen Teilraum der Dimension* $n-1$*) des* $\mathbb{R}^n$ *enthalten ist.*

Hinweis: Eine Hyperebene *ist durch eine lineare Gleichung gegeben: Finden Sie eine lineare Gleichung, die alle Permutationen (gedeutet als Vektoren) erfüllen.*

Aufgabe 10 (⋆)**:** *Schreiben Sie ein kleines Python–Programm, das die Multiplikationstabelle der Gruppe* $\mathfrak{S}_4$ *erzeugt und in einer ansprechenden Form ausgibt. (Wenn Sie sich mit LaTeX gut auskennen, können Sie auch versuchen, die Tabelle in diesem Format auszugeben).*

Hinweis: Betrachten Sie die Code–Beispiele im Skriptum und verwenden Sie die dort skizzierten Bausteine.

Aufgabe 11 (⋆)**:** *Schreiben Sie eine Python–Funktion, die für eine gegebene Permutation π (als Liste oder Tupel dargestellt) alle Inversionen von π ausgibt und deren Anzahl als Rückgabewert liefert.*

Aufgabe 12 (⋆ ⋆)**:** *Sei $n \in \mathbb{N}$. Die* schwache Bruhat–Ordnung *auf $\mathfrak{S}_n$ ist definiert durch die Bedeckungsrelation*

$$\pi \lessdot \varphi \iff \exists \tau = (i, i+1) : \varphi = \pi \circ \tau \text{ und } inv\,\varphi = 1 + inv\,\pi.$$

(D.h., τ ist hier eine kanonische *Transposition.)*

Die starke Bruhat–Ordnung *auf $\mathfrak{S}_n$ ist definiert durch die Bedeckungsrelation*

$$\pi \lessdot \varphi \iff \exists \tau = (i, j) : \varphi = \pi \circ \tau \text{ und } inv\,\varphi = 1 + inv\,\pi.$$

(D.h., τ ist hier eine beliebige *Transposition.)*

Zeichnen Sie ein Hasse–Diagramm für die starke und die schwache Bruhat–Ordnung auf $\mathfrak{S}_3$.

Aufgabe 13 (⋆)**:** *Eine Permutation $\pi \in \mathfrak{S}_n$, für die*

$$\pi = \pi^{-1} \text{ (oder äquivalent: } \pi^2 = \mathbf{id})$$

gilt, nennt man Involution*. Charakterisieren Sie die* Zyklenzerlegung *einer Involution π.*

Aufgabe 14 (⋆ ⋆)**:** *Schreiben Sie drei Python–Funktionen, die das Signum einer Permutation auf je unterschiedliche Art berechnen:*

(1) *Benutzen Sie die Definition des Signums.*
(2) *Benutzen Sie die Proposition zum Zusammenhang zwischen $sgn\,(\pi)$ und $inv\,\pi$ und Übungsaufgabe 11.*
(3) *Benutzen Sie das Korollar zum Signum für eine Zyklenzerlegung und die Funktion* `cycle_decomposition`*.*

Untersuchen Sie die Laufzeitunterschiede der drei Funktionen empirisch mit `%timeit`*.*

Abzählung und erzeugende Funktionen

Aufgabe 15 (⋆)**:** *Berechne die Wahrscheinlichkeit, mit einem Tip beim Lotto "6 aus 45"*

a: *einen Fünfer (also genau 5 richtige Zahlen) zu tippen,*
b: *einen Vierer (also genau 4 richtige Zahlen) zu tippen.*

Aufgabe 16 (⋆ ⋆)**:** *Im Parlament eines Landes gibt es 151 Sitze und drei Parteien. Wieviele Möglichkeiten der Sitzverteilung gibt es, sodaß keine Partei eine absolute Mehrheit (d.h., mehr als 75 Sitze) hat?*

Aufgabe 17 (⋆)**:** *Beweise die Aussage: Sei* $\mathbf{G}\,(V,E)$ *ein Graph. Dann gilt:*

$$\sum_{v\in V(\mathbf{G})} \deg(v) = 2\cdot\#(E)\,.$$

(Hinweis: Erinnern Sie sich an die Regel von der doppelten Abzählung!)

Aufgabe 18 (⋆ ⋆)**:** *Die* Laplace–Matrix $L(G)$ *eines Graphen* $G = G(V,E)$ *ist eine quadratische Matrix, deren Zeilen und Spalten durch die Knotenmenge* V *indiziert werden, sie wird mithilfe der* Inzidenzmatrix $I(G)$ *von* G *definiert (deren Zeilen den Knoten und deren Spalten den Kanten von* G *entsprechen):*

$$L(G) \stackrel{def}{=} I(G)\cdot I(G)^T\,.$$

Finden Sie eine Charakterisierung der Einträge $L(G)_{i,j}$ *in Abhängigkeit von den Knoten* v_i *und* v_j *und zeigen Sie, daß für die Spur von* $L(G)$ *gilt:*

$$trace\,(L(G)) = 2\cdot\#(E)\,.$$

Aufgabe 19 (⋆)**:** *Ein einfacher Graph* G *heißt* kubisch*, wenn alle seine Knoten Grad* 3 *haben. Zeigen Sie:*

- *Ein kubischer Graph hat immer eine gerade Anzahl von Knoten.*
- *Für jede natürliche Zahl* $n \geq 2$ *gibt es kubische Graphen mit* $2n$ *Knoten.*

Hinweis: Zeichnen Sie den ("im Wesentlichen einzigen") kubischen Graphen auf 4 *Knoten und überlegen Sie, wie Sie "aus einem Knoten drei machen können", sodaß wieder ein kubischer Graph (nun auf* 6 *Knoten) entsteht.*

Aufgabe 20 (⋆ ⋆)**:** *Die rationale Zahl*

$$H_n \stackrel{def}{=} \sum_{i=1}^{n} \frac{1}{i}$$

heißt harmonische Zahl*. Aus der Analysis ist bekannt, daß* H_n *ungefähr gleich* $\log(n)$ *ist:* $H_n \sim \log(n)$*.*

Sei $j \in \mathbb{N}$ *und bezeichne* $t(j)$ *die Anzahl der positiven Teiler von* j*. Bezeichne weiters* $\overline{t}(n)$ *die* durchschnittliche *Anzahl der positiven Teiler der Zahlen von* 1 *bis* n*, also*

$$\overline{t}(n) \stackrel{def}{=} \frac{1}{n}\sum_{i=1}^{n} t(i)\,.$$

Zeige:

$$\overline{t}(n) = \frac{1}{n}\sum_{i=1}^{n} \left\lfloor \frac{n}{i} \right\rfloor.$$

(Hinweis: Dies ist eine Anwendung der Regel von der doppelten Abzählung!)

Schätze die Differenz $\left(H_n - \overline{t}(n)\right)$ *(ganz grob) ab und folgere:*

$$H_n - 1 \leq \overline{t}(n) \leq H_n.$$

Aufgabe 21 (⋆ ⋆)**:** *Zeige, daß für die Binomialkoeffizienten* $\binom{n}{k}$ *gilt:*

$$\binom{n}{0} < \binom{n}{1} < \cdots < \binom{n}{\lfloor n/2 \rfloor} = \binom{n}{\lceil n/2 \rceil} > \cdots > \binom{n}{n}.$$

(Diese Eigenschaft heißt Unimodalität *der Binomialkoeffizienten.)*

Aufgabe 22 (⋆ ⋆)**:** *Finden Sie einen geschlossenen Ausdruck für die Summe*

$$S(n) = \sum_{k=1}^{n} (-1)^{k-1} \cdot k \cdot \binom{n}{k} \cdot 2^{n-k}$$

und beweisen Sie dessen Richtigkeit für alle $n \in \mathbb{N}$*.*

Hinweis: Eine Möglichkeit liefert der Binomische Lehrsatz mit einer kleinen "ergänzenden Komplikation".

Aufgabe 23 (⋆ ⋆)**:** *Es ist oft sinnvoll, eine Identität, die man beweisen möchte, vorher in einem kleineren Zahlenraum zu überprüfen: Man setzt also konkrete Zahlen ein und schaut, ob die Identität für diese Zahlen korrekt ist.*

Benutzen Sie Python, um zu überprüfen, ob die Binomialidentitäten aus der folgenden Aufgabe 24 für ganze Zahlen n *und* k *im Zahlenraum* $1 \leq k \leq n \leq 15$ *korrekt sind.*

Hinweis: : Der Binomialkoeffizient ist ab Version 3.8 bereits in Python implementiert: `import math` *liest das Module* `math` *ein, danach liefert* `math.comb(4,2)` *den Binomialkoeffizienten* $\binom{4}{2}$*.*

Aufgabe 24 (⋆ ⋆)**:** *Man gebe kombinatorische Beweise für die folgenden Binomialidentitäten:*

(a) $n\binom{n-1}{k-1} = k\binom{n}{k}$*. (Anleitung: Wieviele Möglichkeiten gibt es, aus einer* n*-elementigen Menge eine* k*-elementige Teilmenge auszuwählen, und in dieser ein Element rot zu färben?)*

(b) $\sum_{i=0}^{n} i\binom{n}{i} = n \cdot 2^{n-1}$.

Aufgabe 25 (⋆ ⋆)**:** *Man gebe kombinatorische Beweise für die folgenden Binomialidentitäten:*

(a) $\binom{n+1}{k+1} = \sum_{m=k}^{n} \binom{m}{k}$.

(b) $\binom{n}{k}\binom{k}{m} = \binom{n}{m}\binom{n-m}{k-m}$.

Aufgabe 26 (⋆ ⋆)**:** *Zeige durch eine* Bijektion*: Für* $n \geq 1$ *ist die Anzahl der Teilmengen von* $[n]$ *mit* gerader *Mächtigkeit genauso groß wie die Anzahl der Teilmengen von* $[n]$ *mit* ungerader *Mächtigkeit*

Aufgabe 27 (⋆ ⋆)**:** *Wieviele Lottotips gibt es bei "6 aus 45", in denen keine zwei* aufeinanderfolgenden *Zahlen vorkommen?*

Hinweis: Finden Sie eine Bijektion der gesuchten Objekte auf die 6*–elementigen Teilmengen aus* $[40]$*.*

Elementares Abzählen

Aufgabe 28 (⋆ ⋆)**:** *Sei* S *eine Menge mit* $\#(S) = n$*, sei* $k \in \mathbb{N}$ *fest gewählt. Wieviele* k*–Tupel*

$$(T_1, T_2, \ldots, T_k)$$

von Teilmengen von S *gibt es, sodaß*

$$T_1 \subseteq T_2 \subseteq \cdots \subseteq T_k \text{ ?}$$

Aufgabe 29 (⋆)**:** *Wieviele Möglichkeiten gibt es,* k *einander nicht schlagende Türme auf einem* $n \times n$ *Schachbrett zu placieren?*

Aufgabe 30 (⋆ ⋆)**:** *Wieviele verschiedene Möglichkeiten gibt es,* n *Personen um einen runden Tisch zu setzen? (Zwei Anordnungen* π *und* τ *betrachten wir in diesem Zusammenhang als "gleich", wenn alle Personen in* π *denselben linken und denselben rechten Nachbarn haben wie in* τ*.)*

Aufgabe 31 (⋆ ⋆)**:** *Auf wieviele verschiedene Arten kann man* $2n$ *Personen zu* n *(ungeordneten) Paaren zusammenfassen? Gib alle Möglichkeiten für* $n = 1, 2, 3$ *explizit an.*

Aufgabe 32 (⋆ ⋆)**:** *An einem Bridgeturnier nehmen* $4n$ *Spieler teil, und das Turnier findet an* n *Tischen statt. Jeder Spieler benötigt einen anderen Spieler als Partner, und jedes Paar von Partnern benötigt ein anderes Paar als Gegner. Auf wieviele Arten kann die Wahl von Partner und Gegner erfolgen?*

Aufgabe 33 (⋆ ⋆)**:** *Auf wieviele Arten können wir die Zahlen* $1, 2, \ldots, n$ *anordnen, sodaß — abgesehen vom ersten Element — die Zahl* k *nur dann placiert werden kann, falls* $k-1$ *oder* $k+1$ *bereits placiert wurden (also links von* k *stehen)? (Zum Beispiel für* $n = 6$*:* $3\ 2\ 4\ 5\ 1\ 6$ *oder* $4\ 3\ 5\ 2\ 1\ 6$*.)*

Aufgabe 34 (⋆)**:** *Zeige folgende Verschärfung des Schubfachprinzips:*

Sei $f : [k] \to [n]$ *mit* $k > n$*, dann gibt es ein Element* $m \in [n]$*, für das gilt:*

$$\#\left(f^{-1}(m)\right) \geq \left\lfloor \frac{k-1}{n} \right\rfloor + 1.$$

Aufgabe 35 (⋆ ⋆)**:** *Beweisen Sie die* Binomial–Identität *aus Übungsaufgabe 25*

$$\binom{n+1}{k+1} = \sum_{m=k}^{n} \binom{m}{k}$$

durch Koeffizientenvergleich *aus einer geeigneten* Polynomidentität. *Zeigen Sie dazu*

$$\sum_{i=1}^{n+1} (1+x)^{i-1} = \sum_{i=1}^{n+1} \binom{n+1}{i} x^{i-1}.$$

Hinweis: Verwenden Sie die geometrische Reihe und den binomischen Lehrsatz.

Aufgabe 36 (⋆ ⋆)**:** *Schreiben Sie ein Pythonprogramm, das für kleine n die Werte*

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

ausgibt und erraten Sie einen geschlossenen Ausdruck *(also eine "Summationsformel"). Beweisen Sie die vermutete Summationsformel.*

Hinweis: Entwickeln Sie die rationale Funktion *(also den Quotienten von zwei Polynomen)*

$$\frac{(x+1)^{n+1}-1}{(n+1)\cdot x}$$

in eine Summe.

Aufgabe 37 (⋆)**:** *Beweise durch direkte Rechnung: Für alle $n,k \in \mathbb{Z}$ mit $k \geq 0$ gilt*

$$(-1)^k \binom{-n}{k} = \binom{n+k-1}{k}.$$

Aufgabe 38 (⋆ ⋆)**:** *Sei $1 \leq k < n$. Zeige, daß unter allen 2^{n-1} Kompositionen von n der Teil k genau $(n-k+3)2^{n-k-2}$ mal auftritt. Nimmt man zum Beispiel $n = 4$ und $k = 2$, dann tritt 2 in $2+1+1$, $1+2+1$, $1+1+2$ je einmal, in $2+2$ zweimal auf, also insgesamt 5–mal.*

Aufgabe 39 (⋆ ⋆)**:** *Verwenden Sie das Python–Modul* `sympy` *in einer kleinen Funktion, die für $n \in \mathbb{N}$ das Monom x^n als Linearkombination der* steigenden Faktoriellen[1]

$$x^{\overline{k}} = x \cdot (x+1) \cdots (x+k-1)$$

darstellt, also

$$x^n = \sum_{k=0}^{n} f_{n,k} \cdot x^{\overline{k}}.$$

Z.B. lautet diese Darstellung für $n = 5$:

$$x^5 = 1 \cdot x^{\overline{1}} - 15 \cdot x^{\overline{2}} + 25 \cdot x^{\overline{3}} - 10 \cdot x^{\overline{4}} + x^{\overline{5}}.$$

Die Funktion sollte ein `sympy`*–Symbol x und die Zahl n als Argumente übernehmen, ihr Rückgabewert sollte die Linearkombination in Form eines symbolischen*

[1] `sympy.functions.combinatorial.factorials.RisingFactorial(x, k)`

`sympy`–*Ausdrucks sein. Überprüfen Sie die Korrektheit Ihrer Funktion mit dem* `sympy`–*Befehl* `factor`.

Aufgabe 40 (⋆ ⋆ ⋆)**:** *Zu Beginn Ihrer Studien haben Sie vermutlich zur Übung einige* Potenzsummenformeln *mit Induktion bewiesen, z.B. diese:*

$$\sum_{k=1}^{m} k^3 = 1^3 + 2^3 + \cdots + m^3 = \frac{1}{4} m^2 (1+m)^2$$

Verwenden Sie das Python–Modul `sympy` *in einer kleinen Funktion, die für* $n \in \mathbb{N}$ *die Potenzsummenformel*

$$P_n(m) = \sum_{x=1}^{m} x^n$$

angibt, und begründen Sie genau, warum Ihre Funktion das richtige Ergebnis liefert. Zeigen Sie insbesondere, daß $P_n(m)$ *immer ein* Polynom *vom Grad* $n+1$ *mit führendem Koeffizienten* $\frac{1}{n+1}$ *und* $P_n(0) = 0$ *ist.*

Hinweis: Zeigen Sie, daß der lineare Operator

$$E^{-1}\Delta = I - E^{-1}$$

auf die steigenden Faktoriellen wie folgt wirkt:

$$\left(E^{-1}\Delta\right) x^{\overline{k}} = k \cdot x^{\overline{k-1}} \text{ für } k = 0, 1, \ldots.$$

Dann wählen Sie den Ansatz *(d.h.,* nehmen Sie einfach einmal an*), daß für alle* $n = 0, 1, \ldots$ Polynome $P_n(m)$ *in* m *existieren, sodaß*

$$\sum_{k=1}^{m} k^n = P_n(m)$$

für alle $m = 0, 1, \ldots$ *gilt. Beobachten Sie, daß dann (natürlich)*

$$\left(E^{-1}\Delta\right) P_n(m) = m^n$$

gelten müßte, und verwenden Sie Übungsaufgabe 39.

Aufgabe 41 (⋆ ⋆ ⋆)**:** *Sei* $I(n,k)$ *die Anzahl der Permutationen* $\pi \in \mathfrak{S}_n$ *mit* k *Inversionen (*$I(n,k) = 0$ *für* $k < 0$*).*

(1) *Zeige, daß für* $n \geq k$

$$I(n+1,k) = I(n,k) + I(n+1,k-1)$$

gilt.

(2) *Folgere mittels der obigen Rekursion, daß für* $n \geq k$ *die Zahl* $I(n,k)$ *ein Polynom in* n *vom Grad* k *und führendem Koeffizienten* $1/k!$ *ist (das bedeutet: Es gibt ein Polynom in* x*, dessen* Auswertung *für natürliche Zahlen* $x \mapsto n$ *genau die Zahlen* $I(n,k)$ *ergibt). Für* $n \geq 2$ *gilt beispielsweise* $I(n,2) = \frac{1}{2}(n+1)(n-2)$*. Berechne das Polynom für* $I(n,3)$*. (Hinweis: Induktion nach* k *unter Verwendung des Delta–Operators!)*

Aufgabe 42 ($\star\star$)**:** *Mit Hilfe des Differenzenoperators $\Delta = E - I$, also definiert durch*

$$\Delta p(x) \stackrel{def}{=} p(x+1) - p(x),$$

kann man die Koeffizienten in Entwicklungen der Gestalt

$$q(x) = \sum_k c_k x^{\underline{k}}$$

*berechnen ($x^{\underline{k}} = x(x-1)\cdots(x-k+1)$)**: Es gilt nämlich***

$$c_k = \frac{1}{k!}(\Delta)^k q(x)\,|_{x=0}.$$

(Siehe auch den Abschnitt zum Delta–Operator in der Vorlesung.) Benutzen Sie dies, um für $n > 0$ die Formel

$$x^{\overline{n}} = x(x+1)\cdots(x+n-1) = \sum_{k=0}^{n} \frac{n!}{k!}\binom{n-1}{k-1} x^{\underline{k}}$$

zu beweisen.

Aufgabe 43 ($\star\star$)**:** *Zeige: Die Anzahl aller fixpunktfreien Permutationen von $[n]$ ist gleich*

$$n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - + \cdots + (-1)^n \frac{1}{n!}\right).$$

Aufgabe 44 ($\star\star$)**:** *Sei n eine natürliche Zahl. Die* Eulersche φ–Funktion *von n ist die Anzahl der zu n relativ primen Zahlen k, $1 \le k \le n$. Verwende das Prinzip der Inklusion–Exklusion, um die aus der Zahlentheorie bekannte Formel*

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right)\cdots\left(1 - \frac{1}{p_t}\right)$$

zu beweisen (wobei $p_1, \ldots, p_t$ die Primteiler *von n sind).*

Algorithmen

Einfache Beispiele

Aufgabe 45 ($\star\star$)**:** *Modifizieren Sie die Python–Funktion zum Euklidischen Algorithmus aus der Vorlesung geeignet, um die* Darstellung *des größten gemeinsamen Teilers als* ganzzahlige Linearkombination *zu erhalten: Ihre modifizierte Funktion sollte also für zwei ganze Zahlen* $0 \leq a \leq b$ *nicht nur*

$$d = \mathrm{ggT}\,(a,b)$$

zurückgeben, sondern auch zwei ganze Zahlen λ *und* μ*, sodaß gilt:*

$$d = \mathrm{ggT}\,(a,b) = \lambda \cdot a + \mu \cdot b.$$

Das Travelling Salesperson Problem

Aufgabe 46 ($\star\star$)**:** *Schreiben Sie eine kleine Python–Funktion, die den (einfachen) Algorithmus zur Lösung des Travelling–Salesperson–Problems implementiert: Diese Funktion soll als Argument eine quadratische, symmetrische* `numpy`*–Matrix* W *übernehmen, deren Eintragung* $W_{i,j} = W_{j,i}$ *dem Gewicht der Kante entspricht, die den* i*–ten und* j*–ten Knoten verbindet (die Eintragungen* $W_{i,i}$ *spielen hier keine Rolle).*

Testen Sie die Funktion (nur für kleine *Matrizen* W*: Z.B. können Sie eine mit Zufallszahlen gefüllte* 10×10*–Matrix mit dem Befehl* `W = np.random.rand((10,10))` *erzeugen, und* `W+W.T` *liefert dann eine* symmetrische *Matrix*[2]*).*

Hinweis: `W.shape` *liefert das Tupel "(Anzahl der Zeilen, Anzahl der Spalten)" von* W*, und mit der Funktion* `itertools.permutations` *aus dem Modul* `itertools` *können Sie alle Permutationen der* $\mathfrak{S}_n$ *(als Tupel) erzeugen: Überlegen Sie, wie Sie damit ganz einfach alle* zyklischen *Permutationen der* $\mathfrak{S}_n$ *erzeugen können.*

[2] `W.T` ist die Transponierte der `numpy`–Matrix `W`.

Grundbegriffe der Komplexitätstheorie

Aufgabe 47 ($\star\ \star\ \star$)**:** *Die "definitionsgemäße" Multiplikation zweier $n \times n$–Matrizen $A = \left(a_{i,j}\right)$ und $B = \left(b_{i,j}\right)$ erfordert n^3 Multiplikationen von Einträgen (die Additionen, die dabei natürlich auch vorkommen, berücksichtigen wir nicht, weil sie* sehr viel weniger *Rechenzeit verbrauchen als die Multiplikationen): Denn für jeden der n^2 Einträge $c_{i,j}$ der Matrix $C = A \cdot B$ gilt ja*

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}.$$

Das Laufzeitverhalten (gemessen in einzelnen Multiplikationen) der Matrixmultiplikation ist also polynomial: $n \mapsto n^3$.

Recherchieren Sie im Internet Strassens Algorithmus*, der eine überraschende Verbesserung (Laufzeitverhalten $n^{\log_2 7}$ Multiplikationen) der Matrixmultiplikation bringt, und schreiben Sie ein kleines Python–Programm, das diese Verbesserung für $n = 2^k$ und umsetzt.*

Aufgabe 48 (∞)**:** *Recherchieren Sie im Internet zum Thema "Eternity 2 Puzzle" und verdienen Sie das Preisgeld von USD 2 MIO für dessen Lösung;-); siehe z.B.:*

`http://www.grokcode.com/10/e2-the-np-complete-kids-game-with-the-2-million-prize`

Algorithmische Erzeugung kombinatorischer Objekte

Aufgabe 49 ($\star\ \star$)**:** *Ein Tupel von ganzen Zahlen $(\lambda_1, \lambda_2, \ldots, \lambda_m)$ entspricht einer* Zahl–Partition *von n, wenn $\lambda_i \geq \lambda_{i+1} \geq 0$ für $1 \leq i < m$ gilt und $\sum_{i=1}^{m} \lambda_i = n$.*

Schreiben Sie eine Python-Funktion, die überprüft, ob es sich bei einem Tupel um eine Zahl-Partition handelt. Wenn das Tupel eine Zahl–Partition von n ist, soll diese Darstellung von $n = \lambda_1 + \cdots + \lambda_m$ mit `print()` *ausgegeben werden.*

Kombinieren Sie diese Funktion mit product_with_itertools, um alle Tupel eines (nicht zu groß gewählten) Cartesischen Produkts auszugeben, die auch Zahl–Partitionen sind.

Aufgabe 50 ($\star$)**:** *Zeigen Sie, daß der im Skriptum beschriebene induktive Algorithmus*

> *Sei der Gray–Code G_{n-1} mit $n-1$ Stellen gegeben, dann*
> - *schreibe eine führende Null vor jedes Tupel in G_{n-1},*
> - *und schreibe einen führenden Einser vor jedes Tupel der Liste G_{n-1}* in umgedrehter Reihenfolge*;*
>
> *das Zusammenhängen dieser beiden Teillisten ergibt G_n.*

zur Erzeugung eines n–stelligen klassischen Gray–Codes G_n das richtige Ergebnis liefert.

Schreiben Sie eine Python–Funktion, die diesen Algorithmus implementiert.

Aufgabe 51 ($\star$)**:** *Die Verallgemeinerung des Würfels im $\mathbb{R}^3$ ist der n–dimensionale* Hyperwürfel*: Seine Ecken sind alle Punkte im $\mathbb{R}^n$ mit Koordinaten in $\{0,1\}$; und zwei Ecken p,q sind genau dann durch eine ("geometrische") Kante verbunden, wenn sich die Koordinatenvektoren von p und q in genau einer Koordinate unterscheiden. Ein Hyperwürfel definiert einen Graphen,*

- *dessen Knoten den Ecken entsprechen*
- *und dessen Kanten den "geometrischen Kanten"*

entsprechen: "By abuse of notation" bezeichnet man auch diesen Graphen einfach als Hyperwürfel.

Zeigen Sie: Der Hyperwürfel ist ein bipartiter *Graph.*

Zeigen Sie: Ein (klassischer) Gray–Code mit n Stellen, dessen erstes und letztes Tupel sich auch *in genau einer Koordinate (oder genau einem Bit) unterscheiden, entspricht einem Hamiltonschen Kreis im Hyperwürfel.*

Finden Sie ein einfaches Beispiel für einen Gray–Code, der keinem *Hamiltonschen Kreis im entsprechenden Hyperwürfel entspricht.*

Aufgabe 52 ($\star\ \star\ \star$)**:** *Der Algorithmus zur Erzeugung eines (verallgemeinerten) Gray–Codes (siehe Skriptum) ist an sich ausführlich kommentiert, dennoch aber gar nicht so einfach zu durchschauen: Beweisen Sie, daß er tatsächlich das richtige Ergebnis liefert, und daß das Ranking/Unranking mit den Funktionen im Skriptum richtig implementiert ist (gefragt ist eine* mathematische *Argumentation, nicht bloß ein Test für ein paar konkrete Werte;-).*

Hinweis: Lassen Sie sich vom Computer "vor Augen führen", wie der Algorithmus für kleine Beispiele funktioniert (z.B. für `product_gray_code([2,3,4])`*).*

Aufgabe 53 ($\star\star$)**:** *Beweisen Sie, daß der Algorithmus, den die im Skriptum definierte Python–Funktion* `classical_gray_code` *verwendet, tatsächlich einen (klassischen) Gray–Code erzeugt.*

Teilmengen natürlicher Zahlen mit k Elementen

Aufgabe 54 ($\star\star$)**:** *Betrachten Sie alle 2^n Teilmengen von $[n]$, bilden Sie jeweils das Produkt der Elemente der Teilmengen (das leere Produkt ist definitionsgemäß gleich 1) und summieren Sie diese Produkte; also*

$$s(n) \stackrel{def}{=} \sum_{M \subseteq [n]} \prod_{i \in M} i.$$

Schreiben Sie ein kleines Programm, das die Zahlenwerte $s(n)$ (für kleine n) liefert, erraten Sie einen geschlossenen Ausdruck für $s(n)$ und beweisen Sie diesen (am einfachsten mit einer Rekursion*).*

(Wenn Sie die Lösung sofort sehen, können Sie das "Programmierexperiment" auch weglassen).

Aufgabe 55 (⋆ ⋆ ⋆)**:** *Der in der Vorlesung vorgestellte Algorithmus für das* Unranking *von k–elementigen Teilmengen in* co–lexikographischer *Ordnung hat* natürlich *die Eigenschaft, daß für die zurückgegebene Liste $[v_i]_{i=1}^{k}$ gilt:*

$$v_1 < v_2 < \ldots v_k,$$

denn wir wissen ja, daß dieser Algorithmus genau die Umkehrung des Ranking–*Algorithmus ist, der* aufsteigend geordneten Listen *(mit denen wir Mengen in Python darstellen) eine ganze Zahl ≥ 0 zuordnet. Aus dem Unranking–Algorithmus ist das aber nicht auf den ersten Blick erkennbar: Zeigen Sie diese Tatsache* direkt *durch genaue Betrachtung des Algorithmus (also* ohne *Ihr Vorwissen betreffend den Zusammenhang mit dem Ranking–Algorithmus).*

Permutationen

Aufgabe 56 (⋆ ⋆)**:** *Zeigen Sie als Hilfsresultat für die folgende Frage:*

$$\forall n \in \mathbb{N}\colon \sum_{k=1}^{n} k \cdot k! = (n+1)! - 1.$$

Zeigen Sie: Jede natürliche Zahl n besitzt eine (bis auf "trailing zeroes", also Nullen "am Ende der Summe") eindeutig bestimmte Darstellung der Form

$$n = a_1 \cdot 1! + a_2 \cdot 2! + \cdots + a_k \cdot k!, \qquad \text{mit } 0 \leq a_i \leq i.$$

Ist k die größte Zahl mit $a_k > 0$, so schreibt man $n = (a_k, a_{k-1}, \ldots, a_1)$.

Finden Sie eine möglichst einfache Methode zur Berechnung der Ziffern a_i in dieser Darstellung und implementieren Sie diese in einer kleinen Python–Funktion.

Wie erkennt man an den Ziffern der Darstellung für zwei Zahlen $n = (a_k, \ldots, a_1)$ und $m = (b_l, \ldots, b_1)$ (falls $k \neq l$: durch führende Nullen ergänzen), daß $n < m$ gilt ?

Aufgabe 57 (⋆ ⋆)**:** *Finden Sie eine möglichst einfache Methode, um die k–te Permutation der $\mathfrak{S}_n$* in lexikographischer Ordnung *zu finden. (Die "triviale Methode" — erzeuge alle $n!$ Permutationen der $\mathfrak{S}_n$, ordne sie lexikographisch und wähle das k–te Element — gilt hier natürlich* nicht *als "einfach".)*

Implementieren Sie diese Methode in einer Python–Funktion `unrank_perm_lex` *und schreiben Sie auch die zugehörige "Umkehrfunktion"* `rank_perm_lex`*.*

Hinweis: Verwenden Sie Übungsaufgabe 56.

Aufgabe 58 (⋆ ⋆)**:** *In Aufgabe 57 war die Anordnung der Permutationen durch die lexikographische Ordnung gegeben; eine weitere Möglichkeit der Anordnung besteht im Abzählen von* Inversionen*: Sei π eine Permutation von $[n]$, und sei a_i die Anzahl der Inversionen (k,l) mit $\pi(l) = n-i$ für $i = 1,2,\ldots,n-1$. Zum Beispiel ist für $n = 7$, $\pi = 5\ 3\ 7\ 2\ 1\ 6\ 4$ die entsprechende Folge durch $(a_1,a_2,\ldots,a_6) = (1,0,3,1,3,4)$ gegeben.*

Zeige: Es gilt $0 \le a_i \le i$, $i = 1,2,\ldots,n-1$.

Zeige: Jede solche Folge, kurz Inversionsfolge *genannt, definiert eine* eindeutig bestimmte *Permutation π.*

Aufgabe 59 (⋆ ⋆)**:** *Übungsaufgabe 58 zeigt, daß man jeder ganzen Zahl k mit $0 \le k \le n!-1$ auch auf folgende Weise eine eindeutig bestimmte Permutation π zuordnen (also ein* Unranking *definieren) kann: "Schreibe k in der Gestalt*

$$k = a_1 \cdot 1! + a_2 \cdot 2! + \cdots + a_{n-1} \cdot (n-1)!$$

(vergleiche auch Übungsaufgabe 56) und interpretiere $(a_1,a_2,\ldots,a_{n-1})$ als Inversionsfolge."

Finden Sie eine möglichst einfache Methode, um aus der Inversionsfolge die entsprechende Permutation π zu konstruieren, und implementieren Sie diese Methode in einer Python–Funktion `inv_word2perm`*.*

Verwenden Sie diese Funktion und `itertools.product` *und implementieren Sie einen* Generator `generate_permutations`*, der einen Iterator erzeugt, der die Permutationen in der durch ihre (lexikographisch geordneten) Inversionsfolgen bestimmten Reihenfolge liefert.*

Aufgabe 60 (⋆ ⋆)**:** *Untersuchen Sie die Frage, wieviele Permutationen der $\mathfrak{S}_n$ mit zwei Anstiegen beginnen, mit einer* Monte–Carlo–Simulation *in Python.*

Wenn Sie alles richtig machen, müßten Sie zu der Vermutung gelangen, daß der relative Anteil dieser Permutationen eine konstante rationale Zahl c ist, also

$$\frac{\#\left(\{\pi \in \mathfrak{S}_n : \pi \text{ beginnt mit 2 Anstiegen}\}\right)}{n!} = c \in \mathbb{Q} \text{ für alle } n > 2.$$

Finden Sie einen einfachen Beweis für diese Vermutung.

Aufgabe 61 (⋆ ⋆ ⋆)**:** *Untersuchen Sie die Frage, für wieviele Permutationen $\pi \in \mathfrak{S}_n$*

$$\pi(i) - i = \pi(j) - j \implies i = j$$

gilt. (Anders gesagt: Die Zahlen $\pi(i) - i$ sollen für $i = 1,2,\ldots,n$ alle verschieden *sein; ein Beispiel einer solchen Permutation ist $\pi = (321)$.)*

Schreiben Sie ein kleines Programm, das diese Bedingung für eine feste Permutation π testet*, und untersuchen Sie die Frage für kleine n.*

Zeigen Sie, daß diese Anzahlen immer ungerade *sind. (Hinweis: Betrachten Sie die entsprechenden Permutationsmatrizen und deren Spiegelungen an Haupt– und Nebendiagonale.)*

Aufgabe 62 (⋆ ⋆)**:** *Das Unranking* `unrank_johnson_trotter` *im Skriptum ist nicht ausführlich kommentiert: Begründen Sie, warum die Implementierung korrekt ist, und fügen Sie ausführliche Kommentare in den Code ein.*

Aufgabe 63 (⋆ ⋆)**:** *Beweisen Sie die Behauptung: Jede natürliche Zahl k mit $0 \le k < n!$ hat eine* eindeutige *Darstellung der Form*

$$k = \sum_{j=1}^{n} b_j \cdot \underline{n^{n-j}},$$

wobei für die Koeffizienten der fallenden Faktoriellen gilt:

$$0 \le b_j < j.$$

Hinweis: Verwenden Sie die rekursive Darstellung

$$rank\,(\pi) = n \cdot rank\,(\pi') + \begin{cases} n-j & \text{wenn } rank\,(\pi') \equiv 0 \mod 2, \\ j-1 & \text{wenn } rank\,(\pi') \equiv 1 \mod 2. \end{cases}$$

und Ranking/Unranking für den Johnson–Trotter–Algorithmus.

Zahl–Partitionen

Aufgabe 64 (⋆)**:** *Sei $p\,(n,k)$ die Anzahl der Partitionen von n, deren größter Teil kleinergleich k ist. Zeigen Sie die folgende Rekursion:*

$$\begin{aligned} p\,(n,0) &= [n=0] = \delta_{n,0}, \\ p\,(n,1) &= 1 \text{ für alle } n, \\ p\,(n,k) &= p(n) \text{ für } k \ge n, \\ p\,(n,k) &= \sum_{j=1}^{k} \sum_{i=1}^{\lfloor n/j \rfloor} p\,(n-i\cdot j, j-1)\,. \end{aligned} \tag{1}$$

Aufgabe 65 (⋆ ⋆ ⋆)**:** *Untersuchen Sie für alle $n \in \mathbb{N}$ die Frage, wieviele Zahl–Partitionen von n es gibt, für die*

(a) *alle* geraden *Teile* verschieden *sind,*
(b) *alle Teile Vielfachheit ≤ 3 haben.*

Schreiben Sie ein kleines Programm, das die Anzahlen entsprechend (a) und (b) bestimmt (für kleine n). Betrachten Sie die Anzahlen in einer kleinen Tabelle und leiten Sie daraus eine Vermutung ab: Formulieren und beweisen Sie diese Vermutung.

Hinweis: Der Beweis gelingt in einer verblüffend einfachen Weise mit erzeugenden Funktionen *(die hier als* unendliche Produkte *darstellbar sind): Versuchen Sie, zuerst die Produktdarstellungen der erzeugenden Funktionen für die Anzahlen gemäß (a) bzw. gemäß (b) zu bestimmen, und zeigen Sie dann (in einer wirklich inspirierten Rechnung!), daß diese beiden unendlichen Produkte identisch sind (in dem Sinne, daß sie dieselbe Reihenentwicklung haben); siehe auch das Beispiel 66.*

Aufgabe 66 (⋆ ⋆ ⋆)**:** *Untersuchen Sie für alle* $n \in \mathbb{N}$ *die Frage, wieviele Zahl–Partitionen von* n *es gibt, für die*

(a) *die Anzahl der Teile* ungerade *ist,*
(b) *die Anzahl der Teile* gerade *ist,*
(c) *alle Teile* verschieden *und* ungerade *sind.*

Schreiben Sie ein kleines Programm, das die Anzahlen entsprechend (a), (b) und (c) bestimmt (für kleine n*). Betrachten Sie die Anzahlen in einer kleinen Tabelle und leiten Sie daraus eine Vermutung ab: Formulieren und beweisen Sie diese Vermutung.*

Hinweis: Die Vermutung ist hier ein bißchen komplizierter als im Beispiel 65, aber mit ein bißchen Ausdauer finden Sie sie sicher. Der Beweis gelingt auch hier wieder mit erzeugenden Funktionen*, die als* unendliche Produkte *darstellbar sind, und einer* inspirierten *Rechnung; wie im Beispiel 65: Halten Sie sich die Darstellung*

$$\sum_{n\geq 0} p\left(n\right) z^n = \prod_{k\geq 1} \frac{1}{1-z^k}$$

vor Augen und überlegen Sie, was Sie erhalten, wenn Sie das unendliche Produkt

$$\prod_{k\geq 1} \frac{1}{1+z^k}$$

in eine Potenzreihe entwickeln; siehe auch Beispiel 65.

Aufgabe 67 (⋆ ⋆ ⋆)**:** *A* k*–bounded partition of a positive integer* N *is a way of writing* N *as a sum of positive integers not exceeding* k*.*

A balanceable *partition is a partition that can be further divided into two parts of equal sums. For example,* $3+2+2+2+2+1$ *is a balanceable* 3*–bounded partition of* 12 *since* $3+2+1 = 2+2+2$*. Conversely,* $3+3+3+1$ *is a* 3*-bounded partition of* 10 *which is* not *balanceable.*

Let $f\left(k\right)$ *be the smallest positive integer* N *all of whose* k*–bounded partitions are balanceable. For example,* $f\left(3\right) = 12$ *and*

$$f\left(30\right) \equiv 179092994 \pmod{1\,000\,000\,007}.$$

Find $f\left(10^8\right)$*. Give your answer modulo* $1\,000\,000\,007$*.*

Mengen–Partitionen

Aufgabe 68 (⋆ ⋆)**:** *Geben Sie einen bijektiven Beweis der folgenden Rekursionsgleichung für die Bell–Zahlen B_n (B_n ist die Anzahl aller Mengen–Partitionen von $[n]$):*

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k.$$

Aufgabe 69 (⋆ ⋆ ⋆)**:** *Welcher Satz über die Darstellbarkeit von ganzen Zahlen k, $0 \leq k < B_n$, ergibt sich aus den Algorithmen zu Ranking/Unranking von Funktionen mit beschränktem Wachstum?*

Rekursionen und formale Potenzreihen

Rekursionen

Aufgabe 70 (⋆ ⋆)**:** *Zeigen Sie, daß für die Fibonacci–Zahlen F_n für $n \geq 2$ die Matrixidentität*

$$\begin{pmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{n-1}$$

gilt, und folgern Sie daraus

$$F_n F_{n-2} - F_{n-1}^2 = (-1)^{n-1}.$$

Aufgabe 71 (⋆ ⋆)**:** *Drücken Sie die folgenden Zahlen durch Fibonaccizahlen aus:*

(a) *Anzahl aller Kompositionen von n, deren Teile alle kleiner oder gleich 2 sind,*
(b) *Anzahl aller Kompositionen von n, deren Teile alle größer oder gleich 2 sind,*
(c) *Anzahl aller Kompositionen von n in ungerade Teile.*

Begründen Sie Ihre Darstellung mit Fibonaccizahlen in allen drei Punkten genau.

Aufgabe 72 (⋆ ⋆)**:** *Zeigen Sie: Jede natürliche Zahl $m > 0$ kann* eindeutig *durch eine Summe von* nicht aufeinanderfolgenden *Fibonacci–Zahlen F_n mit $n \geq 2$ dargestellt werden (d.h.: In dieser Darstellung kommen niemals zwei aufeinanderfolgende Fibonacci–Zahlen F_k, F_{k+1} vor). Zum Beispiel:*

$$1 = F_2, 2 = F_3, 3 = F_4, 4 = F_2 + F_4, 5 = F_5, 6 = F_2 + F_5, 7 = F_3 + F_5, 8 = F_6, \ldots$$

Schreiben Sie eine Python–Funktion, die diese Darstellung liefert (der Rückgabewert sollte eine Liste der Summanden sein).

Aufgabe 73 (⋆ ⋆)**:** *Geben Sie für die Summe aller* geraden *Fibonacci–Zahlen bis zum Index n, also die Summe der Elemente der Menge*

$$\{F_k \colon k \leq n \text{ und } F_k \equiv 0 \pmod 2\},$$

eine (möglichst einfache) geschlossene Form *an (die insbesondere kein Summenzeichen mehr enthält).*

Aufgabe 74 (⋆ ⋆)**:** *Zeigen Sie, daß die Catalan–Zahlen C_n durch die Rekursion*

$$C_0 = 1 \text{ und } C_n = \sum_{k=0}^{n-1} C_k \cdot C_{n-k-1} \text{ für } n > 0$$

gegeben sind.

Hinweis: Am einfachsten durch eine kombinatorische Überlegung — denken Sie an die Triangulierungen eines $(n+2)$–Ecks!

Verwenden Sie die `numpy`*–Funktion* `inner`*, die das innere Produkt zweier Vektoren liefert, um eine (möglichst elegante!) Python–Funktion zu implementieren, die die Catalan–Zahlen C_k für $k = 0, 1, \ldots, n$ mit dieser Rekursion berechnet und in einen* `numpy`*–Vektor der Länge $n+1$ schreibt.*

Erzeugende Funktionen und Formale Potenzreihen

Aufgabe 75 (⋆)**:** *Verwenden Sie die* `sympy`*–Funktion* `apart` *für eine Funktion, die für einen gegebenen Quotienten von Polynomen $p(x)/q(x)$ die* Partialbruchzerlegung *liefert.*

Berechnen Sie mit ihrer Funktion die Partialbruchzerlegung für die rationale Funkion

$$\frac{1 + 2x + 3x^2 + 5x^3}{(x-3)(x-5)(x-8)}$$

Hinweis: Schauen Sie sich die Beispiele auf `https://docs.sympy.org` *an.*

Aufgabe 76 (⋆)**:** *Bestimmen Sie das multiplikative Inverse für die formale Potenzreihe*

$$f(z) = \sum_{n=0}^{\infty} (n+1) z^n.$$

Aufgabe 77 (⋆ ⋆)**:** *Zeigen Sie für die Fibonaccizahlen F_n die Identität*

$$\sum_{k=0}^{n+2} F_k F_{n-k+2} = \sum_{k=0}^{n} (k+1) F_{k+2} (-2)^{n-k}.$$

Aufgabe 78 (⋆ ⋆)**:** *Sei $F(z) = \sum_{n=0}^{\infty} n!\, z^n$.*

(1) *Eine Permutation $(a_1, a_2, \ldots, a_n)$ von $[n]$ heißt unzerlegbar, falls n die kleinste natürliche Zahl j ist, für die $\{a_1, a_2, \ldots, a_j\} = \{1, 2, \ldots, j\}$ gilt. Sei $f(n)$ die Anzahl aller unzerlegbaren Permutationen von $[n]$. Zeige:*

$$\sum_{n=1}^{\infty} f(n) z^n = 1 - F(z)^{-1}\ .$$

(2) *In einer Permutation* $(a_1, a_2, \ldots, a_n)$ *von* $[n]$ *heißt* a_i *ein* $starker\ Fixpunkt$, *falls (1)* $j < i \Rightarrow a_j < a_i$, *und (2)* $j > i \Rightarrow a_j > a_i$. *Sei* $g(n)$ *die Anzahl aller Permutationen von* $[n]$, *die keinen starken Fixpunkt besitzen. Zeige:*

$$\sum_{n=0}^{\infty} g(n) z^n = F(z) (1 + zF(z))^{-1} .$$

Aufgabe 79 ($\star\star$)**:** *Finde einen geschlossenen Ausdruck für die Glieder der Folge* $(a_n)_{n\in\mathbb{N}_0}$, *die der Rekursion* $a_n = a_{n-1} + 2a_{n-2} + (-1)^n$ *mit den Anfangsbedingungen* $a_0 = a_1 = 1$ *genügt.*

Aufgabe 80 ($\star\star$)**:** *Finde einen geschlossenen Ausdruck für die Glieder der Folge* $(a_n)_{n\in\mathbb{N}_0}$, *die der Rekursion* $a_n = -a_{n-1} + 5a_{n-2} - 3a_{n-3}$ *mit den Anfangsbedingungen* $a_0 = 7$, $a_1 = -12$, $a_2 = 49$ *genügt.*

Aufgabe 81 ($\star\star$)**:** *Finde einen geschlossenen Ausdruck für die Glieder der Folge* $(a_n)_{n\in\mathbb{N}_0}$, *die der Rekursion* $a_n = 6a_{n-1} - 4a_{n-2}$ *mit den Anfangsbedingungen* $a_0 = 1$, $a_1 = 3$ *genügt. Zeige, daß* $a_n = \lceil \frac{(3+\sqrt{5})^n}{2} \rceil$.

Aufgabe 82 ($\star\star$)**:** *Löse die Rekursion* $a_n = 3a_{n-1} - a_{n-2}$ *mit den Anfangsbedingungen* $a_0 = 1$, $a_1 = 2$. *Hat das etwas mit Fibonaccizahlen zu tun?*

Aufgabe 83 ($\star\star\star$)**:** *Auf wieviele Arten kann ein Turm, der die Form eines* $2\times2\times n$*-Quaders besitzt, aus* $2\times1\times1$*-Ziegeln aufgebaut werden?*

Anleitung: Sei a_n *die gesuchte Zahl. Stellen wir uns die Grundfläche des Turms parallel zu den Nord–Süd– und Ost–West–Achsen ausgerichtet, und sei* b_n *die Anzahl, einen* $2\times2\times n$*–Turm, dem in der obersten Ebene "der östliche Ziegel in Nord–Süd–Richtung" fehlt, aus solchen Ziegeln zusammenzusetzen. Zeige zunächst die Rekurrenzen*

$$\begin{aligned} a_n &= 2a_{n-1} + 4b_{n-1} + a_{n-2} + [n = 0] , \\ b_n &= a_{n-1} + b_{n-1} . \end{aligned}$$

Leite daraus 2 Gleichungen für die entsprechenden erzeugenden Funktionen her und gewinne daraus die gesuchte erzeugende Funktion.

Aufgabe 84 ($\star\star$)**:** *Bestimme jene eindeutig bestimmte Folge* $(a_n)_{n\in\mathbb{N}_0}$, *für die* $a_0 = 1$ *und*

$$\sum_{k=0}^{n} a_k a_{n-k} = 1$$

für alle $n \in \mathbb{N}$ *gilt.*

Aufgabe 85 ($\star\star$)**:** *Sei* $p^*(n)$ *die Anzahl aller Zahl–Partitionen von* n, *bei denen*

- *der Teil* 1 *beliebig oft vorkommen kann,*
- *der Teil* 3 *höchstens* 4*–mal vorkommen darf,*
- *der Teil* 7 *entweder gar nicht oder genau* 2*–mal vorkommen darf,*
- *und sonst keine andren Teile vorkommen dürfen.*

Wie sieht die erzeugende Funktion der Folge $\left(p^*\left(n\right)\right)_{n=0}^{\infty}$ *aus?*

Aufgabe 86 (⋆ ⋆)**:** *Beweise, daß die Anzahl der Zahl–Partitionen von* n*, deren Summanden alle nicht durch 3 teilbar sind, gleich ist der Anzahl der Partitionen, in denen kein Summand mehr als zweimal erscheint.*
Beispiel: $4 = 4,\ 2+2,\ 2+1+1,\ 1+1+1+1$ *beziehungsweise* $4 = 4,\ 3+1,\ 2+2,\ 2+1+1$.
(Anleitung: Drücke beide Anzahlen unter Verwendung des Prinzipes der Inklusion–Exklusion durch die Partitionsfunktion p *aus, wobei* $p\left(n\right)$ *die Anzahl* aller *Partitionen von* n *bezeichnet.)*

Aufgabe 87 (⋆ ⋆)**:** *Untersuchen Sie experimentell die Folgen der Anzahlen . . .*

(a) *. . . der Zahl–Partitionen von* n*, deren Teile alle kongruent* 1 *oder* 4 *modulo* 5 *sind (also* $\lambda_i \equiv \pm 1 \pmod 5$*),*
(b) *. . . der Zahl–Partitionen von* n*, die keine zwei gleiche oder aufeinanderfolgende Teile haben (also* $\lambda_i - \lambda_{i+1} \geq 2$*).*

Formulieren Sie eine Vermutung, die sich aus dem Experiment (wenn Sie es richtig durchgeführt haben;-) ergibt.

Aufgabe 88 (⋆ ⋆ ⋆)**:** *Beweisen Sie die Vermutung aus der vorigen Aufgabe.*

Hinweis: Beginnen Sie mit der erzeugenden Funktion der ersten Zahlenfolge (Teile kongruent 1 *oder* 4 *modulo* 5*). Wenn Sie Schwierigkeiten mit der erzeugenden Funktion für die zweite Zahlenfolge haben, könnten Sie in* §*19.13 im Lehrbuch "Zahlentheorie" von Hardy und Wright nachschauen; dort finden Sie auch die Tricks, die nötig sind, um die Identität für die erzeugenden Funktionen zu zeigen.*

Aufgabe 89 (⋆ ⋆)**:** *Eine Zahl–Partition* λ *heißt* selbstkonjugiert, *wenn sie mit ihrer konjugierten* λ' *übereinstimmt.*

Untersuchen Sie experimentell die Folgen der Anzahlen . . .

(a) *. . . der selbstkonjugierten Zahl–Partitionen von* n*,*
(b) *. . . der Zahl–Partitionen von* n *mit ausschließlich* ungeraden *und* paarweise verschiedenen *Teilen.*

Formulieren Sie eine Vermutung, die sich aus dem Experiment (wenn Sie es richtig durchgeführt haben;-) ergibt.

Aufgabe 90 (⋆ ⋆)**:** *Beweisen Sie die Vermutung aus des vorigen Aufgabe.*

Hinweis: Betrachten Sie das Ferrers Diagramm einer selbskonjugierten Partition und versuchen Sie, es in verschiedene ungerade "Streifen" zu zerlegen.

Graphen und Netzwerke

Teilgraphen und Zusammenhang

Aufgabe 91 (⋆)**:** *Zeige, daß alle vollständigen Graphen* $\mathbf{K}_n$ *(für* $n > 0$*) stets zusammenhängend sind.*

Zeige, daß ein Graph genau dann Zusammenhangsgrad 0 *hat, wenn er entweder unzusammenhängend ist oder gleich dem vollständigen Graphen* $\mathbf{K}_1$ *ist.*

Zeige, daß der Zusammenhangsgrad des vollständigen Graphen $\mathbf{K}_n$ *gleich* $n-1$ *ist.*

Zeige, daß jeder induzierte *Teilgraph eines vollständigen Graphen wieder ein vollständiger Graph ist.*

Zeige, daß ein Graph $\mathbf{G}\,(V,E)$ *mit* $\#\,(V\,(\mathbf{G})) \geq 3$ *zweifach zusammenhängend ist, wenn es für je zwei Knoten* v *und* w *aus* V *einen* Kreis *(das ist eine geschlossene Wanderung* $v_0,\ldots,v_n$ *in* $\mathbf{G}$*, wobei* $v_i \neq v_j$ *für alle* $i \neq j$ *mit der einzigen Ausnahme* $v_0 = v_n$*) gibt, der* v *und* w *enthält.*

Aufgabe 92 (⋆ ⋆)**:** *Sei* $\mathbf{G}$ *ein einfacher Graph mit* n *Knoten; und sei für je zwei Knoten* v_1*,* v_2*, die nicht durch eine Kante verbunden sind, die Summe ihrer Grade mindestens* $n-1$*. Zeige:* $\mathbf{G}$ *ist zusammenhängend.*

Die Adjazenzmatrix

Aufgabe 93 (⋆)**:** *Sei die Adjazenzmatrix* A *eines Graphen* $\mathbf{G}$ *mit* n *Knoten gegeben. Zeigen Sie: Die Frage, ob* $\mathbf{G}$ *ein Dreieck enthält, läßt sich durch Betrachtung der Matrizen* A *und* $A^2 = A \cdot A$ *entscheiden, und der zeitaufwendigste Teil eines entsprechenden "Entscheidungsalgorithmus" ist die Matrixmultiplikation* $A \cdot A$ *mit Laufzeitverhalten* $O\left(n^3\right)$*.*

Finden Sie einen Algorithmus mit einem besseren *Laufzeitverhalten als* $O\left(n^3\right)$*.*

Hinweis: Siehe auch Aufgabe 47.

Bäume und Wälder

Aufgabe 94 (⋆)**:** *Sei G ein einfacher Graph: Ein kürzester Weg, der zwei Knoten $p,q \in V(G)$ verbindet, heißt eine* Geodäte[3] *Geodäten müssen im allgemeinen nicht eindeutig sein. Wenn es für je zwei Knoten p und q eine* eindeutige *Geodäte gibt, die p und q verbindet, dann nennt man G* geodätisch*.*

Zeigen Sie: Ein einfacher Graph G auf n Knoten ist ein Baum genau dann, wenn er eine der folgenden äquivalenten Eigenschaften hat:

- *G ist kreisfrei (d.h., es gibt keine Kreise in G) und hat genau $n-1$ Kanten.*
- *Für je zwei Knoten in G gibt es* genau einen *Weg, der sie verbindet.*
- *G ist* bipartit *und* geodätisch*.*

Aufgabe 95 (⋆)**:** *Schätzen Sie das Laufzeitverhalten des Algorithmus für das Auffinden eines spannenden Baumes, der im Skriptum vorgestellt wurde, im* Worst–Case *(nur grob) ab; in folgendem Sinn: Wieviele Schleifen–Durchläufe benötigt der Algorithmus im schlimmsten denkbaren Fall?*

Aufgabe 96 (⋆ ⋆)**:** *Sei G ein einfacher zusammenhängender Graph. Für $p,q \in V(G)$ sei die* Distanz *$d(p,q)$ zwischen p und q definiert als die Länge einer* Geodäte *(also eines kürzesten Weges), die p und q verbindet ($d(p,p) = 0$).*

Implementieren Sie einen Algorithmus, der für einen gegebenen Knoten $m \in V(G)$ sukzessive die einzelnen "Kugelschalen"

$$K_r \stackrel{def}{=} \{v \in V(G) : d(v,m) = r\}$$

(als Listen) bestimmt (also zuerst K_1, dann K_2, usw.) und (als Liste von Listen) zurückgibt.

Aufgabe 97 (⋆ ⋆)**:** *Sei G ein einfacher zusammenhängender Graph, der ein* Labyrinth *darstellt: Sei $s \in V(G)$ der Startpunkt, und sei $e \in V(G)$ der Endpunkt, gesucht ist ein* Weg *von s nach e.*

Implementieren Sie einen Algorithmus, der "durch Trial and Error*" einen solchen Weg findet, sodaß während der Suche keine Kante (also kein "Gang des Labyrinths") öfter als zweimal betrachtet ("durchlaufen") wird .*

Aufgabe 98 (⋆)**:** *Modifizieren Sie die Funktion* `find_spanning_tree` *die im Skriptum vorgestellt wurde, geeignet, um Kruskals Algorithmus zu implementieren. (Nehmen Sie dazu an, daß die Kantengewichte als* Einträge *in der Adjazenzmatrix des gewichteten Graphen gegeben sind.)*

[3] Geodäte ist an sich ein Begriff aus der Differentialgeometrie.

Eulersche Graphen.

Aufgabe 99 (⋆ ⋆ ⋆)**:** *Schreiben Sie eine Python–Funktion, die eine symmetrische Adjazenzmatrix A als Argument übernimmt und eine* Eulersche Wanderung *(codiert als Folge der Knoten–Indizes) zurückgibt, wenn der der Matrix A entsprechende einfache Graph mit Schlingen ein* Eulerscher Graph *ist.*

Digraphen und Netzwerke

Aufgabe 100 (⋆ ⋆)**:** *Sei $\mathbf{G}\,(V, E)$ ein Graph, in dem jeder Knoten geraden Grad hat. Zeige, daß man den Kanten von $\mathbf{G}$ eine Orientierung aufprägen kann, sodaß für jeden Knoten der Ausgangsgrad gleich dem Eingangsgrad ist.*

Aufgabe 101 (⋆ ⋆)**:** *Beweise die folgende Variante des Satzes von Euler:*

Ein Digraph $\mathbf{D}$ ohne isolierte Knoten hat eine (orientierte) geschlossene Eulersche Wanderung genau dann, wenn sein zugrundeliegender Graph $\mathbf{G}$ zusammenhängend ist und für jeden Knoten der Eingangsgrad gleich dem Ausgangsgrad ist.

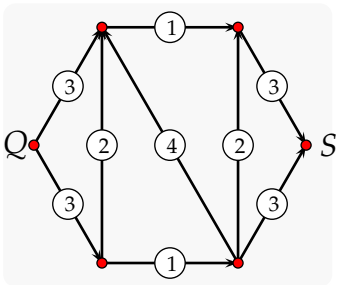
Aufgabe 102 (⋆)**:** *Es sei ein Fluss f in einem Netzwerk gegeben. Zeige, daß der Nettowert des Flusses "aus der Quelle Q heraus", also*

$$\sum_{e\in\{Q\}_{out}} f\,(e) - \sum_{e\in\{Q\}_{in}} f\,(e)\,,$$

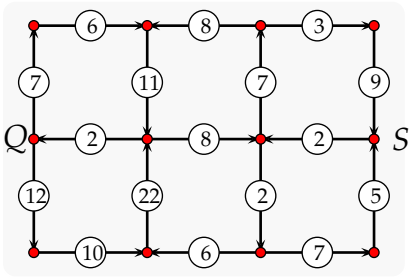
immer gleich ist dem Nettowert des Flusses "in die Senke S hinein", also

$$\sum_{e\in\{S\}_{in}} f\,(e) - \sum_{e\in\{S\}_{out}} f\,(e)\,.$$

Aufgabe 103 (⋆)**:** *Bestimme Flüsse mit maximaler Stärke und Schnitte mit minimaler Kapazität im folgenden Netzwerk (die Kapazitäten der Kanten sind in den kleinen Kreisen eingetragen):*



Aufgabe 104 (⋆)**:** *Betrachte das folgende Netzwerk: Q bezeichnet die Quelle, S die Senke, die Kapazitäten der gerichteten Kanten sind in die kleinen Kreise eingetragen.*



Finde einen maximalen Fluß in diesem Netzwerk und begründe (kurz), warum dieser maximal ist.

Aufgabe 105 (⋆ ⋆ ⋆)**:** *Implementieren Sie den Algorithmus aus dem Beweis des Satzes von Ford–Fulkerson in Python (nur für Netzwerke mit* ganzzahligen *Kapazitäten).*

Die Sätze von Menger, König und Hall

Aufgabe 106 (⋆)**:** *Ein Matching in einem Graphen heißt* perfekt, *wenn jeder Knoten mit (genau) einer der Kanten des Matchings inzident ist. Zeige: Die Anzahl aller perfekten Matchings im vollständigen Graphen K_{2n} ist*

$$\frac{(2n)!}{2^n n!}.$$

Aufgabe 107 (⋆)**:** *Zeige: Ein Graph $\mathbf{G}$ ist genau dann bipartit, wenn jede geschlossene Wanderung in $\mathbf{G}$ gerade Länge hat.*

Aufgabe 108 (⋆ ⋆ ⋆)**:** *Jede* doppelt stochastische $n \times n$ *Matrix M kann als* Konvexkombination *von* höchstens $n^2 - n + 1$ *Permutationsmatrizen dargestellt werden: Arbeiten Sie die folgende Skizze eines Beweises für diese Behauptung aus.*

Der Beweis basiert auf folgender Idee: Für alle $\pi \in \mathfrak{S}_n$ betrachte

$$\min(M, \pi) \stackrel{\text{def}}{=} \min\left\{M_{i,\pi(i)} : 1 \leq i \leq n\right\}$$

und wähle

$$\lambda \stackrel{\text{def}}{=} \max\{\min(M, \pi) : \pi \in \mathfrak{S}_n\}.$$

Wenn $\lambda = \min(M, \tau) > 0$, *dann ist*

$$M - \lambda \cdot M(\tau)$$

das $(1-\lambda)$–Fache einer doppelt stochastischen Matrix M', die mindestens eine *Eintragung 0* mehr *hat als M.*

Solange M' nicht die Null–Matrix ist, wiederholen wir den Schritt für M': Klarerweise bricht der Algorithmus ab, und zwar nach höchstens $n^2 - n + 1$ Schritten.

Der springende Punkt ist, daß für jede doppelt stochastische Matrix M tatsächlich immer ein τ mit $\min(M, \tau) > 0$ existiert: Dazu betrachten wir einen bipartiten Graphen *mit Knotenmenge $R \dot\cup C$, wobei R den Zeilen und C den Spalten von M entspricht, und wo $r \in R$ und $c \in C$ genau dann durch eine Kante verbunden*

sind, wenn $M_{r,c} \neq 0$. *Eine Permutation* τ *mit* $\min(M, \tau) > 0$ *entspricht einem* Matching *in diesem bipartiten Graphen, oder einer "Heirat" im Sinne des Satzes von Hall: Es genügt also zu zeigen, daß die* Bedingung *des Satzes von Hall für diesen bipartiten Graphen erfüllt ist.*

Planare Graphen, Polyedersatz und 5–Farbensatz

Aufgabe 109 ($\star\ \star\ \star$)**:** *Zeigen Sie: Jeder einfache Graph* $\mathbf{G}$ *kann in* $\mathbb{R}^3$ *so eingebettet werden, daß jeder Kante ein* Geradenstück *entspricht.*

Hinweis: Ordnen Sie den Knoten von $\mathbf{G}$ *Punkte auf der Kurve* $\left(t, t^2, t^3\right)$ *zu.*

Aufgabe 110 ($\star\ \star$)**:** *Finde einen Eulerschen Polyedersatz für unzusammenhängende planare Graphen.*

Hinweis: In der entsprechenden Gleichung wird die Anzahl der Komponenten des Graphen eine Rolle spielen.

Aufgabe 111 ($\star\ \star\ \star$)**:** *Angenommen, sämtliche Flächen eines (endlichen) zusammenhängenden planaren Graphen* $\mathbf{G}$ *sind durch Kreise (also geschlossene Wege) mit derselben Anzahl* n *von Kanten begrenzt, und alle Knoten haben denselben Grad* d.

Drücke die Anzahl der Flächen von $\mathbf{G}$ *durch* d *und* n *aus. Wieviele Graphen mit diesen Eigenschaften gibt es?*

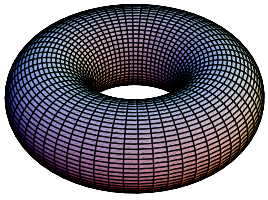
Hinweis: Denken Sie an Platonische Körper!

Aufgabe 112 ($\star\ \star$)**:** *Zeige:* $\mathbf{K}_5$ *und* $\mathbf{K}_{3,3}$ *kann man auf dem* Torus *— das ist die* 2*–dimensionale Mannigfaltigkeit, die duch die Gleichung*

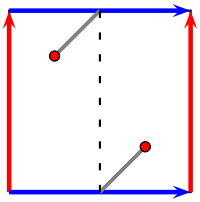
$$\left(c - \sqrt{x^2 + y^2}\right)^2 + z^2 = a^2$$

mit $c > a$ *beschrieben ist — einbetten.*

Hinweis: Den Torus kann man sich als "Fahrradschlauch" (ohne Ventilöffnung) vorstellen.



In der Topologie stellt man sich das auch so vor: Durch zwei kreisförmige Schnitte wird der Torus zu einem Rechteck — um ihn wieder zusammenzusetzen, müßte man gegenüberliegende Seiten des Rechtecks "gleichsinnig" miteinander verkleben (also so, daß die verklebten Pfeile in der folgenden Graphik in dieselbe Richtung zeigen). Die hier gezeichnete Kante ist durch den "waagrechten" Schnitt durchtrennt; im zusammengeklebten Torus verbindet sie die beiden Knoten.



Gewichtserhaltende vorzeichenumkehrende Involutionen

Das Prinzip der Inklusion–Exklusion (revisited)

Aufgabe 113 ($\star\ \star\ \star$)**:** *Zeigen Sie als kleine Vorübung mit einer vorzeichenumkehrenden Involution:*

$$\sum_{\pi\in\mathfrak{S}_n} sgn\left(\pi\right) = \left[n<2\right].$$

Nun betrachten wir eine Multi–Teilmenge von $[k]$*, also eine "Menge, in der Elemente mit gewissen Vielfachheiten auftreten": Wir können solche Multi–Teilmengen mit* k*–Tupeln* $(v_1, v_2, \ldots, v_k)$ *codieren, wo* v_i *die Vielfachheit der Zahl* i *entspricht. Eine Anordnung der Elemente einer solchen Multi–Teilmenge nennen wir eine Multi–Permutation; wir bezeichnen die Menge aller Multi–Permutationen der durch* $(v_1, v_2, \ldots, v_k)$ *"codierten" Multi–Teilmenge mit* $\mathcal{M}\left(v_1, v_2, \ldots, v_k\right)$*. Die Anzahl aller solchen Multi–Permutationen ist*

$$n\left(v_1, v_2, \ldots, v_k\right) \stackrel{def}{=} \#\left(\mathcal{M}\left(v_1, v_2, \ldots, v_k\right)\right) = \frac{\left(v_1+v_2+\cdots+v_k\right)!}{v_1!\cdot v_2!\cdots v_k!}.$$

Sei $m = v_1+v_2+\cdots+v_k$*: Eine Inversion einer Multi–Permutation* $\pi\in\mathcal{M}\left(v_1, v_2, \ldots, v_k\right)$ *ist ein Paar* $1\leq i<j\leq m$ *mit* $\pi_i > \pi_j$*, die Anzahl* aller *Inversionen von* π *bezeichnen wir mit* $inv\,\pi$*, und das Signum von* π *ist dann definiert als*

$$sgn\left(\pi\right) \stackrel{def}{=} \left(-1\right)^{inv\,\pi}.$$

Zeigen Sie: Die Summe

$$\sum_{\pi\in\mathcal{M}\left(v_1, v_2, \ldots, v_k\right)} sgn\left(\pi\right)$$

ergibt 0*, wenn es (mindestens) 2* ungerade *Vielfachheiten* v_i*,* v_j *gibt (*$i\neq j$*), andernfalls ergibt sie*

$$n\left(\left\lfloor v_1/2\right\rfloor, \left\lfloor v_2/2\right\rfloor, \ldots, \left\lfloor v_k/2\right\rfloor\right).$$

Aufgabe 114 ($\star\ \star$)**:** *Beweisen Sie die folgende Identität einmal mit erzeugenden Funktionen und einmal mit einer vorzeichenumkehrenden Involution:*

$$\sum_{k=p}^{n} \binom{n}{k}\binom{k}{p}\left(-1\right)^k = \left[n=p\right]\cdot\left(-1\right)^p.$$

Aufgabe 115 (⋆ ⋆)**:** *Beweisen Sie die folgende Identität einmal mit erzeugenden Funktionen und einmal mit einer vorzeichenumkehrenden Involution:*

$$\sum_{j=0}^{m} \binom{n}{j}\binom{n}{m-j}(-1)^j = \begin{cases} 0 & : \text{ für } m \text{ ungerade,} \\ \binom{n}{k}(-1)^k & : \text{ für } m = 2k. \end{cases}$$

Der Eulersche Pentagonalzahlensatz

Aufgabe 116 (⋆ ⋆)**:** *Für die Anzahl $p(n)$ aller Zahl–Partitionen von n gilt:*

$$\sum_{k=-\infty}^{\infty} (-1)^k \, p\left(n - \left(3k^2 + k\right)/2\right) = 0. \tag{2}$$

Verwenden Sie die Rekursion (2) *(die Summe ist nur* formal *unendlich!) in einer Python–Funktion, die die ersten m Glieder der Folge $(p(n))_{n\geq 0}$ liefert.*

Hinweis: Speichern Sie bereits bekannte Folgenglieder in einem eindimensionalen `numpy`*–array der Länge m.*

Datenstrukturen und Algorithmen

Bestmögliche Algorithmen.

Aufgabe 117 (⋆)**:** *Gegeben seien 3 äußerlich völlig gleiche Münzen, wovon genau eine falsch ist. Wir wissen zwar, daß die falsche Münze ein anderes Gewicht hat als die richtige, wir wissen aber* nicht*, ob sie schwerer oder leichter ist. Das einzige Instrument, mit dem wir die falsche Münze identifizieren können, ist eine gewöhnliche Balkenwaage. Die Aufgabe besteht darin, die falsche Münze mit möglichst wenigen Wägungen zu identifizieren* und *festzustellen, ob sie schwerer oder leichter ist.*

Was ist hier die geringste Anzahl von Wägungen, mit der wir in jedem Fall *die falsche Münze identifizieren und feststellen können, ob sie leichter ist oder schwerer?*

Wurzelbäume (Rooted Trees)

Aufgabe 118 (⋆)**:** *Seien n und q positive natürliche Zahlen mit $q \geq 2$. Zeige: Es gibt* genau dann *einen vollständigen q–Baum mit genau n Blättern, wenn $(q-1) \mid (n-1)$.*

Aufgabe 119 (⋆)**:** *Sei T ein vollständiger 2–Baum mit n Blättern, $e(T)$ bezeichne die Summe der Längen der Blätter, $i(T)$ die Summe der Längen der inneren Knoten. Zeige:*

$$e(T) = i(T) + 2(n-1).$$

Aufgabe 120 (⋆)**:** *Zeige: Jeder 2–Baum mit n Blättern hat einen Teilbaum mit k Blättern, wobei $\frac{n}{3} \leq k \leq \frac{2n}{3}$ gilt.*

Suchalgorithmen

Aufgabe 121 (⋆ ⋆)**:** *Löse das Wägeproblem für n Münzen, wenn bekannt ist, daß die falsche Münze schwerer ist.*

Aufgabe 122 (⋆ ⋆)**:** *Sei S ein Suchraum mit m Elementen, der aus einer geordneten Liste von paarweise verschiedenen Zahlen besteht, und sei* $\mathbf{B}$ *ein* binary search tree *für S.*

Zeigen Sie: Die m Blätter von $\mathbf{B}$ *haben paarweise verschiedene Labels (sie entsprechen also bijektiv dem Suchraum S).*

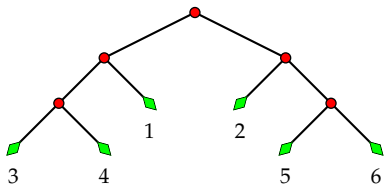
Jedes Blatt in $\mathbf{B}$*, wird durch einen* eindeutigen *Weg von der Wurzel erreicht, und jeden solchen Weg können wir durch eine Folge aus $\{0,1\}$ codieren, wo wir für jeden Schritt "nach links" einen Nuller schreiben und für jeden Schritt "nach rechts" einen Einser.*

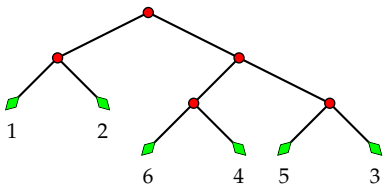
Zeigen Sie: Die lexikographische Ordnung der 01–Folgen, die auf diese Weise jedem Blatt bijektiv zugeordnet sind, entspricht genau der Ordnung der Labels der Blätter.

Aufgabe 123 (⋆)**:** *Es seien $m \cdot n$ Leute in einem $m \times n$-Rechteck angeordnet. Wir sollen die unbekannte Person X durch Fragen der Art „Ist X in der i-ten Zeile?", beziehungsweise „Ist X in der j-ten Spalte?" finden. Wieviele Fragen benötigen wir im Durchschnitt?*

Aufgabe 124 (⋆)**:** *Gegeben sei ein Suchraum mit 4 Elementen $\{x_1, x_2, x_3, x_4\}$, wobei die zugehörigen Wahrscheinlichkeiten p_1, p_2, p_3, p_4 durch $p_1 = 0.5$, $p_2 = 0.3$ und $p_3 = p_4 = 0.1$ gegeben sind. Konstruiere dafür einen Suchbaum mit minimaler durchschnittlicher Blattlänge; unter der Annahme, daß ein Test mit genau zwei möglichen Ausgängen (ja/nein) zur Verfügung steht.*

Aufgabe 125 (⋆)**:** *Gegeben ist die Verteilung $(\frac{30}{100}, \frac{20}{100}, \frac{15}{100}, \frac{14}{100}, \frac{11}{100}, \frac{10}{100})$ für die Blätter $1, 2, \ldots, 6$. Zeige, daß die folgenden binären Suchbäume (2–Bäume) optimal sind. Nur einer ist ein Huffman–Baum, welcher?*





Aufgabe 126 (⋆)**:** *Gegeben sei der Suchraum $\{1, 2, \ldots, 10\}$, in dem ein (zunächst unbekanntes) Element zu suchen ist. Für die Suche stehen Tests zur Verfügung, die den Suchraum* immer *in drei* beliebige *Teile zerlegen können. Weiters ist von vornherein bekannt, daß Element i mit Wahrscheinlichkeit p_i das Gesuchte ist:*

i	1	2	3	4	5	6	7	8	9	10
p_i (in %)	30	22	14	12	9	4	4	2	2	1

Konstruiere in dieser Situation einen optimalen Suchalgorithmus (Suchbaum) im Sinne der Average–Case–Analysis. (Beim Aufbauen des Baumes tritt der Fall aus der Bemerkung in der Vorlesung ein.)

Sortieralgorithmen

Aufgabe 127 (⋆ ⋆ ⋆)**:** *Bestimme die optimalen Sortieralgorithmen für $n = 6,7,8$. (Hinweis: Setze voraus, daß es einen Sortieralgorithmus für $n = 5$ gibt, der immer mit höchstens 7 Vergleichen auskommt.) Was sind die optimalen Suchlängen?*

Aufgabe 128 (⋆ ⋆)**:** *Eine außerordentlich knappe Implementierung des Quicksort–Algorithmus bietet die folgende Funktion:*

LISTING 1. Quicksort–Implementation.

```
def quicksort(objekte):
    """Diese Funktion gibt eine sehr konzise Implementation des
    Quicksort-Algorithmus: Das Argument objekte sollte eine Liste
    (oder ein anderer Iterator) von Objekten sein, die paarweise vergleichbar
    sind (für die also die Operatoren '>' und '<=' definiert sind),
    der Rückgabewert ist die sortierte Liste der Objekte."""
    # Illustration: Eine Liste oder ein Tupel ist
    # zwar an sich kein Wahrheitswert ("True" oder "False"),
    # aber: Eine Liste ist im Zusammenhang einer if-Bedingung
    # "falsy", wenn sie leer ist, sonst "truthy".
    if not objekte:
        # Für eine leere Liste ist natürlich nichts weiter zu tun
        return []
    # Illustration von _Unpacking_: Das erste Element von
    # objekte wird der variablen trenner zugewiesen, die restlichen
    # Elemente bilden die _neue_ Liste objekte:
    trenner, *objekte = objekte
    # Die restlichen Elemente werden in zwei Teile geteilt, je
    # nachdem ob sie kleinergleich oder größer
    kleinere = [o for o in objekte if o <= trenner]
    groessere = [o for o in objekte if o > trenner]
    # Rekursiver Aufruf der Funktion:
    return quicksort(kleinere) + [trenner] + quicksort(groessere)
```

Erklären Sie, warum die Funktion den Algorithmus an sich korrekt umsetzt, aber nicht sparsam mit dem Speicherplatz umgeht.

Implementieren Sie das in der Vorlesung beschriebene, Speicherplatz sparende Verfahren für einen `numpy`*–Vektor.*

Suchbäume in der Praxis

Aufgabe 129 (⋆ ⋆ ⋆)**:** *Implementieren Sie binäre Suchbäume als Python–Klasse und programmieren Sie* member functions *dieser Klasse, die den beiden "Durchlaufungsarten"*

- *depth–first*
- *und breadth–first*

entsprechen.

Hinweis: Verwenden Sie die Hinweise und Skizzen aus dem Jupyter–Notebook. Definieren Sie zunächst eine Klasse `BinaryTreeNode` *für die* Knoten *eines binären Baumes, die (zumindest) folgende* member variables *haben sollte:*

- `key`*: Bezeichnung ("Label") des Knoten (sollte* eindeutig *sein),*
- `data`*: Daten, die mit diesem Knoten verbunden sind (für uns nur eine Erinnerung, daß in "real interessierenden" Suchbäumen Daten gespeichert werden),*
- `predecessor`*: Vorgänger des Knoten (für die Wurzel:* `None`*),*
- `node_type`*: Wurzel oder linker/rechter Nachfolger eines Vorgängers?*
- `left`*: Wurzel des linken Teilbaumes (*`None`*, wenn nicht vorhanden),*
- `right`*: Wurzel des rechten Teilbaumes (*`None`*, wenn nicht vorhanden,*
- `append_left` *bzw.* `append_right`*: Hänge einen* `BinaryTreeNode` *als linken bzw. rechten Teilbaum an.*

Ein binärer Baum entspricht dann einer Klasse `BinaryTree`*, die als* member variable *(zumindest) eine* Instanz *von* `BinaryTreeNode` *(nämlich als Wurzel) enthält, und als* member functions *(zumindest) folgende Funktionen, die einen Knoten (identifiziert durch seinen* `key`*)* v *als Argument übernehmen:*

- `depth_first_search`*: Suche den Knoten* v *mit* depth–first,
- `insert_as_leaf`*: Füge den (neuen) Knoten* v *als (linken oder rechten) Nachfolger (Blatt) eines gegebenen Knoten* w *im Wurzelbaum ein (ebenfalls identifiziert durch seinen* `key`*),* falls w *nicht schon einen (linken oder rechten) Nachfolger hat*
- `delete_leaf`*: Entferne* v*,* falls v *ein Blatt ist.*

Darüber hinaus sollen auch folgende member function *implementiert werden:*

- `breadth_first_traversal`*: Durchlaufe alle Knoten mit* breadth–first,
- `depth_first_traversal`*: Durchlaufe alle Knoten mit* depth–first,

Bedenken Sie: Selbst–definierte Klassen sind mutable *Typen — wenn Sie einer Variablen eine solche Klasse* zuweisen*, entsteht ein neuer "Name" für dasselbe Objekt, keine neue Kopie, und das ist in diesem Zusammenhang "genau das Richtige"! (Wenn Sie sich mit der Sprache* `C` *auskennen: Eine solche Zuweisung funktioniert wie ein* Zeiger *auf das Objekt.)*

Aufgabe 130 ($\star$ $\star$ $\star$)**:** *Schreiben Sie eine Python–Funktion, die eine Wahrscheinlichkeitsverteilung (in Form eines* `numpy`*–Vektors) als Argument übernimmt und einen binären Baum als Rückgabewert liefert, der sich aus dem Huffman–Algorithmus (für* $q = 2$*) ergibt.*

Aufgabe 131 (⋆ ⋆ ⋆)**:** *Implementieren Sie AVL–trees als eine Python–Klasse* `AVLTree`*. Dazu können Sie die Klasse* `BinaryTreeNode` *aus Aufgabe 129 modifizieren: Zusätzlich zu den* member variables `left` *und* `right` *brauchen wir hier auch die* Längen *der an* `left` *und* `right` *hängenden Teilbäume, um die Balance–Bedingung für AVL–trees immer leicht überprüfen zu können.*

Die Suche nach einem Element `key` *in einem AVL–tree sollte natürlich mit einer* member function `find` *erfolgen, die das binary search Verfahren verwendet, und das Einfügen und Löschen eines Knotens sollte stets so erfolgen, daß wieder ein AVL–tree entsteht (eventuell durch rebalancing).*

Hinweis: Verwenden Sie die Hinweise und Skizzen aus dem begleitenden Jupyter–Notebook zu diesem Kapitel.

Algorithmische Geometrie

Konvexe Hülle einer Punktmenge in der Ebene

Aufgabe 132 (⋆)**:** *Beweisen Sie, daß eine Halbebene eine konvexe Menge ist.*

Hinweis: Verwenden Sie die "Normalvektor–Ungleichung" für Halbebenen und die Eigenschaften des inneren Produkts.

Aufgabe 133 (⋆ ⋆)**:** *Arbeiten Sie die im Beweis der Proposition zum inkrementellen Algorithmus zur Bestimmung der konvexen Hülle und seinem Laufzeitverhalten skizzierten Argumente im Detail aus. (Schauen Sie dazu insbesondere den Python–Code genauer an und stellen Sie sich die Sache geometrisch vor.)*

Zeigen Sie insbesondere, daß der inkrementelle Algorithmus asymptotisch *(also für große Mengen von Punkten) schneller läuft als der Algorithmus, der für alle zwei–elementigen Teilmengen von Punkten die entsprechenden Halbebenen betrachtet.*

Das Voronoi–Diagramm

Aufgabe 134 (⋆ ⋆)**:** *Die Kanten in einem zusammenhängenden Voronoi–Diagramm, die mit* nur einem *Knoten des Voronoi–Diagramms inzident sind, sind (wie alle Kanten des Voronoi–Diagramms) Teile (hier: Strahlen) der Streckensymmetralen von gewissen Punkten der Punktwolke. Zeigen Sie: Die Menge all dieser "gewissen Punkten der Punktwolke" stimmt mit der Menge der Eckpunkte der konvexen Hülle der Punktwolke überein.*

Weiterführendes Material

Diskrete Wahrscheinlichkeitsrechnung

Aufgabe 135 (⋆)**:** *Betrachte die Gleichverteilung auf $\mathfrak{S}_n$ und bestimme die Wahrscheinlichkeit, daß eine Permutation $\pi \in \mathfrak{S}_n$* genau *$k \leq n$ Fixpunkte hat.*

Aufgabe 136 (⋆ ⋆)**:** *Zeige, daß die Anzahlen*

- *der Dyck–Pfade der Länge $2n$*
- *und der Triangulierungen des $(n+2)$–Ecks*

dieselbe Rekursion erfüllen.

Aufgabe 137 (⋆ ⋆)**:** *Für zwei ganze Zahlen $h_1 > h_2$ betrachten wir die Familie aller Paare von Gitterpunktwegen (P_1, P_2), wo P_i in $(0, 2h_i)$ beginnt und in $(2n, 2h_i)$ endet, und wo P_1 und P_2 keinen Gitterpunkt gemeinsam haben (man nennt das* nichtüberschneidende Gitterpunktwege*). Zeigen Sie, daß die Anzahl dieser nichtüberschneidenden Gitterpunktwege durch die 2×2–Determinante*

$$\det \begin{vmatrix} \binom{2n}{n} & \binom{2n}{n-h_1+h_2} \\ \binom{2n}{n+h_1-h_2} & \binom{2n}{n} \end{vmatrix}$$

gegeben ist.

Hinweis: Versuchen Sie, die Idee des Spiegelungsprinzips geeignet zu adaptieren!)

Aufgabe 138 (⋆ ⋆ ⋆)**:** *Seien i, j und h ganze Zahlen, sodaß $0 \leq 2i, 2j \leq h$. Zeigen Sie: Die Anzahl aller Gitterpunktwege von $(0, 2i)$ nach $(2n, 2j)$, die* zwischen *den Geraden $y = 0$ und $y = h \geq 0$ liegen (d.h., diese Gitterpunktwege gehen nie unter die x–Achse und nie über die "Höhe" h), ist*

$$\sum_{k \in \mathbb{Z}} \left(\binom{2n}{n-i+j-k(h+2)} - \binom{2n}{n+i+j-k(h+2)+1} \right).$$

Hinweis: Kombinieren Sie das Spiegelungsprinzip mit dem Prinzip der Inklusion–Exklusion.