Algorithmen und Wahrscheinlichkeit

Frühjahrssemester 2019 – Lecture Notes

Sven Pfiffner

Davos|Zürich Mai 19

Inhalt

Graphentheorie *1*

Graphen-Zusammenhang **1**

Artikulationsknoten 1

Brücke 1

Finden von Artikulationsknoten 2

Anzahl disjunkte Knotenpfade **3**

Geschlossene Pfade im Graphen **3**

Eulertour 3

Hamiltonkreis 4

Exkurs: Hamilton-Kreis für Rotary Encoder 4

Satz von Dirac 6

Algorithmus: Hamilton-Kreis-Finder 7

Block **9**

Traveling Salesman **10**

Algorithmus 10

Matching **12**

Bestimmung eines maximal Matchings 13

Der Satz von Hall **14**

Korollar von Frobenius 15

Färbung **15**

Algorithmus 15

Wahrscheinlichkeit *16*

Exkurs: Visuelle Kryptografie **16**

Zufälle sind deterministisch **16**

Grundlagen der Stochastik **17**

Ergebnismenge 17

Ereignismenge 17

Wahrscheinlichkeitsraum 17

Siebformel (Prinzip der Inklusion/Exklusion) 18

Bedingte Wahrscheinlichkeit 19

Additionssatz 19

Multiplikationssatz 19

Totale Wahrscheinlichkeit 20

Unabhängigkeit von Ereignissen 20

Zufallsvariable **21**

Gedächtnislosigkeit 21

Dichtefunktion 22

Verteilungsfunktion 23

Bernoulli-Verteilung 23

Binomialverteilung 24

Geometrische Verteilung 25

Poisson Verteilung 26

Erwartungswert 27

Verteiltes Rechnen: Algorithmus **28**

Varianz **29**
Rechenregeln 30
Mehrere Zufallsvariablen **30**
Gemeinsame Dichte 30
Gemeinsame Verteilung 30
Unabhängigkeit von Zufallsvariablen **31**
Zusammengesetzte Zufallsvariablen 31
Abschätzen von Wahrscheinlichkeiten **32**
Ungleichung von Markov 32
Ungleichung von Chebychev 33
Chernoff Schranken 34
Randomisierte Algorithmen ***36***
Monte Carlo Algorithmen **36**
Fehlerreduktion 36
Las Vegas Algorithmen **37**
Fehlerreduktion 37
Beispielsalgorithmen **38**
Quickselect 38
π-Berechnung durch Target-Shooting 38
Kargers Algorithmus 39
Miller-Rabin Primzahltest 40
Exkurs – Hashing 41
Algorithmen – Highlights ***42***
Graphenalgorithmen **42**
Lange Pfade 42
Flüsse in Netzwerken 42
Minimal Schnitte in Graphen ..... **Fehler! Textmarke nicht definiert.**

Graphentheorie

Graphen-Zusammenhang

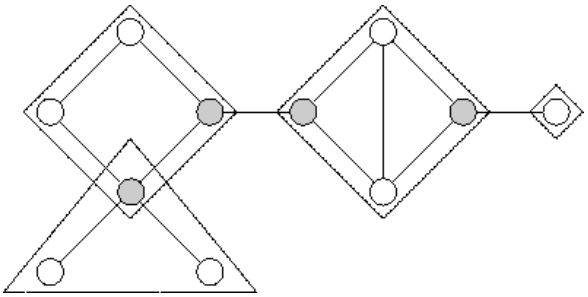
Gegeben ist ein zusammenhängender Graph. Die Frage nach dem Graphen-Zusammenhang befasst sich mit dem Aufwand der nötig ist, um den Zusammenhang im Graphen zu zerstören.

Ein Graph ist k-Zusammenhängend, wenn das Entfernen von k Knoten den Zusammenhang zerstört.

Merke: Jeder Graph ist HÖCHSTENS k-Zusammenhängend für $\min(\deg(v)) = k$.

Artikulationsknoten

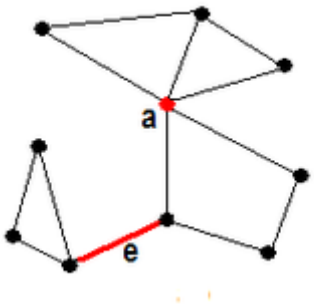
Unter einem Artikulationsknoten versteht man einen Knoten $v \in V$, so dass das Entfernen von v den Graphen teilt (den Zusammenhang zerstört).



Das Entfernen eines grauen Knoten teilt den Graphen in mehrere Teile

Brücke

Unter einer Brücke versteht man eine Kante $e \in E$, so dass das Entfernen von e den Graphen teilt (den Zusammenhang zerstört).



Das Entfernen von e teilt den Graphen in mehrere Teile

Ist $\{x, y\}$ eine Brücke, so gilt: $\deg(x) = 1$ oder x ist ein Artikulationsknoten.

Finden von Artikulationsknoten

Artikulationsknoten in einem Graphen G lassen sich durch eine modifizierte Tiefensuche finden. Dazu führen wir $dfs(G, s)$ von irgendeinem Startknoten s aus und speichern für jeden Knoten einen Wert $low[\,]$ und $dfs[\,]$.

- $dfs[v] :=$ Der Index von v in der Traversierungsreihenfolge.
- $low[v] :=$ Kleinste dfs-Nummer welche durch Nutzung von endlich vielen Vorwärtskanten und höchstens einer Rückwärtskante von v aus erreicht werden kann.

Es sei T der Tiefensuchbaum von G. v ist Artikulationsknoten genau dann, wenn $v = s \wedge \deg(v) \geq 2$ in T oder $v \neq s \wedge \exists_{w \in T} \mid w\ Nachbar\ von\ v\ und\ low[w] \geq low[v]$.

Anzahl disjunkte Knotenpfade

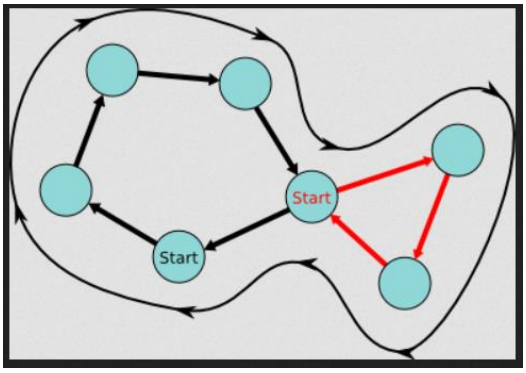
Man stellt sich die Frage, wie viele Kantenverschiedene Pfade in G von v nach w führen.

Satz von Menger: G ist k-Zusammenhängend ⟺ Es gibt k intern-disjunkte $u - v$ Pfade zwischen allen Kanten.

Geschlossene Pfade im Graphen

Eulertour

Eine Eulertour ist ein geschlossener Weg, welcher jede Kante in G genau einmal besucht.



Eine geschlossene Eulertour gibt es genau dann, wenn in einem vollständig verbunden Graphen jeder Knotengrad gerade ist.

Beweis:

Nach Definition muss eine Eulertour jeden Knoten, den sie betritt, auch wieder verlassen (mit Ausnahme des Startknoten, an welchem die Tour endet) und dabei alle Kanten des Graphen ablaufen. → Für jede Kante, welche zu einem Knoten führt, muss auch eine existieren, mit welcher der Knoten wieder verlassen werden kann. Das ist für einen Knoten v genau dann der Fall, wenn die Anzahl an zu v inzidenter Kanten gerade ist, also wenn gilt: $\deg(v)$ gerade

■

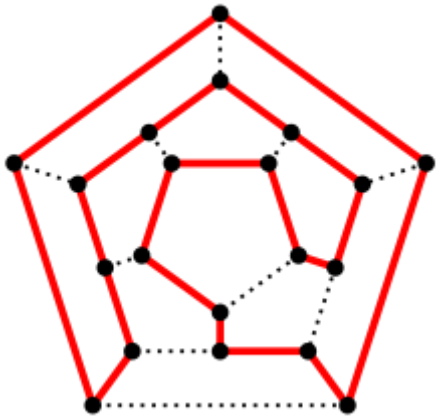
Eulertour Algorithmus

Es ist möglich Eulertouren in einem Graphen in Linearer Zeitkomplexität zu finden Dazu müssen wir aber erst prüfen, ob überhaupt eine Eulertour existiert (Zum Beispiel durch das Prüfen der Knotengrade [siehe: Eulertour]).

1. Wir wählen einen beliebigen Startknoten $s \in V$.
2. Wir traversieren den Graphen von s aus, indem wir zufällige ungenutzte Kanten ablaufen und diese als «genutzt» markieren. Wir machen das so lange bis wir wieder zu s zurückkehren.
3. Aufgrund der bewiesen geraden Knotengrade wissen wir, dass so auf jeden Fall ein Zyklus W entstehen muss
4. Gilt $\forall_{e \in E} \mid e \in W$ so haben wir eine Eulertour gefunden.
5. Anderenfalls beobachten wir, dass in G / W weiterhin alle Knotengrade gerade sind.
6. Wir suchen nun ein $v \in W$ welcher noch mindestens eine ungenutzte inzidente Kante besitzt und suchen von diesem v aus wieder einen Zyklus. Diesen hängen wir dann an W an.
7. Wir wiederholen bis $\forall_{e \in E} \mid e \in W$.
8. W ist nun eine Eulertour durch den Graphen.

Hamiltonkreis

Ein Hamiltonkreis ist ein geschlossener Pfad, welcher jeden Knoten in G genau einmal besucht.



Hamiltonkreis im Gittergraph

Generell gilt das Bestimmen der Existenz eines Hamiltonkreises in einem Graphen als sehr schweres Problem. Für Gittergraphen gibt es jedoch eine sehr einfache Überlegung, um zu zeigen, ob ein solcher Hamiltonkreis existiert.

Sei $m * n$ die Grösse eines Gittergraphen G. G besitzt einen Hamiltonkreis genau dann, wenn $m * n$ $gerade$.

Wir stellen uns einen Gittergraphen bipartit gefärbt vor. Für jeden Pfad durch diesen Graphen alternieren die Knotenfarben für benachbarte Knoten. Es existiere nun ein Hamiltonkreis durch besagten Graphen. Man erkennt, dass dieser zwangsweise auch bipartit sein muss falls er existiert, da es keine Kanten zwischen gleichgefärbten Knoten gibt. Wir wissen, dass ein Zyklus genau dann bipartit ist, wenn er eine gerade Anzahl Knoten enthält. Weil der Hamiltonkreis aber alle Knoten des Gittergraphen beinhaltet, existiert er demzufolge dann und nur dann, wenn der Gittergraph selbst eine gerade Knotenzahl aufweist.

■

Exkurs: Hamilton-Kreis für Rotary Encoder

Unter einem Rotary Encoder verstehen wir ein elektromechanisches Gerät, mit welchen sich die Winkelposition bzw.-Bewegung eines «Drehknopfes» in digitale Signale umwandeln lässt. Das funktioniert durch Drehung einer Lochscheibe, welche durch die Anzahl von «Lichtlöchern» ihre Winkelposition kodiert. Die geläufigste Kodierungsfunktion in Bits $\mathbb{N} \to \{0,1\}^k$ kann hier aber durch eine zu hohe Fehleranfälligkeit nicht genutzt werden (Man stelle sich vor eine Winkelposition wird durch 5 Bits kodiert. Ist die Drehscheibe nicht genügend genau wird aus $00101 = 5$ schnell $10101 = 21$ – also eine komplett andere Position). Die Idee: Man nutzt eine Kodierungsfunktion, bei der sich zwei aufeinanderfolgende Positionen in genau einer Bitstelle unterscheiden. Um besagte Kodierung zu konstruieren nutzen wir Hamilton-Kreise im Hyperwürfel.

Hamilton-Kreis im D-dimensionalen Hyperwürfel

Man stelle sich einen d-dimensionalen Hyperwürfel vor. Die Frage: besitzt ein solcher Würfel einen Hamilton-Kreis? Für 2 bzw. 3-dimensionale Würfel finden wir schnell Beispiele für Hamilton-Kreise:

Tatsächlich lässt sich durch Induktion zeigen, dass für jede beliebige Dimension $d \geq 2$ ein solcher Hamilton-Kreis existiert. Dazu definieren wir einen d-dimensionalen Hyperwürfel wie folgt:

Sei G ein vollständig zusammenhängender Graph mit $|V| = 2^d$ dessen Knoten die Menge $\{0,1\}^d$ repräsentieren. Ferner sei $H(v_1, v_2)$ die Hamming-Distanz zwischen diesen Knoten. G formt einen Hyperwürfel genau dann, wenn $\forall_{x,y \in V} | \exists_{\{x,y\} \in E} \Leftrightarrow H(x,y) = 1$. Heisst, genau dann, wenn sich die beiden zu x und y gehörigen Bitfolgen an genau einer Stelle unterscheiden.

Beweis (Induktion):

- **Base Case:**

Wir haben für $d = 1$ und $d = 2$ bereits gezeigt, dass ein Hamilton Kreis existiert. (Siehe Grafik)

- **Induction Hypothesis:**

Ein d-dimensionaler Hyperwürfel enthält einen Hamilton-Kreis.

- **Induction Step: ($\boldsymbol{d \to d+1}$)**

Zunächst betrachten wir alle Knoten im $(d+1)$-dimensionalen Hyperwürfel, für welche die letzte Stelle im Bitstring 0 ist. Nach Induktionshypothese können wir davon ausgehen, dass es durch diese Knoten einen Kreis gibt, da sie einen d-dimensionalen Hyperwürfel formen. Statt diesem Kreis betrachten wir jetzt nur einen Pfad durch alle diese Knoten, und zwar jenen, der im Knoten $00 \dots 0$ beginnt. Schreiben wir diesen Pfad jetzt zweimal hin, und zwar einmal vorwärts und einmal rückwärts, so können wir die erste Kopie hinten um eine Null ergänzen und die zweite Kopie durch eine Eins, und erhalten so einen Pfad von $0 \dots 00$ nach $0 \dots 01$ durch alle Knoten des $(d+1)$-dimensionalen Hyperwürfels. Und da die beiden Knoten $0 \dots 00$ und $0 \dots 01$ benachbart sind, ist dies dann ein Hamilton-Kreis für $(d+1)$

■

Satz von Dirac

Jeder Graph $G = (V, E)$ mit $|V| \geq 3$ und Minimalgrad $\delta(G) \geq \frac{|V|}{2}$ enthält einen Hamilton-Kreis

Widerspruchsbeweis:

Sei $G = (V, E)$ ein Graph mit $|V| \geq 3$ und Minimalgrad $\delta(G) \geq \frac{|V|}{2}$. Ferner sei G nicht hamiltonisch. Wir sehen, dass G in jedem Fall zusammenhängend ist. Warum? Betrachten wir zwei beliebige Knoten $x, y \in V \mid x \neq y$.

- Sei $\{x, y\} \in E$. Logischerweise existiert also ein Pfad zwischen x und y.
- Anderenfalls können wir aus der Definition von G schlussfolgern, dass $\deg(x), \deg(y) \geq \frac{|V|}{2}$. Damit kann $N(x) \cap N(y)$ nicht leer sein. Es muss also einen $(x - y)$-Pfad geben.

Wir betrachten nun einen beliebigen längsten Pfad $P = < v_1, \dots, v_k >$ in G und erkennen, dass $N(v_1) \cup N(v_k) \subset P$ (ansonsten könnten wir ja P um diese Knoten verlängern und er wäre kein längster Pfad mehr).

Die nächste Beobachtung ist zentral! Es gibt auf jeden Fall ein $2 \leq i \leq k$, so dass $v_i \in N(v_1)$ und $v_{i-1} \in N(v_k)$. Das gilt, weil v_1 ja zu mindestens $\frac{|V|}{2}$ vielen Knoten benachbart sein muss; und wäre v_k nicht mit v_{i-1} benachbart, dann wäre $|N(v_k)| < \frac{|V|}{2}$. Es gibt also auf jeden Fall ein solches v_i. Und $< v_1, v_i, v_{i+1}, \dots, v_k, v_{i-1}, v_{i-2}, \dots, v_2 >$ ist dann ein Kreis der Länge k.

Hier kommt nun der Widerspruch ins Spiel: Für $k = |V|$ wäre dieser Kreis ein Hamilton-Kreis, den es nach Annahme nicht gibt. Also ist $k < n$ was impliziert, dass Knoten in V nicht im Kreis liegen. Weil aber G zusammenhängend ist, müssten diese Knoten mit dem Kreis verbunden sein – es gäbe also einen Pfad der Länge $k + 1$ wenn wir diesen Knoten mit dem Kreis verbinden. Einen solchen Pfad kann es aber nicht geben, weil nach Definition der längste Pfad P die Länge k hat.

■

Algorithmus: Hamilton-Kreis-Finder

Das Finden eines Hamilton-Kreises gilt als algorithmisch sehr schweres Problem. Der naive Ansatz, alle möglichen Pfade durch G auf einen solchen Kreis zu prüfen ist zwar prinzipiell möglich und liefert auch die korrekte Antwort, läuft aber mit $O(n!)$ in einer praktisch unzumutbaren Laufzeit. Es ist jedoch möglich durch dynamische Programmierung den Rechenaufwand drastisch zu senken, was eine Lösung in $O(n^2 2^n)$ erlaubt.

Sei $G = (V, E)$ ein Graph, wobei wir annehmen wollen, das $V = [n]$ gilt. Tatsächlich genügt es zu testen, ob es ein $x \in N(1)$ gibt, für das es einen $(1 - x)$-Pfad P gibt, der alle Knoten aus V enthält; denn dann entspricht $< P, 1 >$ einem Hamilton-Kreis. Um zu testen, ob es ein solches $x \in N(1)$ gibt, definieren wir für alle Teilmengen $S \subseteq V$ mit $1 \in S$ und für alle $x \in S$ mit $x \neq 1$:

$$P_{s,x} = \begin{cases} 1, falls\ es\ in\ G\ einen\ (1-x)\ Pfad, der\ genau\ die\ Knoten\ aus\ S\ enthält\ gibt \\ 0, falls\ es\ keinen\ solchen\ Pfad\ gibt \end{cases}$$

Es gilt dann:

$$G \text{ enthält einen Hamilton-Kreis} \Leftrightarrow \exists_{x \in N(1)} | P_{[n],x} = 1.$$

und wir können $P_{[n],x}$ durch dynamische Programmierung aufbauen. Dazu müssen wir uns wie bei DP-Problemen üblich einen Base-Case und einen Induktionsschritt überlegen. Als erstes fällt auf, dass für die Mengen $S = \{1, x\}$ offenbar gilt: $P_{S,x} = 1$ genau dann, wenn $\{1, x\} \in E$. Wir nutzen diese Fälle als Base-Cases. Für den Schritt von Mengen der Grösse $s \geq 2$ auf Mengen der Grösse $s + 1$ überlegen wir uns folgendes: $P_{S,x}$ ist genau dann gleich Eins, wenn es ein $x' \in N(x) | x' \neq 1 \wedge P_{S\backslash[x],x'} = 1$, denn ein $(1 - x)$-Pfad mit Knoten aus S muss ja einen Nachbarn x' von x als vorletzten Knoten enthalten. Und der Pfad von 1 bis zu diesem Knoten x' ist dann ein $(1 - x')$-Pfad, der genau die Knoten aus $S\backslash[x]$ enthält. Damit gilt: $P_{S,x} = \max\{P_{S\backslash[x],x'} | x' \in S \cap N(x), x' \neq 1\}$.

Laufzeit

Bei der Initialisierung setzen wir für alle Knoten in der Adjazenzliste von 1: $P_{S,x} = 1$; Wir benötigen also lineare Zeit, um alle Base-Cases zu berechnen. Auch die Ausgabe erfolgt in linearer Zeit, denn alles was wir tun müssen ist über alle Knoten zu iterieren und zu prüfen, ob für einen davon gilt, dass $P_{[n],x} = 1$. Damit wird die Laufzeit klar durch die Rekursive Berechnung der DP-Tabelle dominiert. Werfen wir nun einen Blick auf den Pseudocode dieser Berechnung:

---

for all $s \geq 3$ do { //[1] Go through all subset sizes from 3 to $|V|$

for all $S \subseteq [n]$ mit $1 \in S$ und $|S| = s$ do { //[2] Go through all subsets of this size with 1

for all $x \in S, x \neq 1$ do { //[3] Go through all nodes in this subset (excluding 1)

$P_{S,x} = \max\{P_{S\setminus[x],x'} \mid x' \in S \cap N(x), x' \neq 1\}$ //[4] Calc entry

}

}

}

---

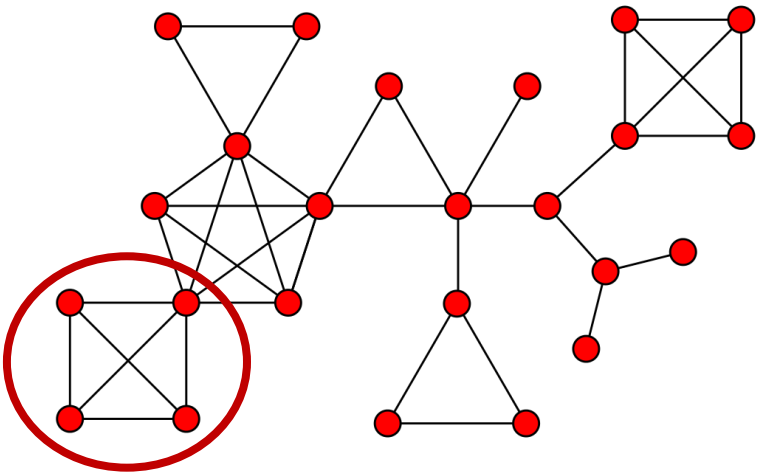
1. Wird genau $n - 2$ mal aufgerufen
2. Wird so oft aufgerufen, wie es verschiedene Teilmengen der gegebenen Kardinalität gibt, welche die 1 enthalten (Zur Erinnerung: Beim Ziehen von k Elementen aus einer n-Elemente-Menge gibt es stets $\binom{n}{k}$ verschiedene Möglichkeiten).
3. Wird linear aufgerufen. Und zwar genau $|S| - 1$ mal
4. Verläuft offensichtlich in $O(n)$ – wir müssen lediglich alle Werte durchgehen und den Max-Wert finden

Damit erhalten wir eine Laufzeit von:

$$\sum_{s=3}^{n}\left(\sum_{S\subseteq[n],1\in S,|S|=s}\left(\sum_{x\in S,x\neq 1}(O(n))\right)\right) = \sum_{s=3}^{n}\binom{n-1}{s-1} * (s-1) * O(n) = O(n^2 2^n)$$

Block

Ein Block bezeichnet eine maximale Menge von Knoten, so dass je zwei dieser Knoten auf einem Kreis liegen.



Beispiel eines Graphen mit mehreren Blöcken.

Traveling Salesman

Das Traveling Salesman Problem (folgend TSP) ist ein graphentheoretisches Problem, welches häufig bei Logistik- und Transportaufgaben auftritt, Beim TSP soll ein reisender Händler eine bestimmte Anzahl an Städten besuchen und zum Schluss wieder in der Anfangsstadt ankommen. Für die Rundreise soll dabei der kürzeste Weg gewählt werden.

Zentral ist hier vor allem die Überlegung, dass es sich beim TSP um einen Spezialfall eines Hamilton-Kreis-Problems handelt. Sei G ein Graph, bei dem jeder Knoten $v \in V$ einer Hauptstadt entspricht und alle Knoten mit allen verbunden sind. Schnell fallen zwei Dinge auf:

- In jedem Fall muss der Graph mindestens einen Hamilton-Kreis aufweisen, weil durch die Vollständigkeit von G jeder Hamiltonpfad $p =< v_s, v_a, \dots, v_z \in N(v_s) >$ im Graphen zu einem Hamilton-Kreis werden kann (Es gibt in jedem Fall immer eine Kante $\{v_z, v_s\}$).
- Sei H_G die Menge aller Hamilton-Kreise in G. Die Lösung des TSP in G ist in jedem Fall Element von eben dieser Menge. Es reicht also theoretisch aus, alle Hamilton-Kreise in G zu verifizieren, um die Lösung des TSP zu finden.

Wir suchen einen Hamilton-Kreis mit $\min_{Hamiltonkreis} \sum_{l \in E(H)} l(e)$. In einem vollständigen Graphen mit Kantengewichten. Also denjenigen aller Hamilton-Kreise im Graphen G, dessen Kantengewichtssumme minimal ist.

Algorithmus

Für das Metrische TSP, also ein TSP bei dem die Kantengewichte des Graphen die Dreiecksungleichung $l(\{x,z\}) \leq l(\{x,y\}) + l(\{y,z\})$ erfüllen, existiert ein 2-Approximations-Algorithmus, welcher das Prinzip der minimalen Spannbäume nutzt.

1. Wir finden einen minimalen Spannbaum in G mit dem Algorithmus von Kruskal oder Prim
2. Wir verdoppeln in diesem Spannbaum alle Kanten.
3. Jetzt suchen wir in dem Spannbaum eine Eulertour; beispielsweise mit dem Eulertour Algorithmus.
4. Wir stellen sicher, dass in der Eulertour keine Knoten mehr als einmal vorkommen.
5. Die daraus resultierende Knotensequenz entspricht einer 2-Approximation für das TSP Problem

Sei C_o ein Hamilton-Kreis im Graphen G, welcher der Optimal-Lösung des TSP entspricht, so findet der 2-Approximations-Algorithmus immer einen Hamilton-Kreis C_1 für den gilt, dass $l(C_1) \leq 2 * l(C_0)$

Beweis:

Nach Annahme erfüllt G die Dreiecksungleichung. Daher wird C_1 durch Auslassen von Knoten sicher nicht grösser. Wir erhalten also immer einen Hamilton-Kreis mit Länge $\leq 2 * l\big(mst(G)\big)$.

Aus jedem Hamilton-Kreis wird durch Weglassen einer beliebigen Kante ein Spannbaum. Daraus können wir schliessen, dass $l\big(mst(G)\big) \leq l(C_0)$. Daher gilt für die Länge $l(C_1)$ der von uns gefundenen Lösung C_1, dass $l(C_1) \leq 2 * l(C_0)$.

■

Laufzeit

Um die Laufzeit zu bestimmen, werfen wir einen Blick auf die vom Algorithmus ausgeführten Schritte:

1. Kruskal/Prim finden einen MST in $O(n^2)$
2. Das Verdoppeln der Kanten geschieht je nach Graphen-Implementation in höchstens $O(n^2)$
3. Wir finden mit dem Eulertour Algorithmus eine Eulertour in $O(n)$
4. Wir löschen ungewollte Knoten in der Sequenz in $O(n)$

Damit beträgt die Laufzeit des 2-Approximations-Algorithmus des TSP $O(n^2)$

Verbesserung der 2-Approximation

Wir können den Genauigkeitskoeffizienten des Approximationsalgorithmus von 2 auf $\frac{3}{2}$ senken, indem wir Eigenschaften von Matchings in Graphen ausnutzen.

Für das Metrische Travelling Salesman Problem gibt es einen $\frac{3}{2}$-Approximationsalgorithmus mit Laufzeit $O(n^3)$.

Beweis:

Beim 2-Approximationsalgorithms haben wir einfach einen MST berechnet, jede Kante des Spannbaums verdoppelt, um einen eulerschen Graphen zu erhalten, und dann durch löschen von Mehrfachknoten aus der Eulertour einen Hamilton-Kreis erstellt. Im Grundsatz gehen wir nun genauso vor, überlegen uns aber, ob es nicht möglich ist, bessere Kanten für das «Eulersch machen» zu finden.

Sei S die Menge derjenigen Knoten, die in T einen ungeraden Grad haben. Da $|S|$ gerade ist (in jedem Graphen ist die Anzahl an Knoten mit ungeradem Grad gerade [siehe Script S.7]), gibt es für den vollständigen Graphen K_n über S ein perfektes Matching. Wir finden nun unter allen solchen Matchings eines mit minimalem Gewicht, was nach Satz 1.45 im Script in $O(n^3)$ möglich ist, und bezeichnen es mit M. Betrachten wir jetzt den Graphen $T \cup M$, welcher auf jeden Fall geraden Knotengrad hat (Wir haben jeden ungeraden Knotengrad durch das Matching um 1 erhöht), können wir ganz analog zum 2-Approximationsalgorithmus eine Eulertour und damit auch einen Hamilton-Kreis finden.

■

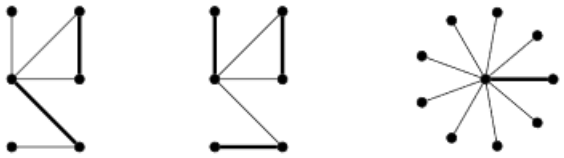
Matching

Wir betrachten folgendes Problem: Wir sind eine Partnerbörse und wollen eine Menge von Männern einer Menge von Frauen zuordnen. Weiter wissen wir für je zwei Personen, ob ein beidseitiges Interesse für eine Beziehung besteht oder nicht. Gibt es nun eine Möglichkeit, Paare zu bilden, so dass jede Person am Ende einer anderen zugeordnet ist?

Wir können das Problem Graphentheoretisch wie folgt formulieren: Wir symbolisieren jede Person durch einen Knoten und verbinden einen Mann und eine Frau genau dann mit einer Kante, wenn die Bedingungen für eine Beziehung gegeben sind. Gesucht ist dann eine Auswahl der Kanten, die jedem Mann genau eine Frau zuordnet und umgekehrt. Eine solche Teilmenge der Kanten nennt man ein Matching des Graphen.

Wir nennen eine Kantenmenge $M \subseteq E$ Matching in einem Graphen $G = (V, E)$, falls kein Knoten des Graphen zu mehr als einer Kante aus M inzident ist; also wenn gilt $\forall_{e,f \in M} | e \neq f \Rightarrow e \cap f = \emptyset$.

Ein Matching M heisst perfektes Matching, wenn jeder Knoten durch genau eine Kante aus M überdeckt wird; wenn also gilt: $|M| = \frac{|V|}{2}$.



Beispiele für Matchings in Graphen

Zurück zu unserer Partnerbörse. Natürlich wollen wir so viele Kunden wie möglich zufriedenstellen – d.h. sie einem Matching zuteilen. Wir interessieren uns also hauptsächlich dafür, für einen Graphen ein möglichst grosses Matching zu finden. Nun gibt generell zwei verschiedene Möglichkeiten, «möglichst gross» zu interpretieren.

- M heisst **inklusionsmaximal (eng. maximal)**, falls gilt $M \cup \{e\}$ ist kein Matching für alle Kanten $e \in E \backslash M$.
- M heisst **kardinalitätsmaximal (eng. maximum)**, falls gilt $|M| \geq |M'|$ für alle Matchings M' in G.

Sei M_{Greedy} ein inklusionsmaximales und M_{max} ein kardinalitätsmaximales Matching von G. Es gilt mit Sicherheit immer, dass: $\left|M_{Greedy}\right| \geq \frac{1}{2}|M_{max}|$.

Beweis:

Da M_{Greedy} inklusionsmaximal ist, muss jede Kante aus M_{max} mindestens eine Kante aus M_{Greedy} berühren. Weil M_{max} ein Matching ist, kann andererseits jeder Knoten aus M_{Greedy} (von denen es $2\left|M_{Greedy}\right|$ viele gibt), nur eine Kante von M_{max} berühren. Aus beiden Fakten zusammen folgt daher $|M_{max}| \leq 2\left|M_{Greedy}\right|$ und damit auch $\left|M_{Greedy}\right| \geq \frac{1}{2}|M_{max}|$

■

Bestimmung eines maximal Matchings

Ein inklusionsmaximales Matching kann sehr einfach mit einem Greedy Algorithmus bestimmt werden:

```
Edges matching = new Edges    //Wir erstellen eine leere Liste von Kanten

while (!E.empty) do           //Während noch Kanten in G
        wähle ein beliebiges e ∈ E
        matching.add(e)
        Lösche e und alle inzidenten Kanten in G
```

Wir schauen jede Kante des Graphen genau einmal an, was uns eine Laufzeit von $O(|E|)$ garantiert. Deshalb ist der Algorithmus, wie für einen Greedy Algorithmus üblich, sehr effizient.

Der Satz von Hall

Ein Graph $G = (V, E)$ heisst bipartit, wenn man die Knotenmenge V so in zwei Mengen A und B partitionieren kann, dass alle Kanten in E je einen Knoten aus A und einen Knoten aus B enthalten. Bipartite Graphen schreiben wir dann entsprechend als $G = (A \uplus B, E)$.

Wir definieren die Nachbarschaft einer Knotenmenge $X \subseteq V$ wie folgt:

$$N(X) \coloneqq \bigcup_{v \in X} N(v)$$

Nach dem Satz von Hall, auch der Heiratssatz genannt, gibt es für einen bipartiten Graphen $G = (A \uplus B, E)$ genau dann ein Matching M der Kardinalität $|M| = |A|$, wenn gilt: $|N(X)| \geq |X|$ für alle $X \subseteq A$.

Beweis:

- $\Rightarrow$ Sei M ein Matching der Kardinalität $|M| = |A|$. In dem Teilgraphen $H = (A \uplus B, M)$ hat jede Teilmenge $X \subseteq A$ nach Definition eines Matchings genau $|X|$ Nachbaren. Wegen $M \subseteq E$ gilt daher auch $|N(X)| \geq |X|$ für alle $X \subseteq A$.
- $\Leftarrow$ Wir induzieren über die Kardinalität $a = |A|$.
  - **Base Case:** $(a = 1)$ Befindet sich nur ein Knoten in A so impliziert die Bedingung $|N(X)| \geq |X|$, dass dieser Knoten zu mindestens einer Kante inzident ist. Jede solche Kante ist dann ein Matching, welches die Bedingung $|M| = |A|$ erfüllt.
  - **Induction Hypothesis:** Jeder Graph $G = (A \uplus B, E)$ mit $|N(X)| \geq |X|$ für alle $X \subseteq A$ besitzt ein Matching M der Kardinalität $|M| = |A|$.
  - **Induction Step:** $(a \to a + 1)$ Wir führen eine Fallunterscheidung ein!
    - **Fall 1:** Sei $|N(X)| > |X|$ für alle $X \subset A$, so wählen wir eine beliebige Kante $e = \{x, y\}$ des Graphen, fügen e zum Matching hinzu und betrachten den Graphen G', den wir erhalten, wenn wir die Knoten x und y (sowie alle inzidenten Kanten) löschen. Es fällt auf, dass die Bedingung $|N(X)| \geq |X|$ für G' gilt (wir haben ja nur einen Knoten aus B gelöscht, also höchstens $|N(X)|$ um 1 verkleinert). Aus der Induktionsannahme folgt daher, dass wir die Kante e durch ein Matching M' so ergänzen können, dass dann alle Knoten in A überdeckt werden.
    - **Fall 2:** Gelte $|N(X)| > |X|$ nicht, so muss es mindestens eine Menge $X_0 \subset A$ geben, für die $|N(X_0)| = |X_0|$ gilt. Wir wenden nun die Induktionshypothese auf die beiden induzierten knotendisjunkten Graphen $G' = G[X_0 \uplus N(C_0)]$ und $G'' = G[A \backslash X_0 \uplus B \backslash N(X_0)]$ an. $|N(X)| \geq |X|$ ist offensichtlich für G' erfüllt, gilt aber auch für den Graphen G'', denn betrachten wir eine beliebige Menge $X \subseteq A \backslash X_0$ für den Graphen G, so wissen wir $|X| + |X_0| = |X \cup X_0| \leq |N(X \cup X_0)| = |N(X_0)| + |N(X) \backslash N(X_0)|$ und wegen $|N(X_0)| = |X_0|$ folgt daraus $|X| \leq |N(X) \backslash N(X_0)| = \left|N(X) \cap \left(B \backslash N(X_0)\right)\right|$. Das heisst, die Nachbarschaft von X in dem Graphen G'' besteht aus mindestens $|X|$ Knoten.

■

Korollar von Frobenius

Für alle k gilt: jeder k-reguläre, bipartite Graph besitzt ein perfektes Matching

Tatsächlich lässt sich aus diesem Korollar schliessen, dass es für einen k-regulärem bipartiten Graphen stets k verschiedene perfekte Matchings gibt.

Beweis:

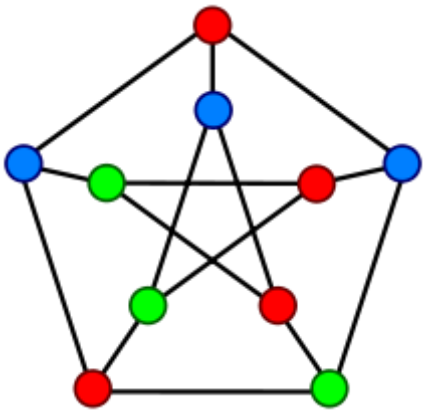
Erstellen wir ein perfektes Matching auf dem k-regulären, bipartiten Graphen G. Entfernen wir alle Kanten des Matchings aus G so erhalten wir einen $k-1$-regulären, bipartiten Graphen G', welcher nach Frobenius wieder ein perfektes Matching enthält.

■

Färbung

Eine Färbung eines Graphen $G = (V, E)$ mit k-Farben ist eine Abbildung $c: V \to [k]$, so dass gilt:

$$c(u) \neq c(v) \text{ für alle Kanten } \{u, v\} \in E.$$



Die chromatische Zahl $X(G)$ ist die minimale Anzahl Farben, die für eine Knotenfärbung von G benötigt wird.

Das Problem

Gegeben ein Graph $G = (V, E)$, gilt $X(G) \leq 3$?

Ist **NP-Vollständig**

Algorithmus

Gegeben sei ein Graph G. Wir wollen G mit möglichst wenigen Farben färben.

- Eine Möglichkeit ist der **Greedy-Algorithmus** (Siehe Vorlesung A&D-HS 18), welcher eine Laufzeit von $O(|V| + |W|)$ aufweist und garantiert eine $(\max(\deg(v)) + 1)$-partite Färbung zurückgibt.
- … (S.72)
- Es gebe einen Artikulationsknoten V in G …(S.72)
    - Farbtausch!!

Wahrscheinlichkeit

Wieso brauchen wir Wahrscheinlichkeit in der Informatik?

⇒ Manchmal sind nicht-deterministische Algorithmen schneller als deterministische. Ein Beispiel dazu wäre der Quicksort-Algorithmus der trotz einer worst-case Laufzeit von $O(n^2)$ bei zufälliger Wahl des Pivot-Elements beweisbar im Schnitt eine Laufzeit von $O(n * \log(n))$ aufweist.

Exkurs: Visuelle Kryptografie

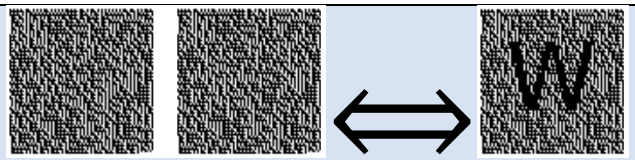
Zufälle finden unter anderem in der visuellen Kryptografie Anwendung. Sagen wir, wir haben ein Bild, das nur aus schwarzen und weissen Pixeln besteht. Wir stellen uns dieses als eine $n * m$ Matrix von Nullen und Einsen vor, wobei eine 0 ein schwarzer Pixel darstellt, und 1 ein weisser. Wollen wir dieses Bild nun verschlüsseln, so greifen wir auf eine Idee von Moni Naor und Adi Shamir zurück.

Wir zerlegen das Bild B in zwei Bilder B_1 und B_2, die jeweils die doppelte Grösse von B haben (beide haben also Dimension $2(n * m) = 2n * 2m$). Wir ersetzen nun jeden Pixel von B durch $2 * 2$ Blöcke nach Schema S wählen dabei für jeden Pixel zufällig, ob wir dem linken- oder rechten Verfahren folgen.

■ ↦ (▚, ▚) oder (▚, ▚)

□ ↦ (▚, ▚) oder (▚, ▚)

Verschlüsselungsschema S



Wir können B nur rekonstruieren, wenn wir sowohl B_1 als auch B_2 kennen.

Zufälle sind deterministisch

Alle Vorgänge in Computerhardware laufen streng deterministisch; also ist es ohne Weiteres nicht möglich, zufällige Werte zu generieren. Wir müssen uns sogenannten Zufallsgeneratoren bedienen, welche meistens unvorhersehbare 'Seeds' (wie zum beispiel System.nanoTime()) nutzen, um zufallszahlen trotz deterministischer Funktionen zu simulieren.

$$f: \{0,1\}^m \longrightarrow \{0,1\}^m * \{0,1\}^m$$
$$S_{i-1} \mapsto (S_i, a_i)$$

Beispiel für eine deterministische «Zufallsfunktion»

Grundlagen der Stochastik

Ergebnismenge

Wir nennen die Menge der möglichen Elementarereignisse die Ergebnismenge und schreiben:

$$\Omega = \{\omega\colon \omega \text{ ist ein mögliches Ergebnis des Experiments}\}$$

Ereignismenge

Die Ereignismenge ist die Menge aller möglichen Teilmengen von Ω; beschreibt also alle Ereignisse (kombiniert sowie elementar), welche auftreten können. Damit ist die Ereignismenge $P(\Omega)$ definiert als die Potenzmenge von Ω und wir wissen von der Mengentheorie, dass $|P(\Omega)| = 2^{|\Omega|}$.

Wahrscheinlichkeitsraum

Sei Ω eine beliebige Ergebnismenge und P das sogenannte Wahrscheinlichkeitsmass, welches jedem Element der Ergebnismenge eine Wahrscheinlichkeit zuordnet.

$$P\colon \text{Powerset}(\Omega) \longrightarrow \mathbb{R}$$

Wir nennen den Raum (Ω, P) den Wahrscheinlichkeitsraum über die Ergebnismenge Ω.

Laplace-Raum

Wir verstehen unter einem Laplace-Raum einen Wahrscheinlichkeitsraum, bei dem alle Elementarereignisse die gleiche Wahrscheinlichkeit haben. Das heisst:

$$P(\{\omega\}) = \frac{1}{|\Omega|} \; \forall_{\omega \in \Omega}$$

Siebformel (Prinzip der Inklusion/Exklusion)

Für Ereignisse $A_1, \dots, A_n (n \geq 2)$ gilt:

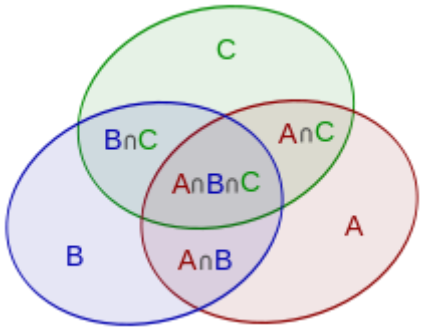
$$\begin{aligned}\Pr\left[\bigcup_{i=i}^{n} A_i\right] &= \sum_{i=1}^{n} \Pr[A_i] - \sum_{1 \leq i_i < i_2 \leq n} \Pr\left[A_{i_1} \cap A_{i_2}\right] \pm \cdots \\ &+ (-1)^{l-1} * \sum_{1 \leq i_1 < \cdots < i_l \leq n} \Pr\left[A_{i_1} \cap \dots \cap A_{i_l}\right] \pm \cdots \\ &+ (-1)^{n-1} * \Pr[A_1 \cap \dots \cap A_n] \\ &\leq \sum_{i=1}^{n} \Pr[A_i]\end{aligned}$$

Insbesondere gilt dann für zwei Ereignisse A und B:

$$\Pr[A \cup B] = \Pr[A] + \Pr[B] - \Pr[A \cap B]$$

Und für drei Ereignisse A_1, A_2, A_3:

$$\begin{aligned}\Pr[A_1 \cup A_2 \cup A_3] = &\Pr[A_1] + \Pr[A_2] + \Pr[A_3] \\ &- \Pr[A_1 \cap A_2] - \Pr[A_1 \cap A_3] - \Pr[A_2 \cap A_3] \\ &+ \Pr[A_1 \cap A_2 \cap A_3]\end{aligned}$$



Beispiel zum Prinzip der Inklusion/Exklusion

Bedingte Wahrscheinlichkeit

Mit $A|B$ bezeichnen wir das Ereignis, dass A eintritt, wenn wir bereits wissen, dass das Ereignis B auf jeden Fall eintritt. Das bedeutet, dass folgende Eigenschaften auf jeden Fall erfüllt sind.

1. $\Pr[B|B] = 1$, denn wissen wir, dass B eintritt, so ist die Wahrscheinlichkeit eines Eintretens von B bei 100%.
2. $\Pr[B|\bar{B}] = 0$, denn wissen wir, dass $\bar{B}$ eintritt, so ist die Wahrscheinlichkeit eines Eintretens von B bei 0%.
3. $\Pr[A|\Omega] = \Pr[A]$, denn wissen wir, dass irgendein Ereignis aus Ω eintritt, so liefert uns das noch keinen Aufschluss auf die Wahrscheinlichkeit eines spezifischen Ereignisses.
4. Wenn wir wissen, dass B eingetreten ist, dann kann ein Ereignis A nur dann eintreten, wenn zugleich auch $A \cap B$ eintritt. Die Wahrscheinlichkeit $\Pr[A|B]$ ist demnach proportional zu $\Pr[A \cap B]$.

A und B seien Ereignisse mit $\Pr[B] > 0$. Die bedingte Wahrscheinlichkeit $\Pr[A|B]$ von A gegeben B ist definiert durch:

$$\Pr[A|B] := \frac{\Pr[A \cap B]}{\Pr[B]}$$

Additionssatz

Wenn die Ereignisse $A_1, \dots A_n$ paarweise disjunkt sind ($\forall_{i \neq j}: A_i \cap A_j = \emptyset$), so gilt:

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] = \sum_{i=i}^{n} \Pr[A_i]$$

Multiplikationssatz

Seien die Ereignisse $A_1, \dots, A_n$ gegeben. Falls $\Pr[A_1 \cap \dots \cap A_n] > 0$ ist (das heisst die Wahrscheinlichkeit, dass alle gegebenen Ereignisse gleichzeitig eintreffen können besteht), so gilt:

$$\Pr[A_1 \cap \dots \cap A_n] = \Pr[A_1] * \Pr[A_2|A_1] * \Pr[A_3|A_1 \cap A_2] \dots \Pr[A_n|A_1 \cap \dots \cap A_{n-1}]$$

Beweis:

Aus der Definition der bedingten Wahrscheinlichkeit folgt:

$$\begin{aligned}
\Pr[A_1 \cap \dots \cap A_n] &= \Pr[A_1] * \Pr[A_2|A_1] * \Pr[A_3|A_1 \cap A_2] \dots \Pr[A_n|A_1 \cap \dots \cap A_{n-1}] \\
&= \frac{\Pr[A_1]}{1} * \frac{\Pr[A_2 \cap A_1]}{\Pr[A_1]} * \frac{\Pr[A_3 \cap A_1 \cap A_2]}{\Pr[A_1 \cap A_2]} * \dots * \frac{\Pr[A_n \cap A_1 \cap \dots \cap A_{n-1}]}{\Pr[A_1 \cap \dots \cap A_{n-1}]} \\
&= \frac{\Pr[A_1] * \Pr[A_2 \cap A_1] * \Pr[A_3 \cap A_1 \cap A_2] * \dots * \Pr[A_n \cap A_1 \cap \dots \cap A_{n-1}]}{\Pr[A_1] * \Pr[A_1 \cap A_2] * \dots * \Pr[A_1 \cap \dots \cap A_{n-1}]} \\
&= \Pr[A_1 \cap \dots \cap A_n]
\end{aligned}$$

■

Totale Wahrscheinlichkeit

Seien $A_1, \dots A_n$ paarweise disjunkte Ereignisse und $E \subseteq A_1 \cup \dots \cup A_n$, so folgt:

$$\Pr[B] = \sum_{i=1}^{n} \Pr[B|A_i] * \Pr[A_i]$$

Beweis:

Halten wir fest, dass $B = (B \cap A_1) \cup \dots \cup (B \cap A_n)$. Da für beliebige i, j mit $i \neq j$ gilt, dass $A_i \cap A_j = \emptyset$ ist (wir haben ja alle A als paarweise disjunkt definiert), sind auch die Ereignisse $B \cap A_i$ und $B \cap A_j$ disjunkt. Wegen des Multiplikationsssatzes gilt $\Pr[B \cap A_i] = \Pr[B|A_i] * \Pr[A_i]$. Unter Anwendung des Additionssatzes folgt dann:

$$\begin{aligned} \Pr[B] &= PR\,[B \cap A_1] + \dots + \Pr[B \cap A_n] \\ &= \Pr[B|A_1] * \Pr[A_1] + \dots + \Pr[B|A_n] * \Pr[A_n] \end{aligned}$$

■

Da der Additionssatz auch für unendlich viele Ereignisse $A_1, A_2, \dots$ gilt, kann dieser Beweis direkt auf den unendlichen Fall übertragen werden. Damit gilt auch:

Seien $A_1, A_2, \dots$ paarweise disjunkte Ereignisse und $B \subseteq \bigcup_{i=1}^{\infty} A_i$, so gilt:

$$\Pr[B] = \sum_{i=1}^{\infty} \Pr[B|A_i] * \Pr[A_i]$$

Unabhängigkeit von Ereignissen

Bei bedingten Wahrscheinlichkeiten $\Pr[A|B]$ kann unter Umständen auftreten, dass B gar keinen Einfluss auf $\Pr[A]$ hat. In diesem Fall gilt dann also $\Pr[A|B] = \Pr[A]$ und wir nennen A und B voneinander unabhängig.

Seien A und B zwei beliebige Ereignisse einer Ereignissmenge. A und B heissen unabhängig, wenn gilt:

$$\Pr[A \cap B] = \Pr[A] * \Pr[B]$$
$$\Rightarrow \Pr[A|B] = \Pr[A] \wedge \Pr[B|A] = \Pr[B]$$

Wir können den Begriff der Unabhängigkeit von 2 auf endlich/unendlich viele Ereignisse erweitern.

Seien $A_1, \dots A_n$ beliebige Ereignisse einer Ereignissmenge. $A_1, \dots, A_n$ heissen unabhängig, wenn für alle Teilmengen $I \subseteq \{1, \dots, n\}$ mit $I = \{i_1, \dots, i_k\}$ gilt:

$$\Pr[A_{i_1} \cap \dots \cap A_{i_k}] = \Pr[A_{i_1}] \dots \Pr[A_{i_k}]$$

Weiter gilt eine unendliche Familie von Ereignissen $A_i : i \in \mathbb{N}$ als unabhängig, wenn die obige Bedingung für jede Endliche Teilmenge $I \subseteq \mathbb{N}$ erfüllt ist. Also wenn gilt:

$$\forall_{I \in Powerset(\mathbb{N})} : \Pr[A_{i_1} \cap \dots \cap A_{i_k}] = \Pr[A_{i_1}] \dots \Pr[A_{i_k}]$$

Gut zu wissen: Seien A, B und C unabhängige Ereignisse. Dann sind auch $A \cap B$ und C bzw. $A \cup B$ und C voneinander unabhängig.

Zufallsvariable

Als Zufallsvariable beschreiben wir eine Abbildung $X: \Omega \to \mathbb{R}$, wobei Ω die Ergebnismenge eines Wahrscheinlichkeitsraumes ist.

Bei diskreten Wahrscheinlichkeitsräumen ist der Wertebereich einer Zufallsvariablen immer endlich oder abzählbar unendlich (hängt von $|\Omega|$ ab). Wir definieren diesen Wertebereich wie folgt:

$$W_X \coloneqq X(\Omega) = \{x \in \mathbb{R} | \exists_{\omega \in \Omega} \text{ mit } X(\omega) = x\}$$

Beispiel:

Wir werfen eine Laplace-Münze drei Mal. Als Ergebnismenge erhalten wir $\Omega \coloneqq \{K, Z\}^3$. Die Zufallsvariable X bezeichnet nun die Gesamtanzahl der Würfe mit Ergebnis «Kopf». Beispielsweise gilt also:

- $X(KZK) = 2$
- $X(KKK) = 3$
- $X(ZZK) = 1$
- …

X hat hier den Wertebereich $W_X = \{0,1,2,3\}$. Denn bei jedem Wurf wird einer dieser Werte erreicht.

Gedächtnislosigkeit

Eine Zufallsvariable heisst gedächtnislos, falls gilt:

$$\Pr[X \geq s + t \mid X > s] = \Pr[X \geq t]$$

Ist X geometrisch verteilt, so ist X gedächtnislos.

Beweis:

$$\Pr[X \geq s + t] = \frac{\Pr[X \geq s + t] \cap \Pr[X \geq s]}{\Pr[X > s]} = \frac{(1-p)^{s+t-1}}{(1-p)^s} = (1-p)^{t-1} = \Pr[x \geq t]$$

■

Dichtefunktion

Dichtefunktionen der Zufallsvariable x werden mit $f(x)$ angegeben. Die Dichtefunktion hat vor allem die Aufgabe, einen visuellen Eindruck der Verteilung zu vermitteln: Wie der Name bereits andeutet, zeigt diese Funktion, in welchen Teilen sich die Werte der Zufallsvariablen am **dichtesten** scharen.

! Das Integral der Dichtefunktion ist die Verteilungsfunktion und die Fläche unter der Dichtefunktion hat immer den Inhalt 1!

Die Wahrscheinlichkeitsfunktion $\Pr[\cdot]$ induziert die Dichte von X gemäss:

$$f_X: \mathbb{R} \to [0,1]$$
$$x \mapsto \Pr[X = x]$$

Beispiel:

Die Kugel einer Kugelbahn rollt auf einer waagrechten Schiene aus. Die Entfernung vom Beginn der waagrechten Rollschiene bis zu dem Punkt, an dem die Kugel anhält, ist eine Zufallsvariable X. Die Kugel rollt jedes Mal mindestens 150 cm, aber höchstens 250 cm. Damit kann X alle Werte aus dem Intervall $[150; 250]$ annehmen. Um nun das Wahrscheinliche Verhalten der Kugel trotz stetiger Zufallsvariable zu beschreiben, nutzen wir eine Dichtefunktion von X.

Dichtefunktion einer stetigen Verteilung

Verteilungsfunktion

Verteilungsfunktionen der Zufallsvariable x werden mit $F(x)$ gekennzeichnet. Sie geben an, wie gross die Wahrscheinlichkeit ist, dass die Zufallsvariable einen Wert gleich oder kleiner als x annimmt. Verteilungsfunktionen müssen Werte zwischen 0 und 1 annehmen und monoton steigend sein.

Die Wahrscheinlichkeitsfunktion $\Pr[\cdot]$ induziert die Verteilung von X gemäss:

$$\begin{aligned} F_X: \mathbb{R} &\to [0,1] \\ x &\mapsto \Pr[X \leq x] = \sum_{x' \in W_X : x' \leq x} \Pr[X = x'] \end{aligned}$$

Beispiel:

Wir werfen eine Münze zweimal hintereinander.

- Wenn 2 mal Zahl fällt, verlieren wir 2 Franken.
- Wenn 1 mal Kopf fällt, gewinnen wir 1 Franken.
- Wenn 2 mal Kopf fällt, gewinnen wir 2 Franken.

$$X(\omega) \mapsto \begin{cases} -2 \text{ für } \omega = ZZ \\ 1 \text{ für } \omega = ZK \\ 1 \text{ für } \omega = KZ \\ 2 \text{ für } \omega = KK \end{cases}$$

$$F(x_i) = P(X \leq x_i) = \begin{cases} 0 \text{ für } x < -2 \\ 0.25 \text{ für } -2 \leq x < 1 \\ 0.75 \text{ für } 1 \leq x < 2 \\ 1 \text{ für } x \geq 2 \end{cases}$$

Bernoulli-Verteilung

Ein Bernoulli Experiment ist ein Zufallsexperiment, bei dem man sich nur dafür interessiert, ob ein bestimmtes Ereignis eintritt oder nicht. Es wird also nur Erfolg oder nicht Erfolg betrachtet.

Sei X eine Zufallsvariable mit Wertebereich $W_X = \{0,1\}$ und Dichte $f_X(x) = \begin{cases} p \text{ für } x = 1 \\ 1 - p \text{ für } x = 0 \end{cases}$. Wir nennen X Bernoulli-verteilt und p die Erfolgswahrscheinlichkeit der Bernoulli-Verteilung.

Beispiel: Wirft man eine faire Münze, so gibt es nur die Ereignisse «Kopf» oder «Zahl», wobei beide dieselbe Wahrscheinlichkeit besitzen:

$$p = (1 - p) = \frac{1}{2}$$

Erwartungswert

Sei $X \sim Ber(p)$, so gilt für den Erwartungswert ganz einfach:

$$\mathbb{E}[X] = 1 * p + 0 * (1 - p) = p$$

Varianz

Sei $X \sim Ber(p)$, so gilt für die Varianz:

$$\begin{aligned} Var[X] &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ &= p + 1^2 + (1 - p) * 0^2 - p^2 \\ &= p * (1 - p) \end{aligned}$$

Binomialverteilung

Eine Bernoulli-verteilte Zufallsvariable erhalten wir beispielsweise als Indikator für «Kopf», wenn wir eine Münze einmal werfen. Doch was passiert, wenn wir die Münze stattdessen n mal werfen und uns fragen, wie oft wir Kopf erhalten haben? In diesem Fall sagen wir die entsprechende Zufallsvariable ist binomialverteilt.

Sei X eine Zufallsvariable mit Wertebereicht $W_X = \{0,1,\dots,n\}$ und Dichte:

$$f_X(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x}, x \in \{0,1,\dots,n\} \\ 0, \text{sonst.} \end{cases}$$

Wir nennen X Binomialverteilt mit Parametern n und p.

Beispiel: Wirft man eine faire Münze 8-mal, so ist die Wahrscheinlichkeit für 5-maliges Auftreten von «Kopf»:

$$\begin{aligned} p &= \frac{1}{2} \\ k &= 5 \\ n &= 8 \\ \Pr[X = 5] &= \binom{8}{5} * \frac{1}{2^5} * \frac{1}{2^3} \\ &= \frac{5!}{5!\,(8-5)!} * \frac{1}{2^5} * \frac{1}{2^3} \end{aligned}$$

Erwartungswert

Sei $X \sim Bin(n,p)$, so gilt für den Erwartungswert:

$$\mathbb{E}[X] = \mathbb{E}[X_1 + \cdots + X_n] = \mathbb{E}[X_1] + \cdots + \mathbb{E}[X_n] = n * p$$

Varianz

Sei $X \sim Bin(n,p)$, so gilt für die Varianz:

$$\begin{aligned} Var[X] &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ &= \sum_{k=0}^{n} k^2 \binom{n}{k} p^k (1-p)^{n-k} - n^2 p^2 \\ &= np(1-p) \end{aligned}$$

Geometrische Verteilung

Wenn ein einzelner Versuch mit Wahrscheinlichkeit p gelingt, so ist die Anzahl der Versuche bis zum Erfolg geometrisch verteilt. Wir schreiben dann $X \sim Geo(p)$.

Der Parameter p heisst in einem solchen Fall die Erfolgswahrscheinlichkeit der geometrischen Verteilung und für die Dichte von X gilt:

$$f_X(i) = \begin{cases} p(1-p)^{i-1}, \text{für } i \in \mathbb{N} \\ 0, \text{sonst} \end{cases}$$

Beispiel: Man stelle sich einen betrunkenen Torwärter vor, welcher den passenden Schlüssel in einem Schlüsselbund mit 5 Schlüsseln zu suchen versucht. Nach jedem gescheiterten Versuch lässt er den Schlüsselbund fallen und weiss nicht mehr, welche Schlüssel er bereits probiert hat. Wie hoch ist die Wahrscheinlichkeit, dass der Torwärter den richtigen Schlüssel in höchstens 3 Versuchen findet?

$$Pr[X \leq 3] = \sum_{i=1}^{3} \frac{1}{5}\left(\frac{4}{5}\right)^{i-1}$$

Erwartungswert

Führt man n Experimente durch, so ist der Erwartungswert für die Anzahl der erfolgreichen Experimente $n * p$ (Siehe Binomialverteilung). Daher ist der zu erwartende Abstand zwischen zwei erfolgreichen Experimenten (einschliesslich eines erfolgreichen Experiments) $\frac{n}{n*p}$

Sei $X \sim Geo(p)$, so gilt für den Erwartungswert also $\mathbb{E}[X] = \frac{n}{n*p} = \frac{1}{p}$

Varianz

Sei $X \sim Geo(p)$, so gilt für die Varianz:

$$Var[X] = \frac{(1-p)}{p^2}$$

Poisson Verteilung

Die Poisson Verteilung ist motiviert durch die Betrachtung von Ereignissen, von denen jedes einzelne zwar mit sehr geringer Wahrscheinlichkeit eintritt, wir aber auf Grund der Vielzahl möglicher Ereignisse dennoch erwarten, dass zumindest ein paar Ereignisse eintreten.

Formal ist eine Poisson Verteilung (mit Parameter λ) definiert durch die Dichtefunktion

$$f_X(i) = \begin{cases} \frac{e^{-\lambda}\lambda^i}{i!} & \text{für } i \in \mathbb{N}_0 \\ 0 & \text{sonst} \end{cases}$$

Beispiel: Das Restaurant Dante Pizza führt Buch über die Anzahl an Gästen, die das Restaurant betreten. Laut der Aufzeichnung ist der Erwartungswert $\mathbb{E}[X] = 12.1$ zwischen $20{:}00$ und $22{:}00$ Uhr. Wie hoch ist die Wahrscheinlichkeit, dass zu dieser Zeit genau 8 Gäste das Restaurant betreten?

$$\Pr[X = 8] = \frac{e^{-12.1} * 12.1^8}{8!} \approx 0.0634$$

Erwartungswert

Sei $X \sim Poi(\lambda)$, so gilt für den Erwartungswert direkt aus Definition von λ:

$$\mathbb{E}[X] = \lambda$$

Varianz

Sei $X \sim Poi(\lambda)$, so gilt für die Varianz:

$$Var[X] = \lambda$$

Erwartungswert

Wir definieren den Erwartungswert $\mathbb{E}[X]$ einer Zufallsvariablen X durch:

$$\mathbb{E}[X] \coloneqq \sum_{x \in W_X} x * \Pr[X = x]$$

Falls die Summe absolut konvergiert. Ansonsten sagen wir, der Erwartungswert sei undefiniert.

Sei X eine Zufallsvariable mit $W_X \subseteq \mathbb{N}_0$. Dann gilt:

$$\mathbb{E}[X] = \sum_{i=1}^{\infty} \Pr[X \geq i]$$

Linearität des Erwartungswertes

Für Zufallsvariablen $X_1, \dots, X_n$ und $X \coloneqq a_1 X_1 + \cdots + a_n X_n + b$ mit $a_1, \dots, a_n, b \in \mathbb{R}$ gilt:

$$\mathbb{E}[X] = a_1 \mathbb{E}[X_1] + \cdots + a_n E[X_n] + b$$

Die Linearität des Erwartungswertes ermöglicht es oft, den Erwartungswert einer Zufallsvariablen sehr einfach zu berechnen, obwohl dies auf den Ersten Blick nicht möglich scheint.

Beispiel:

Betrachten wir einen idealen m-fachen Münzwurf. Mit X bezeichnen wir die Anzahl Teilfolgen, die aus dreimal Kopf bestehen. Da die Vorkommen von KKK überlappen können, ist es auf den ersten Blick nicht klar, wie man $\mathbb{E}[X]$ berechnet.

Man beschreibe nun X als Summe von vielen Zufallsvariablen, deren Erwartungswerte wir leicht berechnen können. Zunächst überlegen wir uns, dass eine Teilfolge KKK an jeder der Positionen $1, \dots, m-2$ starten kann. Für jede dieser Positionenen definieren wir eine Variable, die angibt, ob an dieser Stelle eine Teilfolge KKK startet:

$$X_i \coloneqq \begin{cases} 1, \text{falls an Stelle } i \text{ eine Teilfolge } KKK \text{ beginnt} \\ 0, \text{sonst} \end{cases}$$

Für die gesuchte Zufallsvariable X gilt dann: $X = X_1 + \cdots + X_{m-2}$. Der Erwartungswert von X_i lässt sich einfach berechnen, da X_i ja nur zwei verschiedene Werte annehmen kann:

$$\mathbb{E}[X] = 0 * \Pr[X_i = 0] + 1 * \Pr[X_i = 1] = \Pr[X_i = 1]$$

Nach Definition ist $X_i = 1$ genau dann, wenn an der Stelle i eine Teilfolge KKK beginnt, wenn also der $i$te und der $(i+1)$te und der $(i+2)$te Münzwurf jeweils Kopf ergeben haben. Die Wahrscheinlichkeit hierfür ist $\left(\frac{1}{2}\right)^3$. Somit gilt $\mathbb{E}[X_i] = \frac{1}{8}$ für alle $i = 1, \dots, m-2$ und daher $\mathbb{E}[X] = \frac{(m-2)}{8}$.

Verteiltes Rechnen: Algorithmus

Es sei ein Graph $G = (V, E)$ mit $|V| = n$, $|E| = m$ gegeben. Wir versuchen eine (möglichst grosse) stabile Menge zu finden.

$\Rightarrow$ Eine möglichst grosse Teilmenge $S \subseteq V$, so dass $G[S]$ keine Kante enthält.

Setze $S_v \longleftarrow 1$ mit Wahrscheinlichkeit p und $S_v \longleftarrow 0$ sonst;

for all $u \in V$ mit $\{u, v\} \in E$ do tausche Nachricht mit P_u aus:

    if $S_u = S_v = 1$ then

        setze einen der Werte S_v und S_u auf Null.

return S_v

Man beobachte: Dieser Algorithmus ist nicht deterministisch; wir können also nicht genau sagen, wie gross die berechnete Menge S sein wird. Was wir aber tun können, ist einen Erwartungswert festzulegen.

Als erstes bezeichnen wir mit X_v den Wert, der in Zeile 1 für S_v gewählt wird. Ausserdem setzen wir:

$$X \coloneqq \sum_{v \in V} X_v$$

Für eine Kante $e = \{u, v\}$ definieren wir die Indikatorvariable Y_e, die genau dann 1 ist falls sowohl P_u als auch P_v ihre Werte S_u bzw. S_v zunächst auf 1 gesetzt haben, falls also $X_u = X_v = 1$ ist. Ausserdem sei noch $Y \coloneqq \sum_{e \in E} Y_v$. Nun beobachten wir, dass $S \geq X - Y$ gilt, denn wir konstruieren S ja dadurch, dass wir mit X Knoten anfangen, und dann noch für jede Kante e mit $Y_e = 1$ höchstens einen Knoten löschen. Wegen der Linearität des Erwartungswertes gilt nun $\mathbb{E}[S] \geq \mathbb{E}[X] - \mathbb{E}[Y]$.

Beide Erwartungswerte rechnen wir leicht aus, indem wir wieder die Linearität des Erwartungswertes benutzen. Zum einen ist $\mathbb{E}[X] = \sum_{v \in V} \mathbb{E}[X_v] = np$ und zum Anderen $\mathbb{E}[Y] = \sum_{e \in E} \mathbb{E}[Y_e] = mp^2$. Daher gilt $\mathbb{E}[S] \geq np - mp^2$.

Durch Ableiten sehen wir, dass dieser Erwartungswert maximiert wird, wenn $2mp = n$ ist, das heisst $p = \frac{n}{2m}$. Nehmen wir nun noch an, dass der Graph G d-regulär ist, so gilt $2m = dn$ und daher $p = \frac{1}{d}$. Damit erhalten wir $\mathbb{E}[S] \geq \frac{n}{d} - \frac{n}{2d} = \frac{n}{2d}$. Insbesondere hat daher jeder d-reguläre Graph eine stabile Menge der Grösse mindestens $\frac{n}{2d}$.

Varianz

Wenn zwei Zufallsvariablen denselben Erwartungswert besitzen, so können sie sich dennoch deutlich voneinander unterscheiden. Ein besonders wichtiges Merkmal einer Verteilung ist daher die Streuung um den Erwartungswert.

Beispiel: Man werde vor eine Entscheidung gestellt. Entweder wird eine Münze geworfen und falls diese auf Kopf landet, so erhält man 100CHF, wenn nicht 0CHF. Oder man nimmt 50CHF «auf sicher». Wir modellieren diese Situation nun mathematisch:

$$\Pr[X = 100\ ] = \frac{1}{2}$$
$$\Pr[X = 0] = \frac{1}{2}$$
$$\Pr[Y = 50] = 1$$

Schnell fällt auf: $\mathbb{E}[X] = 50$ und $\mathbb{E}[Y] = 50$. Obwohl der Erwartungswert bei beiden Entscheidungen identisch ist, würden viele sich für die «sichere» Option entscheiden. Offensichtlich sind also doch nicht beide Werte identisch; mindestens nicht gefühlt.

Das Beispiel zeigt, dass es bei vielen Zufallsvariablen sinnvoll ist, die zu erwartende Abweichung vom Erwartungswert zu untersuchen.

Für eine Zufallsvariable X mit $\mu = \mathbb{E}[X]$ definieren wir die Varianz $Var[X]$ durch

$$Var[X] \coloneqq \mathbb{E}[(X-\mu)^2] = \sum_{x \in W_X} (x-\mu)^2 * \Pr[X = x]$$

Die Grösse $\sigma \coloneqq \sqrt{Var[X]}$ heisst Standardabweichung von X.

Diese eben definierte Varianz und Standardabweichung erlaubt uns nun, genauere Aussagen über den Erwartungswert zu treffen.

Zurück zu unserem Beispiel: Wir erhalten für unsere Zufallsvariablen folgende Standardabweichungen:

$$Var[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \frac{2500}{2} + \frac{2500}{2} = 2500$$
$$\Rightarrow \sigma[X] = \sqrt{2500} = 50$$
$$Var[Y] = \mathbb{E}[(Y - \mathbb{E}[Y])^2] = 0$$
$$\Rightarrow \sigma[Y] = \sqrt{0} = 0$$

Unter Betrachtung der Varianz sehen wir also: Die beiden Entscheidungen sind keineswegs identisch!

Rechenregeln

1. $Var[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$
2. $Var[a * X + b] = a^2 * Var[X]$

Beweise:

1.

$$\begin{aligned} Var[X] = \mathbb{E}[(X-\mu)^2] &= \mathbb{E}[X^2 - 2\mu * X + \mu^2] \\ &= \mathbb{E}[X^2] - 2\mu * \mathbb{E}[X] + \mu^2 \\ &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \end{aligned}$$

2.

$$Var[X+b] = \mathbb{E}[(X+b-\mathbb{E}[X+b])^2] = \mathbb{E}[(X-\mathbb{E}[X])^2] = Var[X]$$

Ausserdem:

$$Var[a * X] = \mathbb{E}[(aX)^2] - \mathbb{E}[aX]^2 = a^2\mathbb{E}[X^2] - (a\mathbb{E}[X])^2 = a^2 * Var[X]$$

■

Mehrere Zufallsvariablen

Oftmals interessieren wir uns für mehrere Zufallsvariablen gleichzeitig. Wir beschäftigen uns dann mit der Frage, wie man mit mehreren Zufallsvariablen über demselben Wahrscheinlichkeitsraum rechnen kann, auch wenn sie sich gegenseitig beeinflussen.

Seien X und Y zwei Zufallsvariablen. $\Pr[X = x, Y = y] \coloneqq \Pr[\{\omega \in \Omega | X(\omega) = x, Y(\omega) = y\}]$.

Gemeinsame Dichte

Die Funktion $f_{X,Y}(x,y) \coloneqq \Pr[X = x, Y = y]$ heisst gemeinsame Dichte der Zufallsvariablen X und Y. Hat man die gemeinsame Dichte gegeben, so kann man auch wieder zu den Dichten der einzelnen Zufallsvariablen übergehen, indem man

$$f_X(X) = \sum_{x \in W_Y} f_{X,Y}(x,y) \text{ bzw. } f_Y(y) = \sum_{x \in W_X} f_{X,Y}(x,y)$$

setzt.

Gemeinsame Verteilung

Für zwei Zufallsvariablen X und Y definiert man die gemeinsame Verteilung als

$$\begin{aligned} F_{X,Y}(x,y) \coloneqq \Pr[X \leq x, Y \leq y] &= \Pr[\{\omega \in \Omega | X(\omega) \leq x, Y(\omega) \leq y\}] \\ &= \sum_{x' \leq x} \sum_{y' \leq y} f_{X,Y}(x', y') \end{aligned}$$

Unabhängigkeit von Zufallsvariablen

Wie Ereignisse können auch Zufallsvariablen unabhängig sein. Manchmal ist diese Unabhängigkeit offensichtlich, sie kann aber auch nur mathematisch bestehen.

Beispiel 1: Wir werfen eine Münze 2 mal. Mit X_1 bzw. X_2 bezeichnen wir die Indikatorvariable für das Ereignis, dass der erste bzw. zweite Wurf «Kopf» ist. Da der erste Wurf keinen Einfluss auf den zweiten Wurf hat, gilt für alle $a, b \in \{0,1\}$, dass $\Pr[X_1 = a, X_2 = b] = \Pr[X_1 = a] * \Pr[X_2 = b]$.

Beispiel 2: Wir ziehen zufällig eine Karte aus einem Skatspiel mit 32 Karten und betrachten die Indikatorvariablen X und Y für die folgenden beiden Ereignisse: «Die Karte ist eine Herz-Karte» bzw. «die Karte ist ein Bube». Dann sind die beiden Variablen offensichtlich nicht «physikalisch» unabhängig, denn wir ziehen ja nur eine einzige Karte, es gilt aber:

$$\Pr[Y = 1|X = 1] = \frac{1}{8} = \frac{4}{32} = \Pr[Y = 1]$$

Das heisst, ob wir wissen, dass die Karte eine Herz-Karte ist, hat keinen Einfluss auf die Wahrscheinlichkeit, mit der die Karte ein Bube ist.

Zufallsvariablen $X_1, \dots, X_n$ heissen unabhängig, genau wenn für alle $(x_1, \dots, x_n) \in W_{X_1} \times \dots \times W_{X_n}$ gilt, dass $\Pr[X_1 = x_1, \dots, X_n = x_n] = \Pr[X_1 = x_1] \dots \Pr[X_n = x_n]$.

Zusammengesetzte Zufallsvariablen

Wir können mehrere Zufallsvariablen auf einem Wahrscheinlichkeitsraum miteinander kombinieren; also aus den Zufallsvariablen $X_1, \dots X_n$ eine neue, zusammengesetzte Zufallsvariable Y durch $Y \coloneqq g(X_1, \dots X_n)$. Definieren.

Für zwei unabhängige Zufallsvariablen X und Y sei $Z \coloneqq X + Y$. Es gilt:

$$f_Z(z) = \sum_{x \in W_X} f_X(x) * f_Y(z - x)$$

Beweis:

$$f_Z(z) = \Pr[Z = z] = \sum_{x \in W_X} \Pr[X + Y = z|X = x] * \Pr[X = x]$$
$$= \sum_{x \in W_X} \Pr[Y = z - x] * \Pr[X = x] = \sum_{x \in W_X} f_X(x) * f_Y(z - x)$$

(Durch Anwendung der totalen Wahrscheinlichkeit) ■

Abschätzen von Wahrscheinlichkeiten

Bei der Betrachtung von Zufallsexperimenten möchten wir unter Umständen untersuchen, welches Ergebnis wir bei der Durchführung des Experiments erwarten. Eine mögliche Interpretation dazu haben wir bereits kennengelernt: Der Erwartungswert $\mathbb{E}[X]$ einer Zufallsvariable entspricht dem Durchschnittswert bei sehr vielen Wiederholungen des Experiments. Wenn wir aber von unserer Erwartung sprechen, so beziehen wir uns in der Regel nicht auf den Schnitt aus vielen, sondern nur auf eine einzige Durchführung des Experiments.

Beispiel:

Wir betrachten eine Zufallsvariable X mit Dichte $f_X(x) = \begin{cases} 1 - \frac{1}{n}, \text{falls } x = n \\ \frac{1}{n}, \text{falls } x = n(n-1) \\ 0 \text{ sonst} \end{cases}$. In diesem Fall ist $\mathbb{E}[X] = n\left(1 - \frac{1}{n}\right) - (n-1) = 0$. Dennoch nimmt aber X mit Wahrscheinlichkeit $1 - \frac{1}{n}$ den Wert n an.

Ungleichung von Markov

Sei X eine Zufallsvariable, die nur nicht-negative Werte annimmt. Dann gilt für alle $t \in \mathbb{R}$ mit $t > 0$, dass $\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$. Oder äquivalent dazu $\Pr\left[X \geq t * \mathbb{E}[X]\right] \leq \frac{1}{t}$.

Beispiel:

Der Erwartungswert für «Anzahl Tore» einer Fussball-Mannschaft bei Heimspielen sei 2.

- **Abweichung vom Erwartungswert nach oben:** Wie hoch ist die Wahrscheinlichkeit, dass die Mannschaft z.B. 6 Tore oder mehr in einem Heimspiel schiesst?
  $\Pr[X \geq 6] \leq \frac{2}{6} = \frac{1}{3} \approx 33.33\%$
- **Abweichung vom Erwartungswert nach unten:** Wie hoch ist die Wahrscheinlichkeit, dass die Mannschaft weniger als 6 Tore schiesst?
  Die Wahrscheinlichkeit, dass 6 oder mehr Tore geschossen werden, war höchstens $\frac{1}{3}$. Die Gegenwahrscheinlichkeit entspricht demnach $1 - \frac{1}{3} = \frac{2}{3} \approx 66.66\%$

Beweis:

$$\begin{aligned} \mathbb{E}[X] = \sum_{x \in W_X} x * \Pr[X = x] &\geq \sum_{x \in W_X, x \geq t} x * \Pr[X = x] \\ &\geq t * \sum_{x \in W_X, x \geq t} \Pr[X = x] = t * \Pr[X \geq t] \\ &\Rightarrow \Pr[X \geq t] * t \leq \mathbb{E}[X] \\ &\Rightarrow \Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t} \end{aligned}$$

■

Ungleichung von Chebychev

Sie X eine Zufallsvariable und $t \in \mathbb{R}$ mit $t > 0$. Dann gilt:

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{Var[X]}{t^2}.$$

Oder äquivalent dazu $\Pr\left[|X - \mathbb{E}[X]| \geq t\sqrt{Var[X]}\right] \leq \frac{1}{t^2}$

Beispiel: Angenommen, wir interessieren uns für einen Hundekauf und betrachten eine Hunderasse mit Durchschnittsgewicht $\mathbb{E}[X] = 35.2kg$ und Standardabweichung $\sigma = 3.5kg$. Wir möchten nun die Wahrscheinlichkeit bestimmen, mit der unser zukünftiger Hund eine Mindestabweichung vom mittleren Gewicht in Höhe von mindestens $5kg$ aufweist.

$$\Pr[|X - 35.2kg| \geq 5kg] \leq \frac{(3.5kg)^2}{(5kg)^2} = \frac{12.25kg}{25kg} = 49\%$$

Beweis:

$$\Pr[|X - \mathbb{E}[X]| \geq t] = \Pr[(X - \mathbb{E}[X])^2 \geq t^2]$$

Die Zufallsvariable $Y \coloneqq (X - \mathbb{E}[X])^2$ ist nichtnegativ und hat nach Definition der Varianz den Erwartungswert $\mathbb{E}[Y] = Var[X]$. Damit folgt die Behauptung durch Anwendung der Markov-Ungleichung.

$$\Pr[X - \mathbb{E}[X] \geq t] = \Pr[Y \geq t^2] \leq \frac{\mathbb{E}[Y]}{t^2} = \frac{Var[X]}{t^2}$$

■

Chernoff Schranken

Wenn wir eine faire Münze n mal werfen und mit X die Anzahl Kopf bezeichnen, so ist X binomialverteilt mit Parameter $\frac{1}{2}$. Insbesondere gilt daher $\mathbb{E}[X] = \frac{n}{2}$. Mit der Ungleichung von Markov folgt daraus $\Pr[X \geq n] \leq \frac{1}{2}$. Ein etwas schärferes Resultat liefert die Chebyshev-Ungleichung:

$$X \sim Bin\left(n, \frac{1}{2}\right)$$
$$\Rightarrow Var[X] = n * p(1-p) = \frac{1}{4}n$$

$$\begin{aligned}\Pr[X \geq n] &\leq \Pr\left[|X - \mathbb{E}[X]| \geq n - \mathbb{E}[X]\right] \\ &= \Pr\left[\left|X - \frac{n}{2}\right| \geq n - \frac{n}{2}\right] \\ &= \frac{\frac{n}{4}}{\frac{n^2}{4}} \\ &= \frac{4n}{4n^2} = \frac{1}{n}\end{aligned}$$

Wir wissen aber andererseits bereits, dass $\Pr[X \geq n] = 2^{-n}$ gilt. Die Markov- und Chebyshev-Ungleichungen liefern in diesem Beispiel also zwar korrekte, aber nicht sehr gute Abschätzungen. Dies liegt daran, dass diese Ungleichungen für beliebige nichtnegative Zufallsvariablen gelten – sie sind daher leicht anzuwenden, geben aber nicht immer sehr gute Schranken. Wenn wir die Verteilung der Zufallsvariable kennen, können wir oft bessere Schranken herleiten. Speziell für Summen von Bernoulli-Variablen sind die sogenannten Chernoff-Schranken sehr nützlich.

Seien $X_1, \ldots, X_n$ unabhängige Bernoulliverteilte Zufallsvariablen mit $\Pr[X_i = 1] = p_i$ und $\Pr[X_i = 0] = 1 - p_i$. Dann gilt für $X \coloneqq \sum_{i=1}^{n} X_i$:

1. $\Pr\left[X \geq (1+\sigma)\mathbb{E}[X]\right] \leq e^{-\frac{1}{3}\sigma^2\mathbb{E}[X]}$ für alle $0 < \sigma \leq 1$
2. $\Pr\left[X \leq (1-\sigma)\mathbb{E}[X]\right] \leq e^{-\frac{1}{2}\sigma^2\mathbb{E}[X]}$ für alle $0 < \sigma \leq 1$
3. $\Pr[X \geq t] \leq 2^{-t}$ für $t \geq 2e\mathbb{E}[X]$

Beweis der Chernoff Schranken

Da der Beweis der Chernoff-Schranken etwas länger und komplexer ist als der der anderen Ungleichungen möchte ich ihm ein eigenes Kapitel widmen. Dieser Beweis ist **NICHT** Prüfungsrelevant, erscheint mir aber als wertvolle Übung und trägt zum Verständnis der Chernoff-Schranken bei.

Betrachten wir die sogenannte e-Funktion, welche wie folgt definiert ist:

$$\begin{aligned} f: \mathbb{R} &\to \mathbb{R} \\ x &\mapsto e^x \end{aligned}$$

Schnell fällt auf, dass es sich dabei um eine streng monoton wachsende Funktion handelt. Damit gilt für jedes $t > 0 \in \mathbb{R}$, dass X genau dann grösser als ein Wert Y ist, wenn $e^{tX} > e^{tY}$. Wir definieren nun

$$y \coloneqq (1+\sigma) * \mathbb{E}[X].$$

Nach Markov beträgt die Wahrscheinlichkeit, dass die Zufallsvariable X einen Wert von mindestens y annimmt demnach

$$\begin{aligned} \Pr\left[X \geq (1+\sigma) * \mathbb{E}[X]\right] &= \Pr\left[e^{tX} \geq e^{t(1+\sigma)\mathbb{E}[X]}\right] \\ &\leq \frac{\mathbb{E}[e^{tX}]}{e^{t(1+\sigma)\mathbb{E}[X]}} \end{aligned}$$

Die Zufallsvariablen sind als unabhängig definiert. Damit können wir $\mathbb{E}[e^{tX}]$ mit Hilfe der Multiplikativität des Erwartungswertes berechnen und erhalten

$$\mathbb{E}[e^{tX}] = \mathbb{E}\left[e^{\sum_{i=1}^{n} tX_i}\right] = \mathbb{E}\left[\prod_{i=1}^{n} e^{tX_i}\right] = \prod_{i=1}^{n} \mathbb{E}[e^{tX_i}].$$

Nun wissen wir: jedes X_i ist eine Bernoulli-Variable mit Parameter p_i. Daher gilt:

$$\mathbb{E}[e^{tX}] = e^{t*1}p_i + e^{t*0}(1-p_i) = e^t p_i + 1 - p_i = 1 + p_i(e^t - 1) \overset{1+x \leq e^x}{\leq} e^{p_i(e^t-1)}$$

Damit haben wir

$$\Pr\left[X \geq (1-\sigma)\mathbb{E}[X]\right] \leq \frac{\prod_{i=1}^{n} e^{p_i(e^t-1)}}{e^{t(1+\sigma)\mathbb{E}[X]}} = \frac{e^{\sum_{i=1}^{n} p_i(e^t-1)}}{e^{t(1+\sigma)\mathbb{E}[X]}} = \frac{e^{(e^t-1)\mathbb{E}[X]}}{e^{t(1+\sigma)\mathbb{E}[X]}} =: f(t)$$

Wir wählen für t nun den Wert, der die rechte Seite minimiert, um diese Aussage so genau wie möglich zu machen. Dazu müssen wir den Ausdruck ableiten und gleich Null setzen, um die Extremstelle zu finden.

$$\begin{aligned} f'(t) &= f(t) * \mathbb{E}[X] * \left(e^t - (1+\sigma)\right) \\ &\Rightarrow 0 = f(t_0) * \mathbb{E}[X] * \left(e^{t_0} - (1+\sigma)\right) \\ &\Rightarrow t_0 = \log(1+\sigma) \end{aligned}$$

Daraus folgen dann die Ungleichungen (1) und (3) durch Einsetzen und Verwenden von

$$(1+\sigma)^{1+\sigma} \geq e^{\sigma + \frac{\sigma^2}{3}}$$

für alle $0 \leq \sigma \leq 1$ bzw. $\frac{e}{(1+\sigma)} \leq \frac{1}{2}$ für alle $t \geq 2e\mu$. Ungleichung (2) folgt ähnlich. Die Details lassen wir hier weg.

Randomisierte Algorithmen

Ziel eines Algorithmus ist es, für eine Eingabe I eine Ausgabe $\mathcal{A}(I)$ zu berechnen. Bislang haben wir nur Algorithmen gesehen, die die Eigenschaft haben, dass sie für identische Eingaben auch identische Ergebnisse berechnen. Man nennt solche Algorithmen daher auch deterministische Algorithmen. In diesem Abschnitt wollen wir einen Schritt weiter gehen und sogenannte randomisierte Algorithmen betrachten. Die Idee eines randomisierten Algorithmus ist, dass der Algorithmus zusätzlich zur Eingabe I auch noch Zugriff auf Zufallsvariablen hat. Das Ergebnis des Algorithmus hängt daher nicht nur von der Eingabe I ab, sondern auch noch von den Werten der Zufallsvariablen. Wir bezeichnen die Ausgabe des Algorithmus daher mit $\mathcal{A}(I, \mathcal{R})$, wobei $\mathcal{R}$ für die Werte der Zufallsvariablen steht. Ausserdem schreiben wir $\mathcal{A}(I)$ für die Zufallsvariable, die dem Wert von $\mathcal{A}(I, \mathcal{R})$ bei einer zufälligen Wahl von $\mathcal{R}$ entspricht. Wichtig ist daher: bei randomisierten Algorithmen erhalten wir bei mehrmaligem Aufruf mit der gleichen Eingabe I im Allgemeinen verschiedene Ergebnisse.

Man unterscheidet im Wesentlichen zwischen zwei Arten von randomisierten Algorithmen. Die Monte Carlo Algorithmen und die Las Vegas Algorithmen.

Monte Carlo Algorithmen

Monte Carlo Algorithmen zeichnen sich dadurch aus, dass die Laufzeit des Algorithmus immer beschränkt ist; aber: Monte Carlo Algorithmen dürfen Fehler machen!

Fehlerreduktion

Es ist generell sehr schwer den Fehler eines Monte Carlo Algorithmus zu minimieren; schliesslich sieht man einer Antwort nicht an , ob sie korrekt ist oder nicht.

In dem Spezialfall, dass ein Monte Carlo Algorithmus nur zwei Werte ausgibt und wir wissen, dass einer dieser Werte immer korrekt ist, können wir den Fehler des Algorithmus aber reduzieren.

Sei $\mathcal{A}$ ein randomisierter Monte Carlo Algorithmus für ein Entscheidungsproblem (Antwort «Ja» oder «Nein») mit Laufzeit $O\big(f(n)\big)$. Weiter gelte

$$\Pr[\mathcal{A}(I) = "Ja"] = 1\,, \forall_I\!: Ja\ ist\ korrekt$$

und

$$\Pr[\mathcal{A}(I) = "Nein"] \geq \epsilon, \forall_I\!: Nein\ ist\ korrekt$$

Dann gilt für alle $\sigma > 0$: Sei $\mathcal{A}_\sigma$ der Algorithmus, der $\mathcal{A}$ solange aufruft bis entwerder der Wert «Nein» ausgegeben wird oder bis $N = \epsilon^{-1} \ln(\sigma^{-1})$ mal «Ja» ausgegeben wurde, so gilt für alle Instanzen I

$$\Pr[\mathcal{A}_\sigma(I)\ korrekt] \geq 1 - \sigma$$

Monte Carlo Algorithmen, die diese Bedingung (zwei mögliche Ausgaben, eine ist immer korrekt) erfüllen, nennen wir Algorithmen mit einseitigem Fehler, denn der potenzielle Fehler kann nur bei einer der beiden Antworten auftreten.

Bei Monte Carlo Algorithmen mit zweiseitigem Fehler können wir ebenfalls eine Fehlerreduktion durchführen, aber nur, wenn die Fehlerwahrscheinlichkeit des Algorithmus strikt kleiner als $\frac{1}{2}$ ist

⇒ Intuitiv erklärt sich das dadurch, dass wir zumindest eine gewisse Sicherheit haben müssen, dass unser Resultat richtig ist, ansonsten können wir keinerlei Aussagen treffen, denn beide Resultate könnten theoretisch mit beliebig kleiner Wahrscheinlichkeit korrekt sein.

Las Vegas Algorithmen

Las Vegas Algorithmen zeichnen sich dadurch aus, dass die Laufzeit des Algorithmus immer beschränkt ist; aber: die Antwort, welche ein Las Vegas Algorithmus liefert, ist entweder korrekt oder unbestimmt.

⇒ Wollen wir sicher eine korrekte Antwort, so müssen wir den Algorithmus unter Umständen mehrmals wiederholen. Die Laufzeit, welche eine korrekte Antwort garantiert, ist also unbeschränkt!

Fehlerreduktion

Sei $\mathcal{A}$ ein randomisierter Las Vegas Algorithmus mit Laufzeit $O\big(f(n)\big)$ und $\Pr[\mathcal{A}(I)\ korrekt] \geq \epsilon, \forall_I$ wobei $\epsilon > 0$ aber beliebig klein ist.

Wir möchten nun einen Algorithmus $\mathcal{A}'$ mit Laufzeit $O\big(f(n)\big)$ und $\Pr[\mathcal{A}'(I)\ korrekt] \geq 1 - \sigma\,, \forall_I$ für ein sehr kleines $\sigma > 0$ finden.

Sei $\mathcal{A}$ ein randomisierter Algorithmus, der nie eine falsche Antwort gibt, aber unter umständen unbestimmt $(???)$ endet, wobei

$$\Pr[\mathcal{A}(I)\ korrekt] \geq \epsilon$$

Dann gilt für alle $\sigma > 0$: Sei $\mathcal{A}_\sigma$ der Algorithmus, der $\mathcal{A}$ solange aufruft bis entweder ein Wert verschieden von $(???)$ ausgegeben wird oder bis $N = \epsilon^{-1} * \ln(\sigma^{-1})$ mal $(???)$ ausgegeben wurde, so gilt für den Algorithmus $\mathcal{A}_\sigma$, dass

$$\Pr[\mathcal{A}_\sigma(I)\ korrekt] \geq 1 - \sigma$$

Beweis: Die Wahrscheinlichkeit, dass $\mathcal{A}$ bei N Aufrufen immer den Wert $(???)$ ausgibt, ist nach Annahme höchstens $(1-\epsilon)^N$. Da $1 - x \leq e^{-x}$ für alle $x \in \mathbb{R}$ folgt daher für $N = \epsilon^{-1}\ln(\sigma^{-1})$, dass die Wahrscheinlichkeit, dass $\mathcal{A}_\sigma$ den Wert $(???)$ ausgibt, höchstens $(1-\epsilon)^N \leq e^{-\epsilon N} = e^{\ln(\sigma)} = \sigma$ ist.

■

Beispielsalgorithmen

Quickselect

Ziel: Finde das k-kleinste Element aus einem unsortierten Array

Beispiel: $Select([12,3,33,67,8,15,19,13],4): \rightarrow 13$

Funktionsweise: Quickselect verwendet den gleichen Gesamtansatz wie Quicksort, wählt ein Element als Pivot und teilt die Daten in zwei Teile, basierend auf dem Pivot, entsprechend kleiner oder grösser als der Pivot. Anstatt jedoch, wie bei Quicksort, in beide Seiten zurückzukehren, kehrt die Schnellauswahl nur in eine Seite zurück – die Seite mit dem gesuchten Element. Dies reduziert die durchschnittliche Komplexität von $O(n * \log(n))$ auf $O(n)$ mit einem Worst-Case von $O(n^2)$

Code:

QuickSelect(A, l, r, k)

$p \leftarrow$ Uniform $(\{l, l+1, \dots, r\})$
$t \leftarrow$ Partition (A, l, r, p)
if $k = t$ then
    return $A[t]$
else if $k < t$ then
    return QuickSelect$(A, l, t-1, k)$
else
    return QuickSelect$(A, t+1, r, k-t)$

π-Berechnung durch Target-Shooting

Ziel: Finde eine möglichst genaue Approximation der Kreiszahl π.

Funktionsweise: Man stelle sich einen Kreis mit Mittelpunkt $(0,0)$ und Radius 1 vor. Es werden zufällig Punkte erzeugt, bei denen sowohl x als auch y im Intervall $[0,1[$ liegen. Dann wird die Entfernung dieser Punkte zum Ursprung ermittelt. Ist diese Entfernung kleiner als 1, so liegt der Punkt innerhalb des Kreises. Setzt man bei einer ausreichenden Zahl von Zufallspunkten die Zahl der Treffer in das richtige Verhältnis zur Gesamtzahl der Punkte, so erhält man einen Näherungswert für π.

$$\pi = 4 * \frac{Treffer}{W\ddot{u}rfe}$$

Code:

TargetShooting

Wähle $U_1, \dots, U_N \in U$ zufällig, gleichverteilt und unabhängig
return $N^{-1} * \sum_{i=1}^{N} I_s(u_i)$

Kargers Algorithmus

Ziel: Gegeben sei ein Multigraph G. Wir wollen die Kardinalität des minimalen Schnittes ($\mu(G)$) bestimmen. Also für eine kleinste Menge C von Kanten, deren Entfernung einen unzusammenhängenden Multigraphen erzeugt.

Funktionsweise: Sei n die Anzahl der Knoten in G. Für die Implementierung setzen wir folgendes voraus:

- Eine Kantenkontraktion kann in $O(n)$ Zeit durchgeführt werden.
- Eine gleichverteilt zufällige Kante in G kann in $O(n)$ gewählt werden.

Mit dieser Voraussetzung können wir $Cut(G)$ mit einer Laufzeit von $O(n^2)$ implementieren indem wir laufend gleichverteil zufällige Kanten in G auswählen und diese kontrahieren. Wir wissen dabei, dass eine zufällige Wahl von e mit guter Wahrscheinlichkeit die Grösse des minimalen Schnitts nicht verändert.

Sei $G = (V, E)$ ein Multigraph mit n Knoten. Falls e gleichverteilt zufällig unter den Kanten in G gewählt wird, dann gilt

$$\Pr[\mu(G) = \mu(G/e)] \geq 1 - \frac{2}{n}$$

Beweis: Sei C ein minimaler Schnitt in G und sei $k \coloneqq |C| = \mu(G)$. Sicherlich ist der Grad jedes Knoten in G mindestens k, da die zu einem Knoten inzidenten Kanten immer einen Schnitt bilden. Es gilt daher

$$|E| = \frac{1}{2}\sum_{v \in V} \deg(v) \geq \frac{kn}{2}$$

Wir erinnern uns an $e \notin C \Rightarrow \mu(G/e) = \mu(G)$ und somit

$$\Pr[\mu(G) = \mu(G/e)] \geq \Pr[e \notin C] = 1 - \frac{|C|}{|E|} \geq 1 - \frac{k}{\frac{kn}{2}} = 1 - \frac{2}{n}$$

■

Code:

Cut(G)

while $|V(G)| > 2$ do
    $e \leftarrow$ gleichverteilt zufällige Kante in G
    $G \leftarrow G/e$
return Grösse des eindeutigen Schnitts in G

Miller-Rabin Primzahltest

Ziel: Für eine beliebige Zahl n wollen wir prüfen, ob es sich um eine Primzahl handelt (Beispielsweise zu Zwecken der Verschlüsselung von Daten).

Beispiel:

$$MillerRabinTest(12) \rightarrow false$$
$$MillerRabinTest(13) \rightarrow true$$

Funktionsweise: Wenn n eine Primzahl ist, dann bilden die Zahlen $0 \leq a < n$ bezüglich der Addition und Multiplikation Modulo n einen Körper. Das heisst insbesondere, dass die Kongruenz $x^2 \equiv mod\ n$ für $0 \leq x < n$ genau die zwei Lösungen $x = 1$ und $x = n - 1$ hat.

Wir bestimmen nun eine ungerade Zahl u und eine Zahl t, so dass gilt $2^t * u = (n - 1)$. Das funktioniert immer, da wir davon ausgehen können, dass n ungerade ist (ansonsten könnten wir direkt sagen, dass es keine Primzahl sein kann). Deshalb ist $n - 1$ stets gerade. Berechnen wir nun $a^u\ mod\ n$, dann quadrieren wir dieses Ergebnis t mal. Nach jedem Quadrieren des Zwischenergebnisses x prüfen wir jedoch zusätzlich, ob das Ergebnis 1 ist. War das zuvor berechnete $x \neq 1$ und $x \neq -1 = n - 1$, so ist ein Zertifikat (whitness) gefunden, durch das n als zusammengesetzt und damit nicht prim gilt. Zum Schluss wird noch der Fermat-Test ausgeführt

Fermat-Test: Ist $n \in \mathbb{N}$ prim, so gilt für alle Zahlen $0 < a < n$

$$a^{n-1} \equiv 1\ mod\ n$$

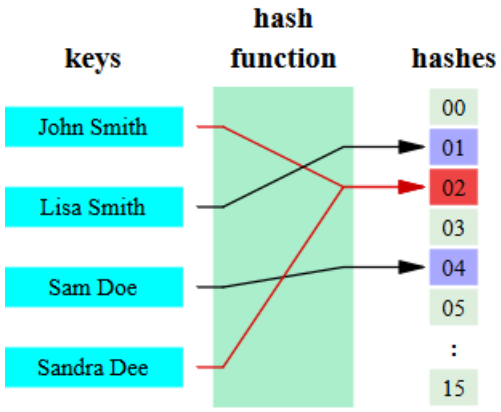
Code:

Miller-Rabin-Primzahltest(n)

```
if n = 2 then
        return true;
else if n gerade oder n = 1 then
        return true;

Wähle a ∈ {2,3, … , n − 1} zufällig
berechne k, d ∈ ℤ mit n − 1 = d2^k und d ungerade
x ← a^d mod n

if x = 1 or x = n − 1 then
        return true;

repeat k − 1 mal
        x ← x^2 mod n
        if x = 1 then
                return false;
        if x = n − 1 then
                return true;
return false;
```

Exkurs – Hashing

Aufgabe von Hash-Verfahren ist es, m Datensätze möglichst gut (und effizient) in n Speicherplätze einzuordnen. Wenn zwei Datensätze demselben Speicherplatz zugeordnet werden, so spricht man von einer Kollision. Da Kollisionen unerwünschte Ereignisse darstellen, möchte man das Verfahren so auslegen, dass Kollisionen nur selten auftreten. Das Geburtstagsproblem beantwortet die Frage, mit welcher Wahrscheinlichkeit keine einzige Kollision auftritt, wenn man die Datensätze zufällig verteilen würde. Die in der Praxis verwendeten Verfahren (die so genannten Hash-Funktionen) garantieren zwar keine «völlige» Gleichverteilung, aber die Abweichungen sind im Allgemeinen sehr gering.



Schema einer Hashfunktion mit Kollision in 02

Eine Hashfunktion f bildet eine (potenziell sehr grosse) Datenmenge auf eine (kleine) natürliche Zahl ab. f soll dabei folgende Eigenschaften erfüllen:

- Alle Hashwerte sollen «gleich oft» vorkommen
- «Geringe» Wahrscheinlichkeit von Kollisionen
- Ähnliche Eingaben sollen zu verschiedenen Ergebnissen führen
- f soll effizient berechenbar sein

Algorithmen – Highlights

Graphenalgorithmen

Lange Pfade

Das Problem: Gegeben sei ein Graph G und eine natürliche, positive Zahl B. Wir wollen feststellen, ob es einen Pfad der Länge B in G gibt. Wir nenne das das «Long-Path» Problem.

Sei G, B gegeben. Zu entscheiden ob es einen Pfad der Länge $\geq B$ gibt gilt als NP-vollständiges Problem.

Mit anderen Worten: Können wir «Long-Path» in polynomieller Zeit lösen so lösen wir das NP-Problem. Wir können also (für die Vorlesung) davon ausgehen, dass dies nicht möglich ist.

Flüsse in Netzwerken

Das Problem: Wir stellen uns ein verzweigtes Netzwerk von Röhren unterschiedlicher Dicke vor, in das man an einer Stelle Flüssigkeit zuführen kann, die an einer anderen Stelle entweichen kann. Uns interessiert wie viel Wasser pro Zeiteinheit von einem gegebenen Startpunkt zu einem gegebenen Endpunkt fliessen kann. Wir modellieren ein solches System wie folgt:

Ein Netzwerk ist ein Tupel $N = (V, A, c, s, t)$, wobei gilt:

- (V, A) ist ein gerichteter Graph
- $s \in V$, die Quelle (source) *[Von hier fliesst das «Wasser» in unser System]*
- $t \in V$, die Senke (sink) *[Hier verlässt das «Wasser» unser System]*
- $c: A \to \mathbb{R}_0^+$, die Kapazitätsfunktion *[Beschränkt, wieviel durch eine Kante fliessen kann]*

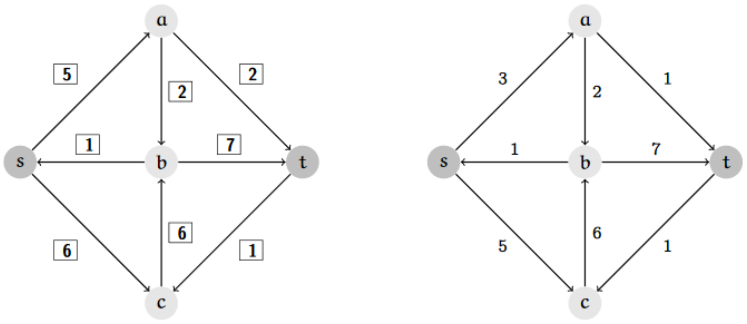
Wichtig ist hierbei dass unser System ja bis auf die Quelle und Senke abgeschlossen ist und damit in einem inneren Knoten weder Fluss entstehen noch verschwinden kann. Diese Eigenschaft lassen wir in die Definition des Flusses in einem Netzwerk eingehen.

Ein Fluss in einem Netzwerk $N = (V, A, c, s, t)$ ist eine Funktion $f: A \to \mathbb{R}$ mit folgenden Bedingungen:

- **Zulässigkeit:** In jeder Kante kann höchstens so viel Fluss herrschen, wie es die maximale Kapazität der Kante erlaubt.
$$\forall_{e \in A}: 0 \leq f(e) \leq c(e)$$
- **Flusserhaltung:** Für jeden Knoten (ausser Quelle und Senke) ist der gesamte Zufluss identisch mit dem gesamten Abfluss.
$$\forall_{v \in V \setminus \{s,t\}}: \sum_{u \in V:(u,v) \in A} f(u,v) = \sum_{u \in V:(v,u) \in A} f(v,u)$$

Wir definieren nun den Fluss eines Netzwerkes $val(f)$ als die Menge an «Wasser» welche von der Quelle in das System fliesst.

$$val(f) \coloneqq netoutflow(s) \coloneqq \left(\sum_{u \in V:(s,u) \in A} f(s,u)\right) - \left(\sum_{u \in V:(u,s) \in A} f(u,s)\right)$$



Ein Netzwerk und ein Fluss mit Wert $3 - 1 + 5 = 7$

Wenn unsere Modellierung vernünftig ist, sollte folglich dieselbe Menge an Flüssigkeit an der Senke ankommen und das System verlassen.

Der Nettozufluss der Senke ist identisch mit dem Nettoabfluss der Quelle, d.h

$$netinflow(t) = netoutflow(s) = \left(\sum_{u \in V:(u,t) \in A} f(u,t)\right) - \left(\sum_{u \in V:(t,u) \in A} f(t,u)\right)$$

Beweis:

$$\begin{aligned}
0 &= \left(\sum_{(v,u) \in A} f(v,u)\right) - \left(\sum_{(u,v) \in A} f(u,v)\right) \\
&= \sum_{v \in V} \left(\left(\sum_{u \in V:(v,u) \in A} f(v,u)\right) - \left(\sum_{u \in V:(u,v) \in A} f(u,v)\right)\right) \\
&= \left(\sum_{u \in V:(s,u) \in A} f(s,u) - \sum_{u \in V:(u,s) \in A} f(u,s)\right) + \left(\sum_{u \in V:(t,u) \in A} f(t,u) - \sum_{u \in V:(u,t) \in A} f(u,t)\right) \\
&= val(f) - netinflow(t) \Rightarrow \boldsymbol{val(f) = netinflow(t)}
\end{aligned}$$

■

Ein Fluss f in einem Netzwerk N ist ein maximaler Fluss genau dann, wenn es im Restnetzwerk N_f einen gerichteten Pfad von der Quelle s zur Senke t gibt.

Wir werden nun zwei Algorithmen beschreiben, welche unser Problem lösen und damit einen maximalen Fluss im Flussnetzwerk finden können.

Ford Fulkerson

Wir suchen mit Hilfe des Restnetzwerks augmentierende Pfade, solange es solche gibt. Dabei erhöhen wir jeweils den Fluss entlang des gefundenen Pfades.

FordFulkerson(V, A, c, s, t)

```
f ← 0                                          //Fluss ist zu Beginn 0
while ∃ s-t-Pfad P in (V, A_f) do              //augmentierender Pfad
        Erhöhe den Fluss f entlang P
return f                                       //maximaler Fluss
```

Sind alle Kapazitäten des gegebenen Netzwerks nichtnegative ganze/rationale Zahlen, so kommt der Algorithmus von Ford und Fulkerson nach endlich vielen Schritten zu Stehen und liefert einen maximalen s-t-Fluss, der ausserdem ganzzahlig/rational ist.

Terminiert der Algorithmus, so haben wir mit Sicherheit einen maximalen Fluss gefunden. Bei irrationalen Zahlen kann es jedoch vorkommen, dass der Algorithmus nicht terminiert

Edmond Karp

Endmond Karp ist eine Weiterentwicklung der Ford Fulkerson Methode, welche eine Laufzeit von $O(EV^2)$ garantiert (selbst für irrationale Zahlen). Wir implementieren Ford Fulkerson wie gehabt, suchen uns aber jeweils als augmentierenden Pfad nicht irgendeinen, sondern den jeweils kürzesten s-t Pfad im Restnetzwerk

EdmondKarp(V, A, c, s, t)

```
f ← 0                                          //Fluss ist zu Beginn 0
while ∃ s-t-Pfad P in (V, A_f) do              //augmentierender Pfad
        Wähle den kürzesten s-t-Pfad mit BFS
        Erhöhe den Fluss f entlang P
return f                                       //maximaler Fluss
```