

# Algorithms and Probability Summary

April 9, 2020

# Chapter 1

## Graphentheorie

### 1.1 Basics and Definitions

**Graph:** A graph is a tuple  $(V, E)$ , where  $V$  is a finite non empty set of vertices and  $E$  is a set of vertex pairs indicating the edges  $V \subseteq E$   
 $(E \subseteq \binom{V}{2} := \{(x, y) | x, y \in V, x \neq y\})$

**Complete (Vollständig):** There is an edge between each pair of vertices (den.  $K_n$ )

**Walk (Weg):** A sequence of vertices  $\langle v_1, v_2, \dots, v_n \rangle$  if  $\forall i$  there exists an edge from  $v_i$  to  $v_{i+1}$ . The length of the walk is given by the number of steps, i.e.  $n-1$ .

**Path (Pfad):** A walk which doesn't contain any vertex more than once (den.  $P_n$ )

**Closed Walk (Zyklus):** A walk in which  $v_1 = v_n$  (den.  $C_n$ )

**Cycle (Kreis):** A closed walk with length of at least three and the vertices  $v_1, \dots, v_{k-1}$  are pairwise distinct (in a directed graph it must have length of at least two)

**Loops (Schlingen):** An edge from a vertex to itself

**Multiple edges (Mehrfachkanten):** When vertex pairs are connected by multiple edges

**Multigraph (Multigraph):** A Graph which contains loops and multiple edges (In this lecture we assume that a graph is not a multigraph unless stated otherwise)

**Neighbourhood (Nachbarschaft):** All outgoing and incoming edges to/from a vertex  $v$  denoted  $N_G(v) := \{u \in V | \{v, u\} \in E\}$

**Degree (Grad):** Indicates the size of the neighbourhood  $deg_G(v) := |N_G(v)|$

**k-regular (k-regulär):** If every vertex  $v \in V$  has degree  $deg(v) = k$   
◦ A complete graph  $K_n$  is  $n-1$ -regular

**Adjacent (Adjazent):** Two vertices  $u$  and  $v$  if there is an edge  $u, v$

**Satz 1.2** For any Graph  $G=(V, E)$  we have  $\sum_{v \in V} deg(v) = 2|E|$

**Korollar 1.3** For any Graph  $G = (V, E)$  the number of vertices with uneven degree is even. (Direct Proof by splitting  $V$  into even and odd degree sets then use S1.2)

**Subgraph (Teilgraph):** A Graph  $H = (V_H, E_H)$  is a subgraph of a graph  $G = (V_G, E_G)$  if  $V_H \subseteq V_G$  and  $E_H \subseteq E_G$  denoted  $H \subseteq G$

**Induced Subgraph (Induzierte Teilgraph):** If  $E_H = E_G \cap \binom{V_H}{2}$  denoted  $H = G[V_H]$ . If there is an edge  $(u, v)$  in  $G$  and  $u, v$  are also vertices in  $H$  then there must be an edge  $(u, v)$  in  $H$  as well

#### 1.1.1 Connectivity and Trees

**Connected (Zusammenhängend):** if for any Vertices  $s, t \in V$  there is a  $s$ - $t$  path. A subgraph  $C \subseteq G$  for which this trait is maximal is called a connected component (hence for all subgraphs  $H \neq C$  with  $C \subseteq H \subseteq G$  is not connected)

**Cycle Free (Kreisfrei):** A Graph which doesn't contain a cycle

**Tree (Baum):** A Graph which is Cycle free and connected

**Leaf (Blatt):**  $T=(V,E)$  a tree and  $v \in V$  a vertex with  $\deg(v) = 1$

**Lemma 1.5:**  $T = (V,E)$  a tree with  $|V| \geq 2$ , it follows:

**a):**  $T$  contains atleast 2 leafs. (Proof: If there was only one leaf  $2|E| = \sum_{v \in V} \deg(v) \geq 1 + 2(|V| - 1)$  which is a contradiction to S1.6)

**b):** if  $v \in V$  is a leaf, the graph  $T-v$  is also a tree.

**Satz 1.6**  $G=(V,E)$  a Graph with  $|V| \geq 1$  vertices, the following is equivalent:

- $G$  is a tree
- $G$  is connected and cycle free
- $G$  is connected and  $|E| = |V| - 1$
- $G$  is cycle free and  $|E| = |V| - 1$
- for any  $x,y \in V$ ,  $G$  contains exactly one  $x$ - $y$  path.

**Forrest (Wald):**  $W = (V,E)$  a graph which is cycle free, every component of a forrest is a tree

**Lemma 1.7** A forrest  $G = (V,E)$  contains  $|V| - |E|$  connected components (Proof by induction)

**Directed Graph (Gerichteter Graph):** A graph where the edges are represented by ordered pairs, i.e The directed graph  $D$  is given by the tuple  $(V,A)$  where  $V$  is the set of vertices and  $A \subseteq V \times V$  a set of directed edges. Compared to an undirected graph, between two vertices there can be two edges  $(x,y)$  and  $(y,x)$

**Out-Degree(Aus-Grad):**  $\deg^+(v) := |\{(x,y) \in A | x = v\}|$

**In-Degree(In-Grad):**  $\deg^-(v) := |\{(x,y) \in A | y = v\}|$

**Satz 1.8** For any directed graph  $D = (V,A)$  the following is true.  
 $\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |A|$ .

**Acyclic (Azyklisch):** A directed graph which doesn't contain a cycle (DAG). DAG's have a topological ordering.

**Satz 1.9** For any DAG  $D=(V,A)$  we can find a topological ordering in  $\mathcal{O}(|V| + |A|)$

**Strongly Connected (start zusammenhängend):** For a DAG  $D=(V,A)$ , if for every pair of vertices  $u,v \in V$  a directed  $u$ - $v$ -Path exists

**Weakly Connected (schwach zusammenhängend):** When the underlying graph(i.e ignoring the direction of the edges) is connected

### 1.1.2 Datastructures

The two main ways of storing graphs, is with Adjacency matrices and Adjacency lists.

## 1.2 Trees

## 1.3 Paths

**Shortest Paths** Given: A connected Graph  $G = (V,E)$ , two vertices  $s,t \in V$  and a costfunction  $c : E \rightarrow \mathbb{R}$

Goal: Find an  $s$ - $t$ -path  $P$  in  $G$  with  $\sum_{e \in P} c(e) = \min$

Algorithms which can solve shortest path problems:

1. Dijkstras Algorithm
2. Floyd-Warshall
3. Bellman-Ford
4. Johnson's Algorithm

## 1.4 Connection

**Definition 1.23** A Graph  $G = (V,E)$  is **k-connected (k-zusammenhängend)** if  $|V| \geq k+1$  and for all subsets  $X \subseteq V$  with  $|X| < k$  the following is true: The Graph  $G[V \setminus X]$  is connected. (Hence you would need to remove atleast  $k$  vertices to destroy the connectivity of the graph, the only exception is a complete graph which is by definition  $k-1$  connected)

**Definition 1.24** A Graph  $G = (V, E)$  is **k-edge-connected (k-kanten-zusammenhängend)**, if for all subsets  $X \subseteq E$  with  $|X| < k$  the following is true:  $(V, E \setminus X)$  is connected. (Hence at least  $k$  edges must be removed to destroy the connectivity of the Graph)

**Satz 1.25 Menger**  $G = (V, E)$  the following applies:

1.  $G$  is  $k$ -connected iff for all pairs of vertices  $u, v \in V$ ,  $u \neq v$ , at least  $k$  internal-vertex disjoint  $u$ - $v$  paths exist
2.  $G$  is  $k$ -edge-connected iff for all pairs of vertices  $u, v \in V$ ,  $u \neq v$ , at least  $k$  edge-disjoint  $u$ - $v$ -paths exist

### 1.4.1 Articulation vertex (Artikulationsknoten)

**Articulation vertex (Artikulationsknoten):** If a graph is connected but not 2-connected, then there exists a vertex  $v$  with the attribute that  $G[V \setminus \{v\}]$  is not connected. Articulation vertices can be detected using a modified DFS.

**Forward Edge (Vorwärtskante):** An edge starting from a vertex with a lower dfs number than the destination vertex

**Backwards Edge (Rückwärtskante):** An edge starting from a vertex with a higher dfs number than the destination vertex

we assign  $\in V$  a number  $\text{low}[v] :=$  the smallest dfs-Number, that can be reached from the vertex  $v$  using any number of forward edges and at most one backward edge. It follows for all  $v \in V$ :  $\text{low}[v] \leq \text{dfs}[v]$

$v$  is an Articulation vertex  $\Leftrightarrow v = s$  and  $s$  has degree of at least 2 or  $v \neq s$  and there exists a  $w \in V$  with  $\{v, w\} \in E(T)$  and  $\text{low}[w] \geq \text{dfs}[v]$

**TODO:** Implement DFS-Visit which finds the articulation vertices for a given graph

**Satz 1.27** For a connected graph  $G = (V, E)$ , implemented with an adjacency list Articulation vertices can be found in  $\mathcal{O}(|E|)$

### 1.4.2 Bridges (Brücken)

**Bridge (Brücke):** An edge  $e \in E$  such that  $(V, E \setminus \{e\})$  is not connected

From the definition of the bridge it follows that a spanning tree must contain all bridges of a graph and that the vertices at the end of the bridge are either Articulation vertices or vertices with degree 1.

An edge  $(v, w)$  of the depth-first-search tree is a bridge iff  $\text{low}[w] > \text{dfs}[v]$

**Satz 1.28** For a connected graph  $G = (V, E)$  implemented with an adjacency list, articulation vertices and bridges can be found in  $\mathcal{O}(|E|)$

## 1.5 Cycles

### 1.5.1 Eulerwalk (Eulertour)

**Definition 1.29** A Eulerwalk in a graph  $G = (V, E)$  is a cycle which contains each edge exactly once. If  $G$  contains a Eulerwalk then  $\deg(v)$  of all  $v \in V$  is even. (Proof by contradiction assume  $G$  has vertices of even degrees pick a starting node  $v$  and an arbitrary node  $u$  and show that the path cannot end in  $u$  arguing with the parity of the degree)

In a connected graph eulerian graph a Eulerwalk can be found in  $\mathcal{O}(|E|)$

**TODO:** Implement an Algorithm which can find a Eulerwalk in  $\mathcal{O}(|E|)$

### 1.5.2 Hamilton Cycles (Hamiltonkreise)

**Definition 1.31** A Hamilton Cycle in a graph  $G = (V, E)$  is a cycle in which all vertices of  $V$  are visited exactly once. If a graph contains a Hamilton Cycle it is called hamiltonian (hamiltonisch). Whether or not a graph contains a hamilton cycle is NP-complete.

**Satz 1.33** The Algorithm HAMILTONKREIS to find a Hamilton cycle of a given graph  $G$  needs  $\mathcal{O}(n * 2^n)$  memory and has a runtime of  $\mathcal{O}(n^2 * 2^n)$ , where  $n = |V|$

**TODO:** Implement the Algorithm HAMILTONKREIS

### 1.5.3 Special Cases

**Lattice (Gittergraph)** An  $m \times n$  lattice is hamiltonian if  $m$  or  $n$  is even (Proof using parity argument)

**Lemma 1.35**  $G = (A \uplus B, E)$  a bipartite Graph with  $|A| \neq |B|$ , then  $G$  cannot contain a Hamilton cycle

**Hypercube (Hyperwürfel):** The set of edges of a Hypercube  $H_d$  is  $\{0, 1\}^d$  hence the set of all 0-1 sequences of length  $d$ . Two vertices are connected if their sequences differ at exactly one spot. A  $d$ -dimensional Hypercube contains a Hamilton cycle for all  $d \geq 2$  (Proof by induction)

**Satz 1.37(Dirac)** If  $G = (V, E)$  is a graph with  $|V| \geq 3$  vertices and each vertex has at least  $|V|/2$  neighbours, then  $G$  is hamiltonian.

### 1.5.4 The Travelling Salesman Problem

**Given:** A complete Graph  $K_n$  and a function  $l : \binom{[n]}{2} \rightarrow \mathbb{N}$  which gives each edge a length

**Goal:** Find a Hamilton cycle  $C$  in  $K_n$  with  $\sum_{e \in C} l(e) = \min\{\sum_{e \in C'} l(e) \mid C' \text{ is a Hamilton cycle in } K_n\}$

For a graph  $G = (V, E)$  with  $V = [n]$  vertices we define a weight function  $l$  as  $l(\{u, v\}) = \begin{cases} 0, & \text{if } \{u, v\} \in E \\ 1, & \text{else} \end{cases}$

hence the length of a minimal Hamilton cycle in  $K_n$  when using  $l$  is 0 iff  $G$  contains a Hamilton cycle, hence this allows us evaluate the efficiency of a solution. We define the optimum solution as  $opt(K_n, l) := \min\{\sum_{e \in C'} l(e) \mid C' \text{ is a Hamilton cycle in } K_n\}$ . An algorithm which always finds a Hamilton cycle with  $\sum_{e \in C} l(e) \leq \alpha * opt(K_n, l)$  is known as an  $\alpha$ -Approximation algorithm.

**Satz 1.39** If there exists an  $\alpha$ -Approximation algorithm for an  $\alpha > 1$  for the Traveling Salesman Problem with a runtime of  $\mathcal{O}(f(n))$ , then an algorithm exists for all graphs with  $n$  vertices which decides whether it is hamiltonian or not in the same time complexity

#### Metric Travelling Salesman Problem (Metrisches TSP)

**Given:** A complete Graph  $K_n$  and a function  $l : \binom{[n]}{2} \rightarrow \mathbb{N}$  with the condition  $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$  for all  $x, y, z \in [n]$

**Goal:** Find a Hamilton cycle  $C$  in  $K_n$  with  $\sum_{e \in C} l(e) = \min\{\sum_{e \in C'} l(e) \mid C' \text{ is a Hamilton cycle in } K_n\}$  The condition is called the triangle inequality (Dreiecksungleichung) and states that the direct connection between two vertices  $x$  and  $z$  cannot be longer than the detour over the vertex  $y$ .

**Satz 1.40** The Metric Travelling Salesman Problem has a 2-Approximation algorithm with a runtime of  $\mathcal{O}(n^2)$

## 1.6 Matchings

**Matching** A set of edges  $M \subseteq E$  is called a Matching of a graph  $G = (V, E)$ , if no vertex of the graph is incident to more than one edge from  $M$  i.e.  $e \cap f = \emptyset$  for all  $e, f \in M$  with  $e \neq f$ . A vertex  $v$  is "covered" (überdeckt) if there is an edge  $e \in M$  which contains  $v$ .

**Perfect Matching** If each vertex is covered by exactly one edge of the Matching  $M$  i.e.  $|M| = |V|/2$ . Not all graphs contain a perfect matching for example a star graph.

**Maximal matching (Inklusionsmaximal):**  $G = (V, E)$ , for a Matching  $M$  if  $M \cup \{e\}$  is not a Matching for all edges  $e \in E \setminus M$ .

**Maximum matching (Kardinalitätsmaximal):**  $G = (V, E)$  for a Matching  $M$  if  $|M| \geq |M'|$  for all Matchings  $M'$  in  $G$

**Example:** A path consisting of three edges. Creating a Matching using the middle edge would create a Maximal matching but not a Maximum matching. A Maximum and a Maximal Matching can be created by taking the two outer edges

### 1.6.1 Algorithms

**GREEDY MATCHING** This algorithm picks random edges from  $E$  and adds it to the matching at the same time it deletes all incident edges from  $E$ . The algorithm stops when  $E = \emptyset$ . This algorithm can find a Maximal matching in time  $\mathcal{O}(|E|)$  for which the following applies:  $|M_{Greedy}| \geq \frac{1}{2} |M_{max}|$  where  $M_{max}$  is a Maximum matching

**Union of two Matchings:** Let  $M_1$  and  $M_2$  be arbitrary Matchings.  $G_M = (V; M_1 \cup M_2)$ . Every vertex in  $G_M$  has degree at most 2, hence all components of the graph are paths and/or cycles (cycles having even length). If we assume  $|M_1| < |M_2|$ , then every cycle/path of even length will have the same amount of edges from  $M_1$  as  $M_2$ . From our assumption we know that there must be a path  $P$  which contains more edges from  $M_2$  than  $M_1$  the two outer edges belonging to  $M_2$ . Hence we can create a new Matching  $M'_1$  using  $P$  which will contain one more edge. We achieve this by switching the edges in  $P$  i.e.  $M'_1 := (M_1 \cup (P \cap M_2)) \setminus (P \cap M_1)$ . We say  $P$  is a  $M_1$ -augmented path.

**M-augmented Path (augmentierender Pfad):** Let  $M$  be an arbitrary Matching, an  $M$  augmented path is a path where the last two edges of  $P$  are not covered by  $M$  and  $P$  consists of edges alternating between edges belonging to  $M$  and not belonging to  $M$

**AUGMENTED MATCHING** This algorithm finds a Maximum matching. We start by finding a Matching which consists of only one arbitrary edge. As long as the matching is not a maximum matching we repeat the following: We take an augmented path and we increase the size of the Matching. We know that after  $|V|/2 - 1$  times the matching will be maximum because a matching can't have more than  $|V|/2$  edges. We can easily find augmented paths in a bipartite Graph using a modified BFS. The total runtime of our algorithm is  $\mathcal{O}(|V| \cdot |E|)$

**Satz 1.45** If  $n$  is even and  $l : \binom{[n]}{2} \rightarrow \mathbb{N}$  a weight function of the complete graph  $K_n$  then we can find a minimal perfect Matching (i.e a matching where the sum of edge weights are minimal) in  $\mathcal{O}(n^3)$

**Satz 1.46** From S1.45 it follows that for the Metric Travelling Salesman Problem there is a 3/2-Approximation algorithm with a runtime of  $\mathcal{O}(n^3)$

## 1.6.2 Der Satz von Hall

**Bipartite** A graph  $G=(V,E)$  is bipartite if we can partition the set of vertices  $V$  into two sets  $A$  and  $B$  such that all edges in  $E$  contain a vertex from  $A$  and a vertex from  $B$  (denoted:  $G = (A \uplus B, E)$ )

**Satz von Hall/Heiratssatz:** For a bipartite graph  $G = (A \uplus B, E)$  there is a Matching  $M$  of cardinality  $|M| = |A|$  iff  $|N(X)| \geq |X|$  for all  $X \subseteq A$ . From the Satz von Hall it follows that a  $k$ -regular graph always has a perfect matching.

**Satz 1.48** Let  $G = (A \uplus B, E)$  be a  $k$ -regular bipartite graph. There exists an  $M_1, \dots, M_k$  such that  $E = M_1 \uplus \dots \uplus M_k$  and all  $M_i, 1 \leq i \leq k$  are perfect matchings in  $G$ . The perfect matching can be found in  $\mathcal{O}(|E|)$ .

**Satz 1.49** Let  $G=(V,E)$  a  $2^k$ -regular bipartite graph. We can find a perfect matching in  $\mathcal{O}(|E|)$

## 1.7 Colouring (Färbungen)

**Vertex colouring (Knoten Färbung):** The vertex colouring of a graph  $G = (V,E)$  with  $k$  colours is a mapping  $c : V \rightarrow [k]$  such that the following holds:  $c(u) \neq c(v)$  for all edges  $u,v \in E$

**Chromatic number (chromatische Zahl):** denoted  $\chi(G)$  is the minimal number of colours needed to color the vertices of  $G$ . A complete graph has the chromatic number  $n$ . Cycles of even length have chromatic number 2, uneven length have a chromatic number 3. Trees with atleast two vertices have a chromatic number 2. Graphs with chromatic number  $k$  are also called  $k$ -partite. To decide whether or not a graph  $G$  is bipartite can be done in  $\mathcal{O}(|E|)$  with a DFS or BFS.

**Satz 1.53** A graph  $G=(V,E)$  is bipartite iff it does not contain a cycle of odd length as a subgraph

**Satz 1.54 (Vierfarbensatz):** Any map can be coloured using 4 colours.

**GREEDY FÄRBUNG** This algorithm calculates the colouring of a graph by picking vertices at random and giving it the lowest colour not used by its neighbours. There exists a order of vertices for which the GREEDY algorithm needs  $\chi(G)$  colors.

**Satz 1.55** Let  $G$  be a connected graph. For the number  $C(G)$  of colours needed by GREEDY FÄRBUNG to color the graph  $G$  the following applies:  $\chi(G) \leq C(G) \leq \Delta(G) + 1$  ( $\Delta(G) := \max_{v \in V} \deg(v)$ , hence the max degree of a vertex in  $G$ ). If the graph is saved in an adjacency list then we can find the colouring in  $\mathcal{O}(|E|)$

**Satz 1.59 (Satz von Brooks)** Let  $G=(V,E)$  be a connected graph which is not complete nor a cycle with odd degree i.e  $G \neq K_n$  and  $G \neq C_{2n+1}$  then the following holds  $\chi(G) \leq \Delta(G)$  and there exists an Algorithm which can colour the graph in  $\mathcal{O}(|E|)$  with  $\Delta(G)$  colours.

**Satz 1.60** Let  $G=(V,E)$  be a graph and  $k \in \mathbb{N}$  a natural number such that every induced subgraph of  $G$  has a vertex with degree at most  $k$ . It follows that  $\chi(G) \leq k + 1$  and we can find a  $(k+1)$ -colouring in  $\mathcal{O}(|E|)$

**Satz 1.61 (Mycielski-Konstruktion):** For all  $k \geq 2$  there is a triangle free graph  $G_k$  with  $\chi(G_k) \geq k$  (Proof by induction)

**Satz 1.62** Every 3-colourable graph  $G=(V,E)$  can be coloured in  $\mathcal{O}(|E|)$  with  $\mathcal{O}(\sqrt{|V|})$  colours. Given a graph  $G=(V,E)$ , is  $\chi(G) \leq 3$  is NP-Complete.

## Chapter 2

# Probability Theory and Randomised Algorithms

### 2.1 Definitions and Notations

**Definition 2.1** A discrete Probabilitiespace(diskreter Wahrscheinlichkeitsraum) is defined by a Set of **outcomes** (Ergebnismenge) denoted  $\Omega = \{\omega_1, \omega_2, \dots\}$  of **elementary outcomes**(Elementarereignissen). Each elementary outcome  $\omega_i$  is assigned a **(elementary)probability**(Elementar – Wahrscheinlichkeit) denoted  $Pr[\omega_i]$  where  $0 \leq Pr[\omega_i] \leq 1$  and  $\sum_{\omega \in \Omega} Pr[\omega] = 1$ . A set  $E \subseteq \Omega$  is called an **outcome**(Ereignis) The probability of  $Pr[E]$  of an outcome is defined by:  $Pr[E] := \sum_{\omega \in E} Pr[\omega]$

If  $E$  is an outcome, we define  $\bar{E} := \Omega \setminus E$  the **compliment outcome**(Komplementarereignis)

**Finite probabilitiespace (endlicher Wahrscheinlichkeitsraum):** A Probability space  $\Omega = \{\omega_1, \dots, \omega_n\}$  (Assumption for infinite probability spaces  $\Omega = \mathbb{N}_0$ )

**Lemma 2.2** For outcomes A,B the following applies:

1.  $Pr[\emptyset] = 0, Pr[\Omega] = 1$
2.  $0 \leq Pr[A] \leq 1$
3.  $Pr[\bar{A}] = 1 - Pr[A]$
4. if  $A \subseteq B, Pr[A] \leq Pr[B]$
5. (Additionssatz) Wenn die Ereignisse  $A_1, \dots, A_n$  paarweise disjunkt sind (also wenn für alle Paare  $i \neq j$  gilt, dass  $A_i \cap A_j = \emptyset$ ) so folgt:

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

**Union Bound (Boolesche Ungleichung):** Für beliebige Ereignisse  $A_1, \dots, A_n$  gilt:

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

**Siebformel, Prinzip Inklusion/Exklusion:** Für Ereignisse  $A_1, \dots, A_n (n \geq 2)$  gilt:

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

**Laplace Raum:** endlicher Wahrscheinlichkeitsraum, in dem alle Elementarereignisse gleich wahrscheinlich sind. In einem Laplace-Raum gilt für jedes Ereignis E:

$$Pr[E] = \frac{|E|}{|\Omega|}$$

**# Möglichkeiten k Elemente aus einer n-elementigen Menge zu ziehen** :  $(n^k := n(n-1)(n-2) \dots (n-k-1) = \frac{n!}{(n-k)!})$

**Bedingte Wahrscheinlichkeit:** A und B seien Ereignisse mit  $Pr[B] > 0$ . Die bedingte Wahrscheinlichkeit  $Pr[A|B]$  (Die W'keit, dass Ereignis A eintritt, wenn wir schon wissen dass Ereignis B eingetreten ist) von A gegeben B ist t definiert durch:

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

	geordnet	ungeordnet
mit Zurücklegen	$n^k$	$\binom{n+k-1}{k}$
ohne Zurücklegen	$n^{\underline{k}}$	$\binom{n}{k}$

(a)

Beispiel:  $k=2$  Elemente aus  $S=\{1,2,3\}$  ziehen ( $n=3$ )

	geordnet	ungeordnet
mit Zurücklegen	$(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)$	$\{1,1\}, \{1,2\}, \{1,3\}, \{2,2\}, \{2,3\}, \{3,3\}$
ohne Zurücklegen	$(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)$	$\{1,2\}, \{1,3\}, \{2,3\}$

(b)

**Satz 2.12 Satz der totalen W'keit:** Die Ereignisse  $A_1, \dots, A_n$  seien paarweise disjunkt und es gelte  $B \subseteq A_1 \cup \dots \cup A_n$  dann folgt:

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

**Satz 2.10 Multiplikationsatz:** Seien die Ereignisse  $A_1, \dots, A_n$  gegeben. Falls  $Pr[A_1 \cap \dots \cap A_n] > 0$  ist, gilt:

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

**Satz von Bayes:** Die Ereigniss  $A_1, \dots, A_n$  seien paarweise disjunkt. Ferner  $B \subseteq A_1 \cup \dots \cup A_n$  ein Ereignis mit  $Pr[B] > 0$  Dann gilt für ein beliebiges  $i = 1, \dots, n$ :

$$Pr[A_i|B] = \frac{Pr[A_i \cap B]}{Pr[B]} = \frac{Pr[B|A_i] \cdot Pr[A_i]}{\sum_{j=1}^n Pr[B|A_j] \cdot Pr[A_j]}$$

**Unabhängigkeit Zwei Ereignisse:** Die Ereignisse A und B heissen unabhängig, wenn gilt:

$$Pr[A \cap B] = Pr[A] \cdot Pr[B]$$

**Unabhängigkeit:** Die Ereignisse  $A_1, \dots, A_n$  heissen unabhängig, wenn für alle Teilmengen  $I \subseteq \{1, \dots, n\}$  mit  $I = \{i_1, \dots, i_k\}$  gilt, dass

$$Pr[A_{i_1} \cap \dots \cap A_{i_k}] = Pr[A_{i_1}] \dots Pr[A_{i_k}] \quad (2.2)$$

Eine unendliche Familie von Ereignissen  $A_i$  mit  $i \in \mathbb{N}$  heisst unabhängig, wenn (2.2) für jede endliche Teilmenge  $I \subseteq \mathbb{N}$  erfüllt ist. Dies ist offensichtlich erfüllt, wenn die Ereignisse physikalisch unabhängig sind (e.g wenn jedes  $A_i$  einem unabhängigen Münzwurf entspricht) aber ist nicht unbedingt erforderlich.

Die Ereignisse  $A_1, \dots, A_n$  sind genau dann unabhängig wenn für alle  $(s_1, \dots, s_n) \in \{0, 1\}^n$  gilt dass

$$Pr[A_1^{s_1} \cap \dots \cap A_n^{s_n}] = Pr[A_1^{s_1}] \dots Pr[A_n^{s_n}]$$

wobei  $A_i^0 = \overline{A_i}$  und  $A_i^1 = A_i$

Seien A, B und C unabhängige Ereignisee. Dann sind auch  $A \cap B$  und C bzw.  $A \cup B$  und C unabhängig.

## 2.2 Zufallsvariablen

Eine funktion welche jede element unser Wahrscheinlichkeitsraum ein Reellezahl zuordnet.

$$X : \Omega \rightarrow \mathbb{R}$$

Beispiel:

" $X \leq 5$  steht für das Ereignis, dass die Zufallsvariable einen Wert kleiner gleich 5 annimmt also:

$$\Rightarrow X \leq 5 \triangleq \{\omega \in \Omega : X(\omega) \leq 5\}$$

**Dichtefunktion:**

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

**Verteilungsfunktion:**

$$F_X : \mathbb{R} \rightarrow [0, 1], \quad x \mapsto Pr[X \leq x] = \sum_{x' \in W_X : x' \leq x} Pr[X = x']$$

Beispiele:



Wir werfen eine Münze drei Mal:

$X := \text{Anzahl „Kopf“}$

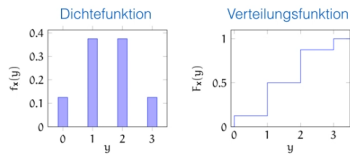
Wahrscheinlichkeitsraum:  $\Omega = \{KKK, KKZ, \dots, ZZZ\}$

$$\text{Prob}[X=0] = \text{Pr}[\{ZZZ\}] = 1/8$$

$$\text{Prob}[X=1] = \text{Pr}[\{ZZK, ZKZ, KZZ\}] = 3/8$$

$$\text{Prob}[X=2] = \text{Pr}[\{ZKK, KKK, KZK\}] = 3/8$$

$$\text{Prob}[X=3] = \text{Pr}[\{KKK\}] = 1/8$$



(a)

Wir werfen einen Würfel drei Mal:

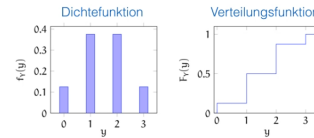
$Y := \text{Anzahl ungerader Augenzahlen}$

Wahrscheinlichkeitsraum:  $\Omega = \{1,2,3,4,5,6\}^3$

$$\text{Prob}[Y=0] = \text{Pr}[\text{erster Wurf gerade}] \cdot \text{Pr}[\text{zweiter Wurf gerade}] \cdot \text{Pr}[\text{dritter Wurf gerade}] = 1/8$$

$$\begin{aligned} \text{Prob}[Y=1] = & \text{Pr}[\text{erster Wurf gerade}] \cdot \text{Pr}[\text{zweiter Wurf gerade}] \cdot \text{Pr}[\text{dritter Wurf ungerade}] + \\ & \text{Pr}[\text{erster Wurf gerade}] \cdot \text{Pr}[\text{zweiter Wurf ungerade}] \cdot \text{Pr}[\text{dritter Wurf gerade}] + \\ & \text{Pr}[\text{erster Wurf ungerade}] \cdot \text{Pr}[\text{zweiter Wurf gerade}] \cdot \text{Pr}[\text{dritter Wurf gerade}] = 3/8 \end{aligned}$$

$$\text{Prob}[Y=2] = 3/8, \quad \text{Prob}[Y=3] = 1/8$$



(b)

### Bernoulli-Verteilung:

$$X \sim \text{Bernoulli}(p)$$

$$f_X(x) = \begin{cases} p & \text{für } x = 1, \\ 1 - p & \text{für } x = 0 \\ 0 & \text{sonst} \end{cases}$$

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

### Binomial Verteilung:

$$X \sim \text{Bin}(n, p)$$

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

$$\mathbb{E}[X] = np \quad \text{Var}[X] = np(1-p) \quad (\text{gleichung für Varianz gilt nur wenn die } X_i \text{'s unabhängig sind})$$

$\text{Bin}(n, \frac{\lambda}{n})$  konvergiert für  $n \rightarrow \infty$  gegen  $\text{Po}(\lambda)$  Beispiel: Werfen einer Münze n mal,  $X = \text{Anzahl Kopf}$

### Negative Binomialverteilung:

$$X \sim \text{NegativeBinomial}(n)$$

$$f_X(k) = \begin{cases} \binom{k-1}{n-1} (1-p)^{k-n} p^n, & \text{für } k = 1, 2, \dots \\ 0, & \text{sonst} \end{cases}$$

$$\mathbb{E}[X] = \frac{n}{p}$$

Beispiel: Warten auf den n-ten Erfolg

### Geometrische Verteilung:

$$X \sim \text{Geo}(p)$$

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

$$F_X(n) = 1 - (1-p)^n \quad \text{für alle } n=1,2,\dots$$

$$\mathbb{E}[X] = \frac{1}{p} \quad \text{Var}[x] = \frac{1-p}{p^2}$$

Beispiel: Wiederholtes Werfen einer Münze,  $X = \# \text{ Würfe bis zum ersten Mal Kopf}$

Gedächtnislosigkeit: Ist  $X \sim \text{Geo}(p)$ , so gilt für alle  $s, t \in \mathbb{N}$ :

$$\text{Pr}[X \geq s+t | X > s] = \text{Pr}[X \geq t]$$

Beispiel: Wahrscheinlichkeit im ersten Wurf Kopf zu bekommen ist identisch zur Wahrscheinlichkeit nach 1000 Fehlversuchen im 1001ten Wurf Kopf zu bekommen.

### Poisson-Verteilung:

$$X \sim \text{Po}(\lambda)$$

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

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

Beispiel: Modellierung seltener Ereignisse e.g.  $X := \# \text{ Herzinfarkte in der Schweiz in der nächsten Stunde}$

**Erwartungswert:** Den Zufallsvariablen  $X$  definieren wir den Erwartungswert  $\mathbb{E}[X]$  durch:

$$\mathbb{E}[X] := \sum_{x \in W_X} x \cdot Pr[X = x]$$

sofern die Summe konvergiert. Ansonsten sagen wir, dass der Erwartungswert undefiniert ist. Ist  $X$  eine Zufallsvariable, so gilt:

$$\mathbb{E}[X] = \sum_{\omega \in \Omega} X(\omega) \cdot Pr[\omega]$$

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

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

Beobachtung: Für ein Ereignis  $A \subseteq \Omega$  ist die zugehörige Indikatorvariable  $X_A$  definiert durch:

$$X_A(\omega) = \begin{cases} 1 & \text{falls } \omega \in A \\ 0 & \text{sonst.} \end{cases}$$

Für den Erwartungswert von  $X_A$  gilt:  $\mathbb{E}[X_A] = Pr[A]$

- Schnitt:  $A \cap B \quad X_{A \cap B} = X_A \cdot X_B$
- Komplement:  $\bar{A} := \Omega \setminus A \quad X_{\bar{A}} = 1 - X_A$
- Vereinigung:  $A \cup B \quad X_{A \cup B}$

Beispiel:

Beispiel: Wir werfen eine Münze 100 Mal  $X := \text{Anzahl Kopf}$

Setze für alle  $i = 1, \dots, 100$ :

$X_i := \text{Indikatorvariable für „Kopf“ im } i\text{ten Wurf}$

Dann  $X = X_1 + \dots + X_{100}$

und  $\mathbb{E}[X_i] = 1/2$  und wegen der Linearität des Erwartungswertes

daher  $\mathbb{E}[X] = \mathbb{E}[X_1] + \dots + \mathbb{E}[X_{100}] = 50$

Figure 2.3

Linearität des Erwartungswerts: Für Zufallsvariablen  $X_1, \dots, X_n$  und  $X = a_1 X_1 + \dots + a_n X_n + b$  mit  $a_1, \dots, a_n, b \in \mathbb{R}$  gilt:

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

Stabiler Menge: Knoten, die nicht durch Kanten verbunden sind.

**Satz** Für jeden Graphen  $G=(V,E)$  mit  $|V| = n$  und  $|E| = m$  bestimmt der Algorithmus (gehe durch die Knotenmenge und entferne den Knoten und inzidente Kanten mit wahrscheinlichkeit  $1-p$ , bei den übrig gebliebene kanten lösche ein Knoten) eine stabile Menge  $S$  mit

$$\mathbb{E}[S] \geq np - mp^2$$

Beweis:

$X :=$  Anzahl Knoten, die erste Runde „überleben“

$\Rightarrow$  jeder einzelne Knoten überlebt mit Wahrscheinlichkeit  $p$ ,

wir haben  $n$  Knoten

$\Rightarrow$  (Linearität des Erwartungswertes)  $E[X] = np$

$Y :=$  Anzahl Kanten, die erste Runde „überleben“

$\Rightarrow$  jede einzelne Kante überlebt mit Wahrscheinlichkeit  $p^2$ ,

wir haben  $m$  Kanten

$\Rightarrow$  (Linearität des Erwartungswertes)  $E[Y] = mp^2$

$S \geq X - Y$  da wir höchstens einen Knoten pro Kante löschen

$\Rightarrow$  (Linearität des Erwartungswertes)  $E[S] \geq E[X] - E[Y]$

Figure 2.4

### Coupon Collector:

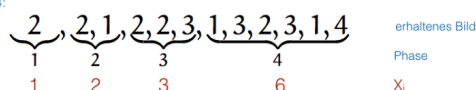
Szenario: Es gibt  $n$  verschiedene Bilder in jeder Runde erhalten wir (gleichwahrscheinlich) eines der Bilder  
 $X :=$  Anzahl Runden bis wir alle  $n$  Bilder besitzen Ziel: Berechne  $E[X]$

Lösungsansatz: betrachte  $n$  Phasen

Phase  $i$ : Runden während wir  $i-1$  verschiedene Bilder besitzen

$X_i :=$  Anzahl Runden in Phase  $i$

Beispiel  $n=4$ :



(a)

$X_i :=$  Anzahl Runden in Phase  $i$ ,  $X_i \sim \text{Geo}((n-(i-1))/n)$

$$E[X] = \sum_{i=1}^n E[X_i] = \sum_{i=1}^n \frac{n}{n-i+1} = n \cdot \sum_{i=1}^n \frac{1}{i} = n \cdot H_n,$$

$$H_n = \ln n + O(1)$$

(b)

$\Rightarrow$  die Laufzeit der Coupon Collector ist  $O(n \log n + n)$

**Varianz:** Für ein Zufallsvariable  $X$  mit  $\mu = E[X]$  definieren wir die Varianz  $Var[X]$  durch:

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

$$Var[X] = E[X^2] - E[X]^2$$

$$Var[a \cdot X + b] = a^2 \cdot Var[X] \quad X \text{ beliebig, } a, b \in \mathbb{R}$$

(b verschwindet, da verschieben der werte kein einfluss auf die Abweichung vom Durchschnitt hat)

### Standardabweichung:

$$\sigma := \sqrt{Var[X]}$$

**Dichten:**  $X, Y$  Zufallsvariablen:

Gemeinsame Dichte:

$$f_{X,Y}(x, y) := Pr[X = x, Y = y]$$

Randdichte:

$$f_X(x) = \sum_{y \in W_Y} f_{X,Y}(x, y)$$

**Unabhängigkeit:** Zufallsvariablen  $X_1, \dots, X_n$  heißen unabhängig genau dann, wenn für alle  $(x_1, \dots, x_n) \in W_{X_1} \times \dots \times W_{X_n}$  gilt:

$$Pr[X_1 = x_1, \dots, X_n = x_n] = Pr[X_1 = x_1] \cdot \dots \cdot Pr[X_n = x_n]$$

Alternativ:

$$f_{X_1, \dots, X_n}(x_1, \dots, x_n) = f_{X_1}(x_1) \cdot \dots \cdot f_{X_n}(x_n) \text{ für alle } (x_1, \dots, x_n) \in W_{X_1} \times \dots \times W_{X_n}$$

Beispiel:

$$\Omega = \{1, 2, 3, 6\} \text{ mit } Pr[\omega] = 1/4 \text{ für alle } \omega \in \Omega$$

$$\begin{aligned} X(\omega) &= \begin{cases} 1 & \text{wenn } \omega \text{ durch } 2 \text{ teilbar} \\ 0 & \text{sonst} \end{cases} & f_X(i) &= \begin{cases} 1/2 & \text{für } i=0,1 \\ 0 & \text{sonst} \end{cases} \\ Y(\omega) &= \begin{cases} 1 & \text{wenn } \omega \text{ durch } 3 \text{ teilbar} \\ 0 & \text{sonst} \end{cases} & f_Y(i) &= \begin{cases} 1/2 & \text{für } i=0,1 \\ 0 & \text{sonst} \end{cases} \\ f_{X,Y}(i,j) &= \begin{cases} 1/4 & \text{für alle } (i,j) \in \{0,1\} \times \{0,1\} \\ 0 & \text{sonst} \end{cases} & \Rightarrow & \boxed{\text{X und Y sind unabhängig}} \end{aligned}$$

Figure 2.6

Für zwei Indikatorvariablen X und Y gilt:

$$X \text{ und } Y \text{ sind unabhängig} \iff f_{X,Y}(1,1) = f_X(1) \cdot f_Y(1)$$

**Lemma:** Sind  $X_1, \dots, X_n$  unabhängige Zufallsvariablen und  $S_1, \dots, S_n$  beliebige Mengen mit  $S_i \subseteq W_{X_i}$ , dann gilt:

$$Pr[X_1 \in S_1, \dots, X_n \in S_n] = Pr[X_1 \in S_1] \cdot \dots \cdot Pr[X_n \in S_n]$$

**Korollar:** Sind  $X_1, \dots, X_n$  unabhängige Zufallsvariablen und ist  $I = \{i_1, \dots, i_k\} \subseteq [n]$ , dann sind  $X_{i_1}, \dots, X_{i_k}$  ebenfalls unabhängig.

**Satz:**  $f_1, \dots, f_n$  seien reellwertige Funktionen ( $f_i: \mathbb{R} \rightarrow \mathbb{R}$  für  $i = 1, \dots, n$ ). Wenn die Zufallsvariablen  $X_1, \dots, X_n$  unabhängig sind dann gilt dies auch für  $f_1(X_1), \dots, f_n(X_n)$

**Summe von Zufallsvariablen:** Für zwei unabhängige Zufallsvariablen X und Y sei  $Z := X + Y$ . Es gilt:

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

Es folgt:

$$Poisson(\lambda_1) + Poisson(\lambda_2) = Poisson(\lambda_1 + \lambda_2) \quad Bin(n_1, p) + Bin(n_2, p) = Bin(n_1 + n_2, p)$$

falls die lambda's und n's unabhängig sind

**Rechenregeln für Momente:**

$$\begin{aligned} \mathbb{E}[X + Y] &= \mathbb{E}[X] + \mathbb{E}[Y] & \forall X, Y \\ \mathbb{E}[X \cdot Y] &= \mathbb{E}[X] \cdot \mathbb{E}[Y] & \forall X, Y \text{ unabhängig} \\ \text{Var}[X + Y] &= \text{Var}[X] + \text{Var}[Y] & \forall X, Y \text{ unabhängig} \\ \text{Var}[X \cdot Y] &\neq \text{Var}[X] \cdot \text{Var}[Y] & \text{muss ausgerechnet werden} \end{aligned}$$

**Multiplikativität des Erwartungswerts:** Für unabhängige Zufallsvariablen  $X_1, \dots, X_n$  gilt:

$$\mathbb{E}[X_1 \cdot \dots \cdot X_n] = \mathbb{E}[X_1] \cdot \dots \cdot \mathbb{E}[X_n]$$

**Satz:** Für unabhängige Zufallsvariablen  $X_1, \dots, X_n$  und  $X := X_1 + \dots + X_n$  gilt:

$$\text{Var}[X] = \text{Var}[X_1] + \dots + \text{Var}[X_n]$$

## 2.3 Abschätzen von Wahrscheinlichkeiten

**Waldsche Identität:** N und X seien zwei unabhängige Zufallsvariable, wobei für den Wertebereich von N gelte:  $W_N \subseteq \mathbb{N}$ . Weiter sei:

$$Z := \sum_{i=1}^N X_i$$

wobei  $X_1, X_2, \dots$  unabhängige Kopien von X seien. Dann gilt:

$$\mathbb{E}[Z] = \mathbb{E}[N] \cdot \mathbb{E}[X]$$

**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

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t} \quad \forall X \geq 0, \forall t > 0$$

**Ungleichung von Chebyshev:** Sei  $X$  eine Zufallsvariable und  $t \in \mathbb{R}$  mit  $t > 0$  Dann gilt:

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\text{Var}[X]}{t^2} \quad \forall X, \forall t > 0$$

insbesondere

$$\Pr[X \geq \mathbb{E}[X] + t] \leq \frac{\text{Var}[X]}{t^2}$$

**Ungleichung von Chernoff:** Die Obergrenze von Chernoff liefert ein viel kleineres Fehler als dass von Chebyshev

$$\Pr[X \geq (1 + \delta)\mathbb{E}[X]] \leq e^{-\frac{1}{3}\delta^2\mathbb{E}[X]} \quad \forall X \sim \text{Bin}(n, p), \forall 0 < \delta < 1$$

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

$$(i) \Pr[X \geq (1 + \delta)\mathbb{E}[X]] \leq e^{-\frac{1}{3}\delta^2\mathbb{E}[X]} \quad \text{für alle } 0 < \delta \leq 1$$

$$(ii) \Pr[X \leq (1 - \delta)\mathbb{E}[X]] \leq e^{-\frac{1}{2}\delta^2\mathbb{E}[X]} \quad \text{für alle } 0 < \delta \leq 1$$

$$(iii) \Pr[X \geq t] \leq 2^{-t} \quad \text{für } t \geq 2e\mathbb{E}[X]$$

## 2.4 Randomisierte Algorithmen:

**Target-Shooting:** Gegeben zwei endliche Mengen  $S \subseteq U$  bestimme  $|S|/|U|$ . Annahmen:

- Wir können ein Element aus  $U$  effizient zufällig gleichverteilt wählen
- es gibt eine effizient berechenbare Funktion

$$\mathbb{I}_S(u) := \begin{cases} 1 & \text{falls } u \in S \\ 0 & \text{sonst} \end{cases}$$

---

### TARGET-SHOOTING

---

1: Wähle  $u_1, \dots, u_N \in U$  zufällig, gleichverteilt und unabhängig  
 2: return  $N^{-1} \cdot \sum_{i=1}^N \mathbb{I}_S(u_i)$

---

Notation:  $Y_i := \mathbb{I}_S(u_i) \quad \text{für alle } i=1, \dots, N$

$Y_1, \dots, Y_N$  unabhängige Bernoulli-Variablen mit  $\Pr[Y_i = 1] = |S|/|U|$

$$Y := \frac{1}{N} \sum_{i=1}^N Y_i = \frac{1}{N} \sum_{i=1}^N \mathbb{I}_S(u_i)$$

Dann gilt:  $\mathbb{E}[Y] = |S|/|U| \dots$  unabhängig von der Wahl von  $N$

$$\text{Var}[Y] = \frac{1}{N} \left( \frac{|S|}{|U|} - \left( \frac{|S|}{|U|} \right)^2 \right)$$

Seien  $\delta, \epsilon > 0$ . Falls  $N \geq 3 \frac{|U|}{|S|} \cdot \epsilon^{-2} \cdot \log(2/\delta)$  so ist die Ausgabe des Algorithmus TARGET-SHOOTING mit Wahrscheinlichkeit mindestens  $1 - \delta$  im Intervall  $[(1 - \epsilon) \frac{|S|}{|U|}, (1 + \epsilon) \frac{|S|}{|U|}]$

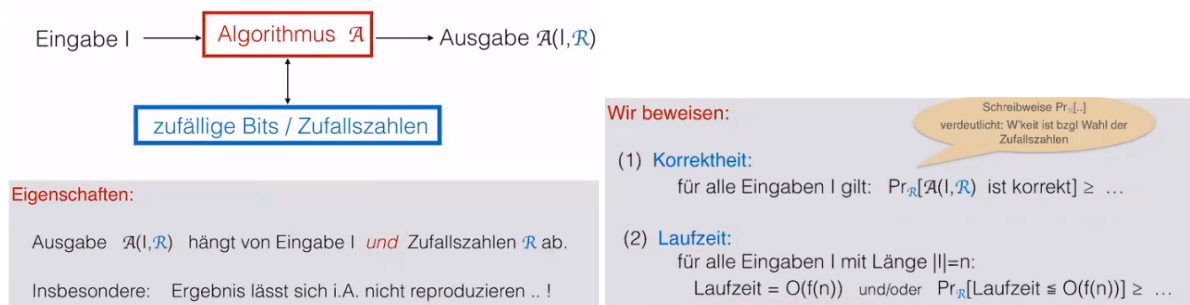
**Las-Vegas Algorithmen:** Geben nie eine falsche Antwort, aber manchmal keine Antwort (Ausgabe = ???).

Ziel:  $\Pr[\text{Antwort} = ???] = \text{winzig}$

**Monte-Carlo Algorithmen:** Geben immer eine Antwort aber manchmal auch eine falsche Antwort.

Ziel:  $\Pr[\text{AntwortFalsch}] = \text{winzig}$

**Entscheidungsproblem:** Problem wo man entscheidet zwischen Ja oder Nein.



**Fehlerkorrektur:** Für Las Vegas und Monte Carlo Algorithmen kann man durch wiederholungen den fehler kleiner machen. Beispiel: Las Vegas

$$\Pr[\text{Antwort} = ???] \leq 1 - \epsilon$$

$$\Rightarrow N := \epsilon^{-1} \ln(\delta^{-1}) \text{ Wiederholungen reduzieren den Fehler auf } \Pr[\text{Antwort immer} ???] \leq (1 - \epsilon)^N \leq \delta$$

In der Praxis ruft man Las-Vegas-Algorithmen meist einfach so lange auf, bis sie eine Antwort geben.

Beispiel Monte Carlo Algorithmen für Entscheidungsprobleme: Einseitiger Fehler:

$$\Pr[\text{Antwort falsch} | \text{Antwort} = \text{Ja}] = 0$$

$$\Pr[\text{Antwort falsch} | \text{Antwort} = \text{Nein}] \leq 1 - \epsilon$$

$$\Rightarrow \epsilon^{-1} \ln(\delta^{-1}) \text{ Wiederholungen reduzieren Fehler auf } \Pr[\text{Antwort falsch}] \leq \delta$$

Der Algorithmus liefert Ja, wenn ein Aufruf Ja ausgibt und Nein, wenn alle Wiederholungen Nein ausgeben.

Zweiseitiger Fehler (Algorithmus kann falsch sein wenn es die Antwort Ja oder Nein ausgibt):

$$\Pr[\text{Antwort falsch}] \leq \frac{1}{2} - \epsilon \Rightarrow 12(\frac{1}{2} - \epsilon) \ln(\delta^{-1}) \text{ Wiederholungen reduzieren Fehler auf } \Pr[\text{Antwort falsch}] \leq \delta$$

Der Algorithmus liefert die Mehrheit der gesehenen Antworten (i.e Ja falls mehr Ausgaben Ja waren sonst Nein)