Algorithms and Probability PVK

Day 3

Michelle Honegger - CAB G59

1. Three events A, B, C are independent if and only if $P[A \cap B \cap C] = P[A]P[B]P[C]$.

2. For any three events A, B, C, we have $P[A \cup B \cup C] = P[A] + P[B] + P[C] - P[A \cap B] - P[A \cap C] - P[B \cap C] + P[A \cap B \cap C]$.

3. We consider the following random experiment: First, we roll a six-sided die and then flip a coin. This random experiment can be described by the sample space $\Omega = \{1, 2, 3, 4, 5, 6, H, T\}$.

4. For any two events A, B with $P[A] > 0$ and $P[B] > 0$ we have $P[A \mid B]\, P[B] = P[B \mid A]\, P[A]$.

5. You roll a six-sided die. Then the events A = “the result is even” and B = “the result is 6” are independent.

6. If A, B, C are three events satisfying $P[A \cap B] = P[A]P[B]$, $P[A \cap C] = P[A]P[C]$, and $P[B \cap C] = P[B]P[C]$, then A, B, C are independent.

7. If A, B, C are three independent events, then $P[A \cap B] = P[A]P[B]$, $P[B \cap C] = P[B]P[C]$, and $P[A \cap C] = P[A]P[C]$.

8. We have 180 random people in a room. The probability that two of them share the same birthday is less than 1/2.

9. Suppose G is a graph that contains an Euler tour, and the number of vertices of G is even. Then G contains a perfect matching.

10. If A and B are independent events then $P[A \cup B] = P[A] + P[B]$.

Markov

Satz *(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

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}.$$

Chebyshev

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

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

Chernoff

Satz (Chernoff-Schranken). Seien $X_1, \ldots, X_n$ unabhängige Bernoulli-verteilte Zufallsvariablen mit $\Pr[X_i = 1] = p_i$ and $\Pr[X_i = 0] = 1 - p_i$. Dann gilt für $X := \sum_{i=1}^{n} X_i$:

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

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

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

Exercise

Aufgabe 2 – *Obere Schranken*

Wir werfen einen fairen 6-seitigen Würfel $n \geq 1$ mal. Sei X die Anzahl der Würfe, bei denen der Würfel '6' zeigt. Berechnen Sie eine möglichst gute obere Schranken für $\Pr[X \geq 2n/9]$.

(a) Mithilfe der Ungleichung von Markov. *(1 Punkte)*

(b) Mithilfe der Ungleichung von Chebychev. *(2 Punkte)*

(c) Mithilfe der Chernoff-Schranken. *(2 Punkte)*

Exercise 16. Consider a football league. A victory gives 3 points and a loss 1 points. There is no draw possible. Each team plays 25 games. Consider a team that wins each game independently with probability $p = 0.6$. Let X be the random variable for the number of points obtained by the team. Compute $\mathbb{E}[X]$ and $\mathrm{Var}[X]$.

Exercise 20. Consider a test with 18 questions and two possible answers to each question. If one guesses each answer, what is the probability of answering correctly to at least 5 and at most 13 answers? Give an expression for the exact probability (no need to calculate it). Then use all three inequalities if possible to give bounds on that probability and compare the results.

Randomised Algorithms

- Deterministic: Same input -> same output
- Non-deterministic: same input not necessarily same output

- Monte Carlo: Correctness as random variable (always same runtime)
- Las Vegas: Runtime as random variable (for us: after some time we return ???)

Error Reduction

Las-Vegas Algorithms

- Let A be a Las-Vegas algorithm with P[A(I) correct] >= ϵ
- Let A_δ for δ > 0 be an algorithm that either returns the first output different to ??? Or after N = $\epsilon^{-1} \cdot ln(\delta^{-1})$ tries it returns ???
- Then it holds Pr[A_δ(i) wrong] >= δ

ϵ	δ	N
0.1	0.01	47
0.5	0.01	10
0.5	10^{-80}	369
0.9	10^{-30}	77
$\in (0,1)$	> 1	< 0

Error Reduction

Monte Carlo - One-Sided Error

- Pr[A(I) = Yes] = 1 for all yes-instances I
- Pr[A(I) = No] >= ϵ = 1 - δ for all no-instances

- Let A_δ for δ > 0 be an algorithm that either returns no, as soon as he first sees no, or that returns yes after N = $\epsilon^{-1} \cdot ln(\delta^{-1})$ tries

Error Reduction

Monte-Carlo - Two-Sided-Error

- Pr[A(I) correct] >= 0.5 + ϵ for all instances I
- For δ > 0, if N = $4 \cdot \epsilon^{-2} \cdot ln(\delta^{-1})$
- A_δ returns answer that was produced the most after N trials
- Pr[A(I) correct] >= δ for all instances I

Exercise 24. How can we transform a Las Vegas algorithm to a Monte Carlo algorithm? What about the opposite direction?

Quicksort

- Sorts Array in expected runtime O(nlog(n))

```
QuickSort(A, ℓ, r)
1: if ℓ < r then
2:     p ← Uniform({ℓ, ℓ + 1, ..., r})        ▷ wähle Pivotelement zufällig
3:     t ← Partition(A, ℓ, r, p)
4:     QuickSort(A, ℓ, t − 1)
5:     QuickSort(A, t + 1, r)
```

Quickselect

- Returns k-smallest Element in expected runtime O(n)

```
QuickSelect(A, ℓ, r, k)
1: p ← Uniform({ℓ, ℓ + 1, ..., r})          ▷ wähle Pivotelement zufällig
2: t ← Partition(A, ℓ, r, p)
3: if t = ℓ + k − 1 then
4:     return A[t]                           ▷ gesuchtes Element ist gefunden
5: else if t > ℓ + k − 1 then
6:     return QuickSelect(A, ℓ, t − 1, k)    ▷ gesuchtes Element ist links
7: else
8:     return QuickSelect(A, t + 1, r, k − t)  ▷ gesuchtes Element ist rechts
```

Primality Testing

- GCD: Pr[“prime” | not prime] = $\frac{|\mathbb{Z}_n^*|}{n-1}$
- Fermat: All elements have ord(n-1), in case n is prime, Pr[“prime” | not prime] < 1/2 if no number is a charmichaelnumber
  P[“prime” | not prime] = $\frac{|PB_n|}{n-1}$
- Miller Rabin: Pr[“prime” | not prime] <= 1/4
  n-1 = 2^kd

```
MILLER-RABIN-PRIMZAHLTEST(n)
1: if n = 2 then
2:     return 'Primzahl'
3: else if n gerade oder n = 1 then
4:     return 'keine Primzahl'
5: Wähle a ∈ {2, 3, ..., n − 1} zufällig und
6: berechne k, d ∈ ℤ mit n − 1 = d2^k und d ungerade.
7: x ← a^d (mod n)
8: if x = 1 or x = n − 1 then
9:     return 'Primzahl'
10: repeat k − 1 mal
11:     x ← x^2 (mod n)
12:     if x = 1 then
13:         return 'keine Primzahl'
14:     if x = n − 1 then
15:         return 'Primzahl'
16: return 'keine Primzahl'
```

Exercise 25. We stated multiple times that these algorithms have one-sided error. Which answer ("prime" or "not prime") is always correct?

Hase-Igel Algorithm

- Given Array with n Positions, each of which contains value between 1 and n-1
- Construct graph with n vertices and n edges, edges always from I to A[I]
- There exists a path of length $k \geq 1$ from n with cycle of length $l \geq 3$
- We define $s \geq 1$ and $0 \leq r < l$
  - If $k \leq l$ -> $s = 1$, else $s = k/l$ (ceiled)

Hase - Igel Algorithm

- $x_{k+r} = x_{2(k+r)}$ -> find this position with Algo
- Calculate k with $x_k = x_{i+k}$

```
igel = a[n]; hase := a[a[n]]; i := 1
while (igel ≠ hase)
    igel = a[igel]; hase := a[a[hase]]; i := i + 1
```

So far, we mainly used randomization to construct efficient algorithms. However, randomization can also be used to prove statements that don't involve randomness themselves. In fact, such proves can usually be phrased by describing a Monte-Carlo algorithm. Answer the two following questions. If you want to, you can describe your solution as a Monte-Carlo algorithm.

(a) You are given 1000 subsets $A_1, \ldots, A_{1000}$ of $\{1, \ldots, n\}$. Each subset has size at least 11. Can the numbers $1, \ldots, n$ be colored 'red' and 'blue', such that each set A_i contain at least one 'red' and at least one 'blue' number?

(b) You are given 10 points in the plane (i.e., $\mathbb{R}^2$). Can you place (filled) disks of radius 1 in the plane, such that (i) each of the 10 points is contained in a disk, and (ii) all disks are disjoint (i.e., their centers have distance at least 2)?
Remark: You may cover multiple points with the same disk. In particular, the task would be trivial if all points are very close to each other (then you only need one disk to cover all of them). Similarly, if all points are very far apart, you could use one disk per point without having to worry about disjointness.
Remark: a formal solution to this exercise would be very technical. Focus more on the ideas than on technical details

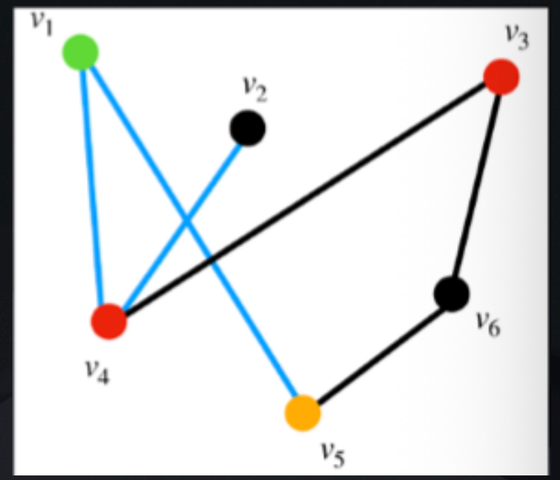
Long Paths

- Given Graph G and number B, there exists a path in G with length B?
- Hamiltonian cycle hard -> long paths hard
  - Construct G’ where we replace v and incident edges {v,wi} with {wi’,wi}
  - This has ≤ 2n - 2 vertices
  - Show construction of hamiltonian cycle in that graph

Long Paths

Coloured Paths

- **Given**: a graph G and a colouring y : V -> [k] (the colouring doesn't need to be correct as in chapter 1)
- **Seeked**: does G contain a coloured path of length k-1?
  - Coloured path: all vertices have different colour
- $P_i(v)$:= all coloured paths of length I, that end in v



- $P_i(v) := \{S \in \binom{[k]}{i+1} \mid \exists$ a coloured path, that ends in v and is coloured with the colours in S$\}$
- **Seeked**: is $\bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$

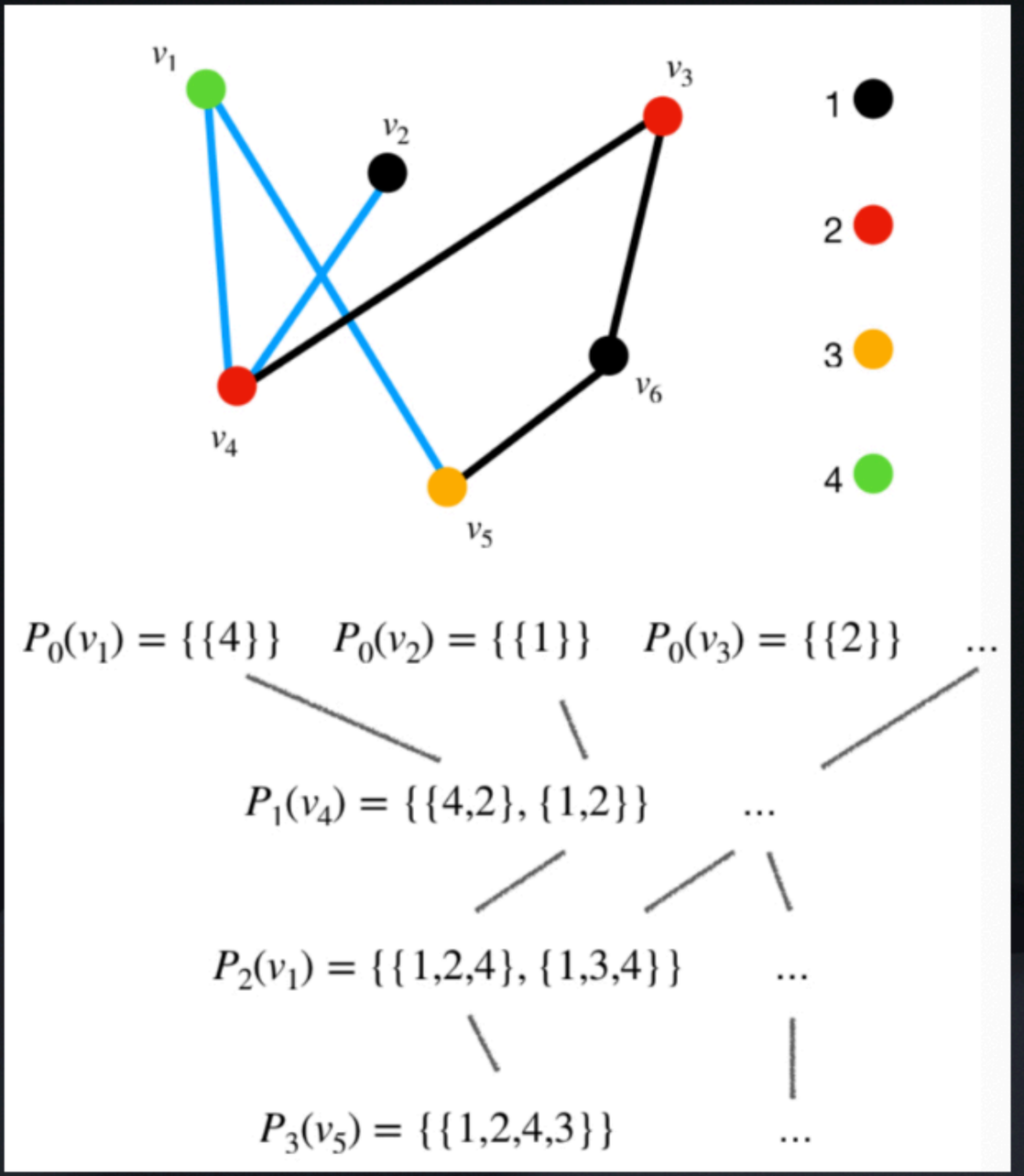
Lange Pfade

Bunte Pfade

BUNT(G, i)

1: **for all** $v \in V$ **do**
2: $\quad P_i(v) \leftarrow \emptyset$
3: $\quad$ **for all** $x \in N(v)$ **do**
4: $\quad\quad$ **for all** $R \in P_{i-1}(x)$ **mit** $\gamma(v) \notin R$ **do**
5: $\quad\quad\quad P_i(v) \leftarrow P_i(v) \cup \{R \cup \{\gamma(v)\}\}$

- findet alle bunten Pfade der Länge i, basierend auf bunten Pfaden der Länge i-1.
- $O(\binom{k}{i} \cdot i \cdot m)$



Lange Pfade

Bunte Pfade

BUNT(G, i)

1: **for all** $v \in V$ **do**
2: $\quad P_i(v) \leftarrow \emptyset$
3: $\quad$ **for all** $x \in N(v)$ **do**
4: $\quad\quad$ **for all** $R \in P_{i-1}(x)$ **mit** $\gamma(v) \notin R$ **do**
5: $\quad\quad\quad P_i(v) \leftarrow P_i(v) \cup \{R \cup \{\gamma(v)\}\}$

- findet alle bunten Pfade der Länge i, basierend auf bunten Pfaden der Länge i-1.
- $O(\binom{k}{i} \cdot i \cdot m)$

Regenbogen(G, γ)

1: **for all** $v \in V$ **do** $P_0(v) \leftarrow \{\{\gamma(v)\}\}$
2: **for** $i = 1..k-1$ **do** Bunt(G, i)
3: **return** $\bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$

findet, ob es einen bunten Pfad der Länge k-1 gibt.

$O(2^k km)$

Long Paths

- **Given**: a graph G and k in natural numbers
- **Seeked**: does G contain a path of length k? -> NP-hard can show through reduction to hamiltonian path problem
- We solve rainbow(G,y'), where y' : V -> [k+1] is a u.a.r. distributed colouring with k+1 colours.

1 Versuch	$\lceil \lambda e^k \rceil$ Versuche
$O(2^k km)$	$O(\lambda e^k \cdot 2^k km)$
$P_{Erfolg} \geq e^{-k}$	$P_{Erfolg} \geq 1 - e^{-\lambda}$

Lange Pfade

- **gegeben**: ein Graph G und k $\in \mathbb{N}_0$
- **gesucht**: ob G einen Pfad der Länge k enthält: NP-schwer zeigen durch Reduktion von HamiltonPfad-Problem
- Wir lösen Regenbogen (G, γ'), wobei $\gamma' : V \to [k+1]$ ist eine gleichverteilte Färbung mit k+1 Farben
-

1 Versuch	$\lceil \lambda e^k \rceil$ Versuche
$O(2^k km)$	$O(\lambda e^k \cdot 2^k km)$
$P_{Erfolg} \geq e^{-k}$	$P_{Erfolg} \geq 1 - e^{-\lambda}$

Probability DP

- Frog Feeding in Code Expert