Algorithms and Probability

Week 10

Georg Hasebe — 02/05/2024

Remarks on Exercise Sheet 4

$$X_i := \begin{cases} 1 & \text{Hund } i \text{ hat Vergnügen.} \\ 0 & \text{sonst.} \end{cases}$$

1. $\Pr[X_i = 1] \leq 4/k$ by subtask (b), then $\Pr[X_i = 0] \geq (1 - 4/k)$.

2. Let $X :=$ "# vergnügliche Spaziergänge" $= \sum_{i=1}^{n} X_i$, $\Pr[X = n] \cancel{\geq} (1 - 4/k)^n$.

Das zwei Spaziergänge k/ein Vergnügen sind, ist **nicht unabhängig**!

Nehme an dass Hunde i, j selbe Spazier-Route haben, dann gilt:

$$\Pr[X_i = 1 \,|\, X_j = 1] = 1 \neq \Pr[X_j = 1].$$

- Bitte immer alle Zufallsvariablen definieren!

Minitest 5

Seien X, Y unabhängige Zufallsvariablen mit $Var(X) = 2$ und $Var(Y) = 4$.

Berechnen Sie $Var(X - Y)$.

Satz 2.41. Für eine beliebige Zufallsvariable X und $a, b \in \mathbb{R}$ gilt

$$\mathrm{Var}[a \cdot X + b] = a^2 \cdot \mathrm{Var}[X].$$

Satz 2.62. Für unabhängige Zufallsvariablen $X_1, \ldots, X_n$ und $X := X_1 + \ldots + X_n$ gilt

$$\mathrm{Var}[X] = \mathrm{Var}[X_1] + \ldots + \mathrm{Var}[X_n].$$

Wir betrachten eine Variante des Target Shooting Algorithmus, den wir in der Vorlesung gesehen haben. Bitte beachten Sie, dass es sich nicht um genau den gleichen Algorithmus handelt. Wir gehen, wie in der Vorlesung davon aus, dass es eine unbekannte Menge S gibt, die Teilmenge einer bekannten Menge U ist. Ausserdem können wir Elemente $u \in U$ uniform zufällig auswählen und überprüfen, ob $u \in S$.

Angenommen, wir wählen so oft zufällige Elemente $u \in U$ aus, bis wir insgesamt 100 Elemente gesehen haben, die $u \in S$ erfüllen. Sei X die Anzahl an Elementen, die wir auswählen müssen, bis das zutrifft.

- X ist geometrisch verteilt. ✗
- Falls $|S| = |U|$, dann $X = 100$. ✓
- $\mathbb{E}[X] = 100|U|/|S|$. ✓
- Falls $X = 100$, dann $|S| = |U|$. ✗

Ein einfacher Algorithmus zum Finden des kleinsten umschliessenden Kreises einer Punktemenge P iteriert durch alle Tripel $Q \in \binom{P}{3}$. Ein einfacher Trick um die Laufzeit dieses Algorithmus zu verkürzen, ist es, die Tripel $Q \in \binom{P}{3}$ wiederholt uniform zufällig auszuwählen (ohne P zu verändern) bis der Umkreis gefunden ist.

Betrachten Sie den folgenden Algorithmus (Fermat-Primzahltest)

Wähle $a \in [n-1]$ zufällig gleichverteilt

if $\mathrm{ggT}(a, n) > 1$ oder $a^{n-1} \not\equiv 1 \pmod{n}$ **then**

    return 'keine Primzahl'

else

    return 'Primzahl'

Die Ausgabe 'keine Primzahl' ist immer richtig	✓
Die Ausgabe 'Primzahl' ist immer richtig	✗

Sei $P = \{p_1, p_2, p_3\}$ eine Menge von drei Punkten in der Ebene in allgemeiner Lage. Der kleinste umschliessende Kreis von P ist der eindeutige Kreis, der alle drei Punkte auf seinem Rand hat.

Seien A, B, C drei Ereignisse in einem Wahrscheinlichkeitsraum mit $Pr[A], Pr[B], Pr[C] > 0$.

Falls $Pr[A \cup B] = \mathrm{Pr}[A] + \mathrm{Pr}[B]$, dann $Pr[A \cap B] = 0$.

Falls A, B, C unabhängig sind, dann gilt
$Pr[A \cap (B \cup C)] = Pr[A] \cdot Pr[B \cup C]$.

Falls $\mathrm{Pr}[A \cap B] = Pr[A] \cdot Pr[B]$, dann sind A und B unabhängig.

Falls $Pr[A] \leq Pr[B]$, dann $Pr[A \cup C] \leq Pr[B \cup C]$.

Theory Recap

Lange Pfade

Gegeben: (G, B), G ein Graph und $B \in \mathbb{N}_0$.

Problem: gibt es einen Pfad der Länge B in G?

Zur Erinnerung:

Ein **Pfad der Länge** ℓ in einem Graph $G = (V, E)$ ist eine Folge von paarweise verschiedenen Knoten

$$\langle v_0, v_1, \ldots, v_\ell \rangle, \text{ mit } \{v_{i-1}, v_i\} \in E \quad \text{für} \quad i = 1, \ldots \ell.$$

Es liegen $\ell + 1$ Knoten auf einem Pfad der Länge ℓ.

Lange Pfade

Für beliebige $B \in \mathbb{N}_0$ vermutlich sehr schwer (vgl. $B = n - 1 \Rightarrow$ Hamiltonpfad).

Was passiert wenn B klein ist?

Konkret:

$$B = O(\log n).$$

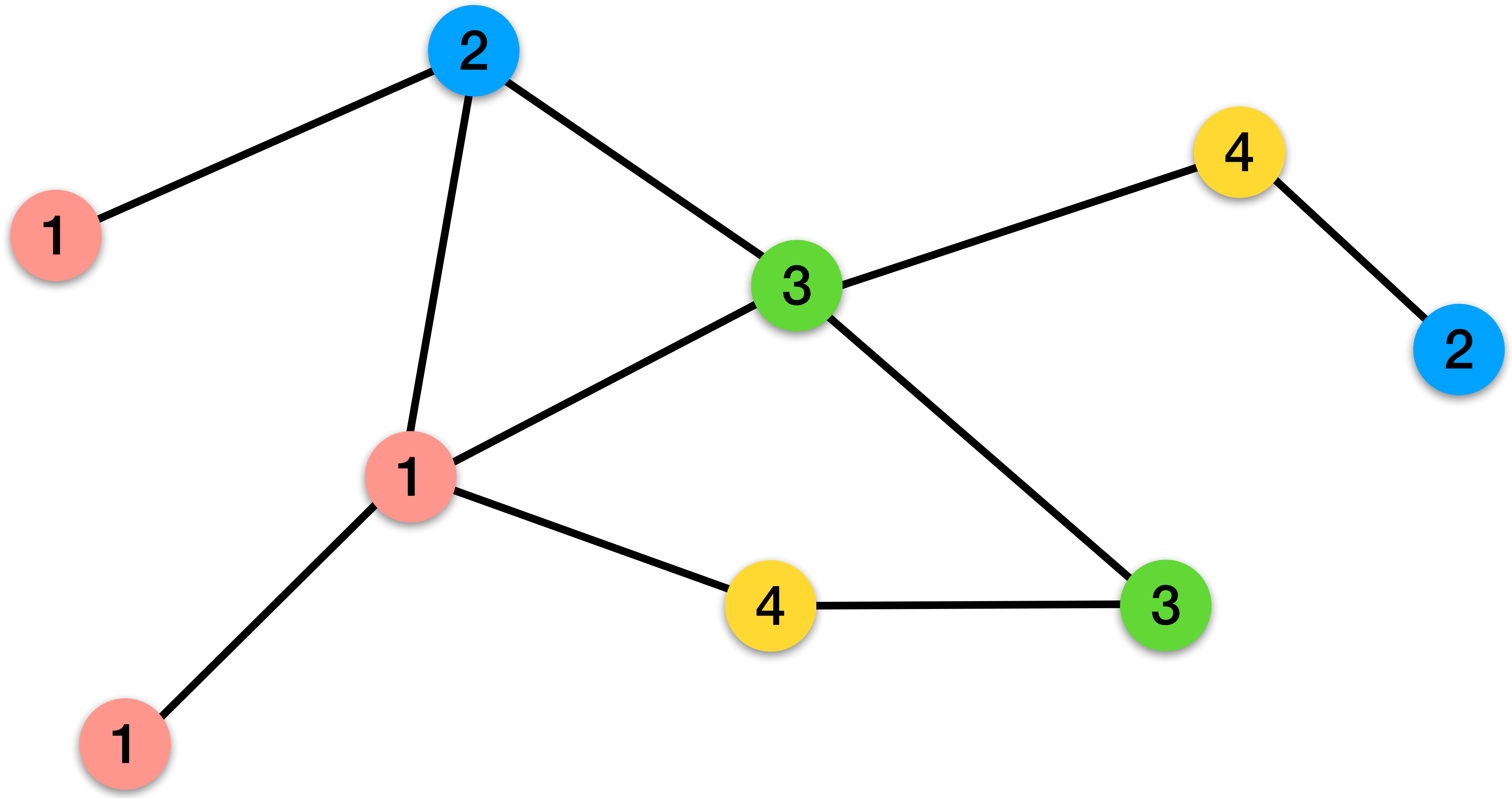
Colorful-Path Probelm

Gegeben: (G, γ), $G = (V, E)$ ein Graph und $\gamma : V \to [k]$ eine Färbung (muss nicht gültig sein; d.h.  ist erlaubt)

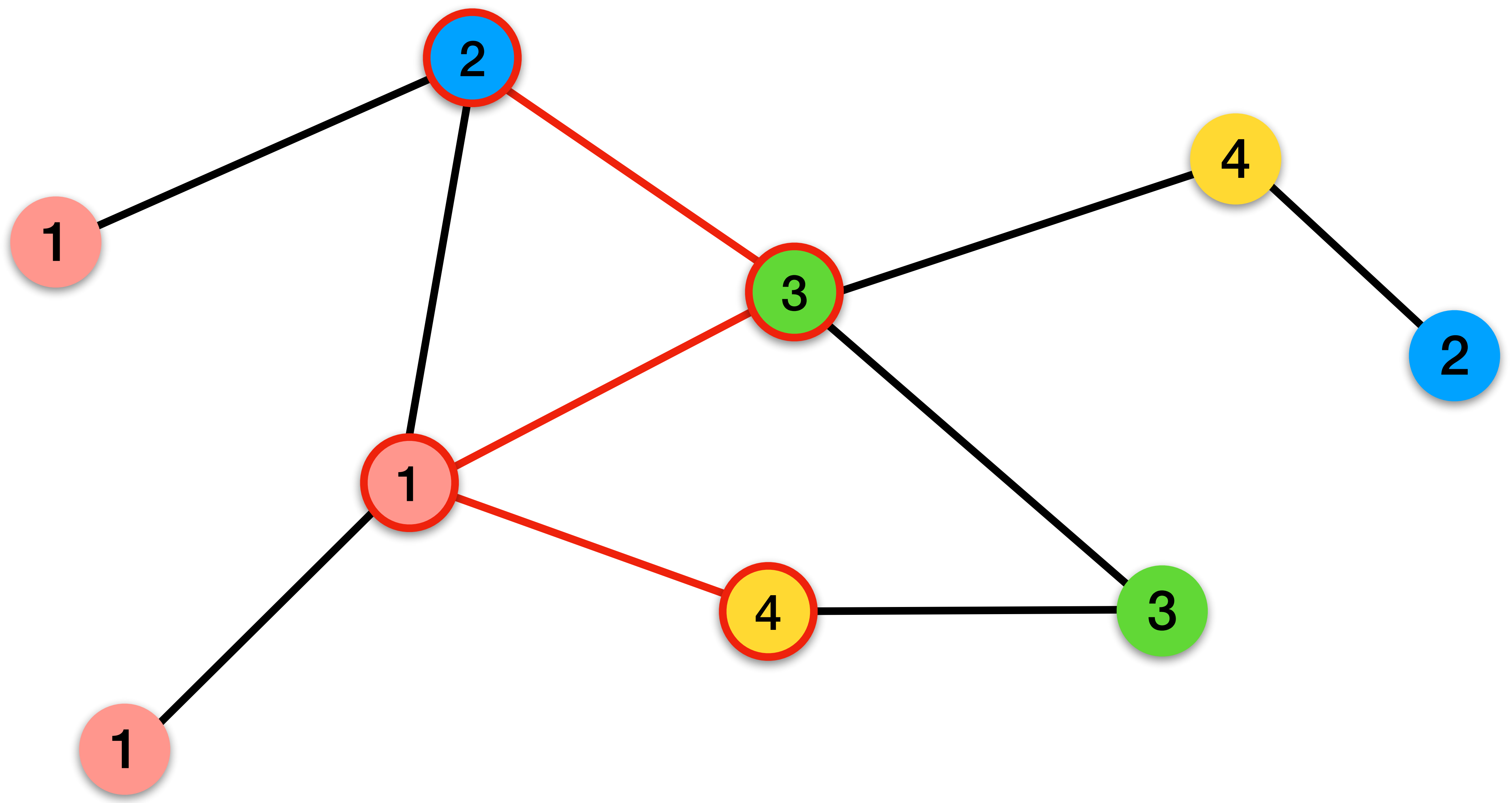
d.h. mit k Knoten

Problem: gibt es einen bunten Pfad der Länge $k - 1$ in G?

Definition: Ein Pfad heisst **bunt**, falls alle seine Knoten verschiedene Farben haben.



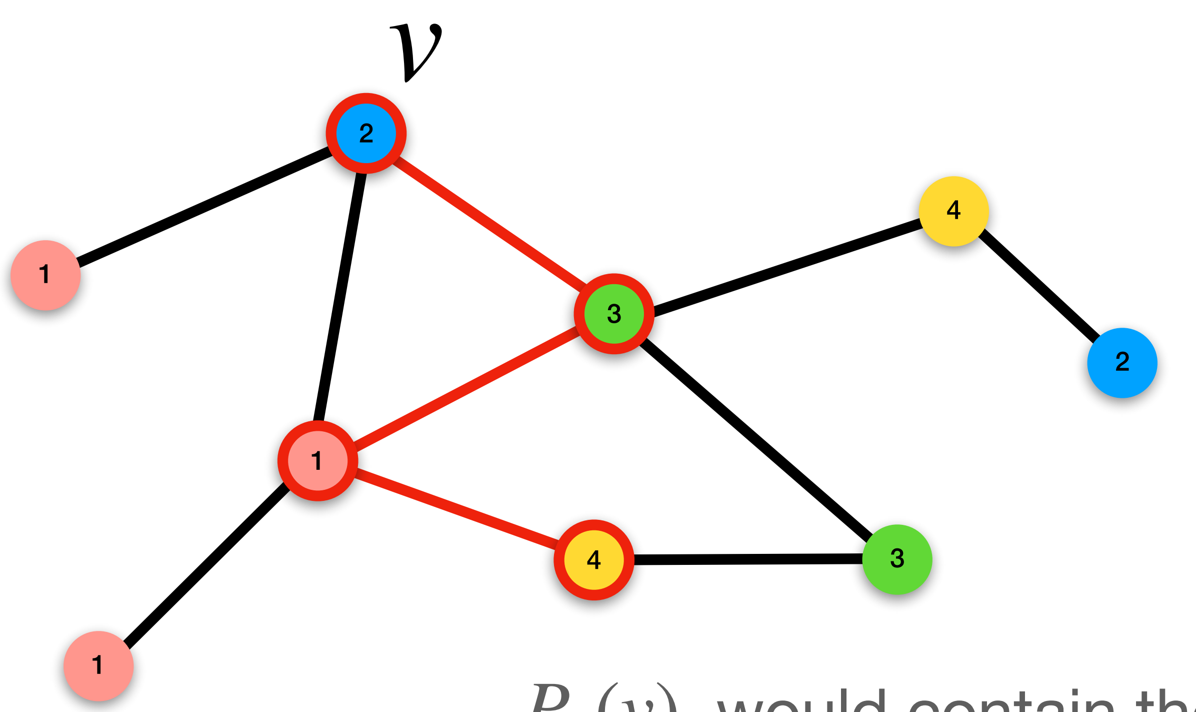
Graph $G = (V, E)$ mit Färbung $\gamma : V \to [4]$.



Ein bunter Pfad der Länge 3.

Definition: Ein Pfad heisst **bunt**, falls alle seine Knoten verschiedene Farben haben.

DP Algorithmus



$P_3(v)$ would contain the red path.

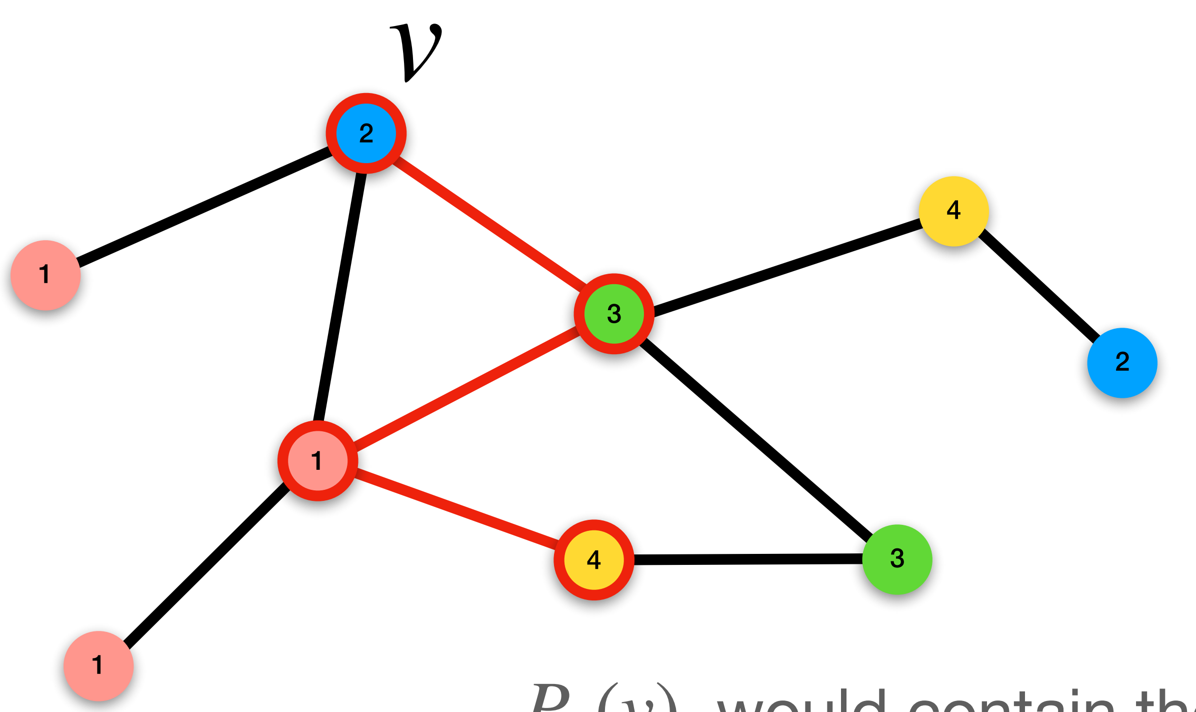
Idee:

$$P_i(v) = \text{"Menge aller in } v \text{ endender bunten Pfaden der Länge } i\text{"}.$$

Wir brauchen:

Rekursion um von $P_i(v)$ zu $P_{i+1}(v)$ zu kommen.

Die **Lösung** (bunter Pfad der Länge $k-1$, falls existent) ist in $\bigcup_{v \in V} P_{k-1}(v)$.



$P_3(v)$ would contain the red path.

DP Algorithmus

Idee:

$$P_i(v) = \text{"Menge aller in } v \text{ endender bunten Pfaden der Länge } i\text{"}.$$

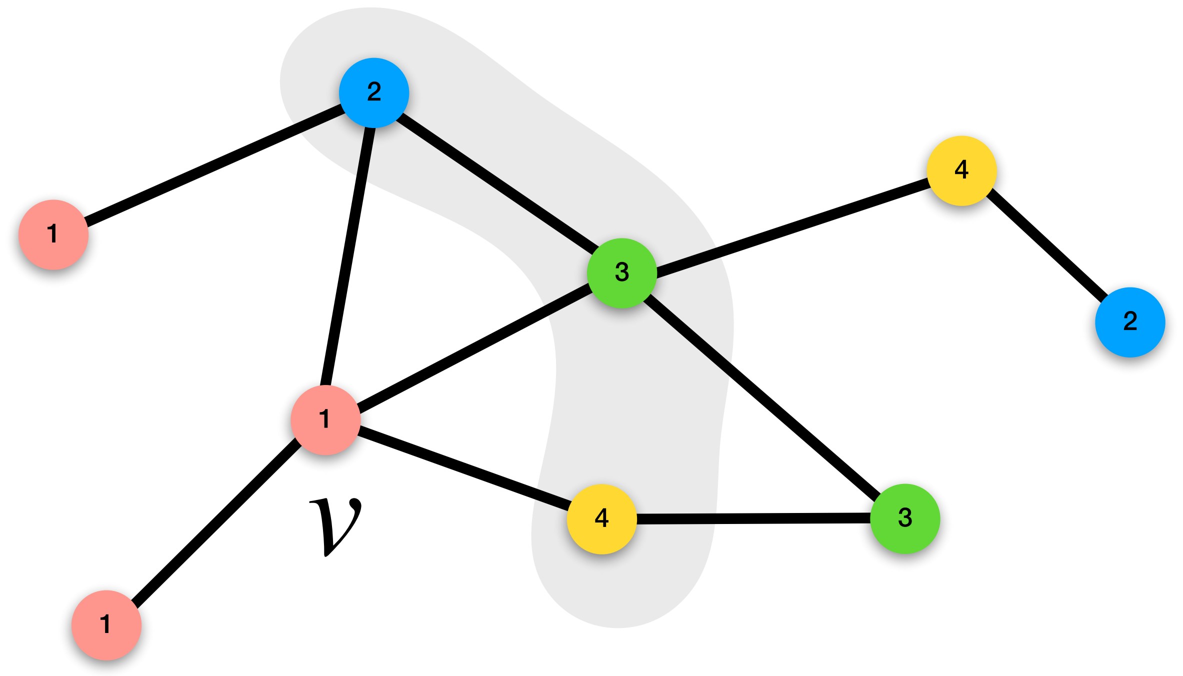
Wir definieren:

$$P_i(v) := \{S \in \underbrace{\binom{[k]}{i+1}} \mid \exists \text{ in } v \text{ endender mit genau } S \text{ gefärbter bunter Pfad}\}.$$

Ein bunter Pfad kann eine verschiedene Farben-Abfolge haben. Bei k Farben gibt es genau $\binom{k}{i+1}$ Möglichkeiten um aus k Farben $i+1$ auszuwählen. Dieser Gedanke motiviert auch die Notation.

$P_i(v) := \{S \in \binom{[k]}{i+1} \mid \exists$ in v endender mit genau S gefärbter bunter Pfad$\}$.

DP Algorithmus



Wie kommen wir nun zu unserer Rekursion?

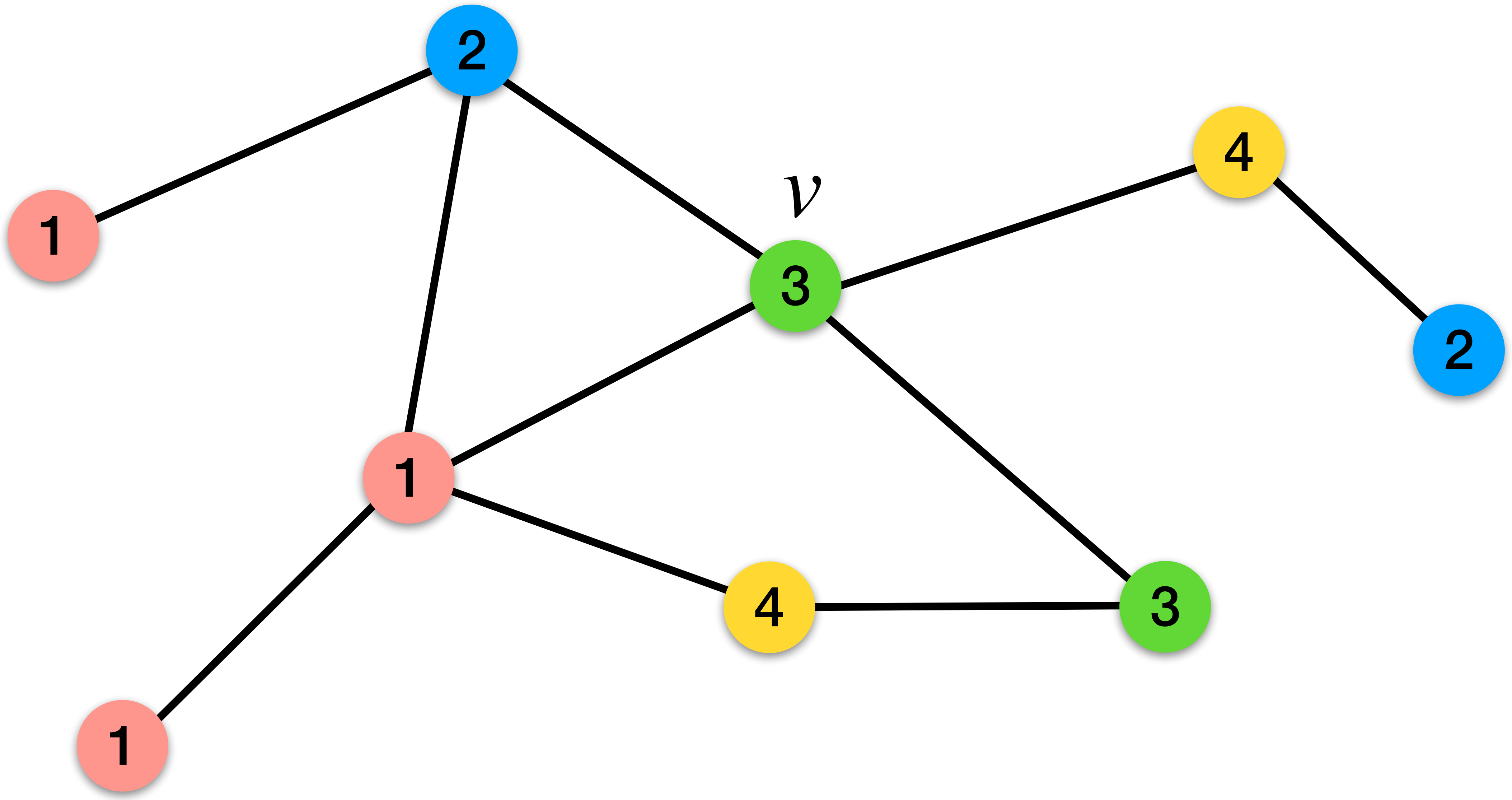
Was ist $P_0(v)$?

Ein in v endender bunter Pfad (Länge 0) $\Rightarrow P_0(v) = \{\{\gamma(v)\}\}$.

Was ist $P_1(v)$?

Ein bunter Pfad der Länge 1 zu v besteht aus einem $\gamma(v)$-freien bunten Pfad der Länge 0 zu einem Nachbarn x von v plus dem Schritt zu v.

$$P_1(v) = \{\{\gamma(x), \gamma(v)\} \mid x \in N(v), \gamma(x) \neq \gamma(v)\}.$$



$$P_0(v) = \{ 3 \}$$

$$P_1(v) = \{ \{ \gamma(x), 3 \} \mid x \in N(v), \gamma(x) \neq \gamma(v) \} = \{ \{ 3 , 4 \}, \{ 3 , 2 \}, \{ 3 , 1 \} \}$$

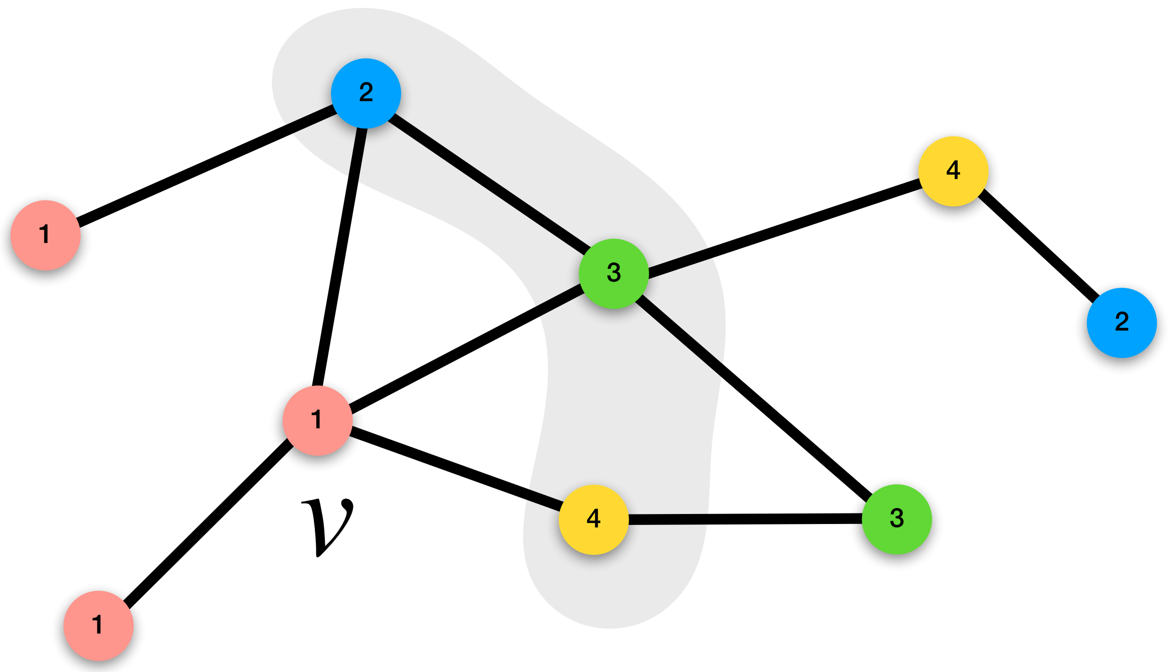
$P_i(v) := \{S \in \binom{[k]}{i+1} \mid \exists$ in v endender mit genau S gefärbter bunter Pfad$\}$.

DP Algorithmus

Die vorherigen Überlegungen motivieren folgende Rekursion:

$$P_i(v) = \bigcup_{x \in N(v)} \{R \cup \{\gamma(v)\} \mid R \in P_{i-1}(x) \text{ und } \gamma(v) \notin R\}.$$

Ein bunter Pfad der Länge i zu v besteht aus einem $\gamma(v)$-freien bunten Pfad der Länge $i-1$ zu einem Nachbarn x von v plus dem Schritt zu v.



Rekursion ähnelt dem Algorithmus für Hamiltonkreise mit dynamischer Programmierung.

Betrachten Sie einen Graph $G = (V, E)$ mit $|V| = n$ Knoten und eine k-(Knoten)-Färbung $c : V \to [k]$, wobei $k = \lceil \log n \rceil$.

Welche der folgenden Definitionen kann benutzt werden, um mittels DP in polynomieller Zeit (in n) zu entscheiden, ob G einen bunten Pfad mit k Knoten enthält?

$C_i(v) :=$ { Farbfolgen c $\in [k]^i$ so dass $\exists$ ein Pfad mit i Knoten beginnend bei v mit Farben $c_1, c_2, \ldots$ }

$A_i(v) := \{S \in \binom{[k]}{i}$: $\exists$ ein Pfad mit i Knoten und Endknoten v, der alle Farben von S genau einmal verwendet$\}$

$B_i(v) :=$ {alle bunten Pfade mit i Knoten und v als Endknoten}

DP Algorithmus

$$P_i(v) := \{S \in \binom{[k]}{i+1} \mid \exists \text{ in } v \text{ endender mit genau } S \text{ gefärbter bunter Pfad}\}.$$

$$P_i(v) = \bigcup_{x \in N(v)} \{R \cup \{\gamma(v)\} \mid R \in P_{i-1}(x) \text{ und } \gamma(v) \notin R\}.$$

Bunt(G, i) — G ein γ-gefärbter Graph

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: $\quad\quad\quad P_i(v) \leftarrow P_i(v) \cup \{R \cup \{\gamma(v)\}\}$

Ein bunter Pfad der Länge i zu v besteht aus einem $\gamma(v)$-freien bunten Pfad der Länge $i-1$ zu einem Nachbarn x von v plus dem Schritt zu v.

$$P_i(v) := \{S \in \binom{[k]}{i+1} \mid \exists \text{ in } v \text{ endender mit genau } S \text{ gefärbter bunter Pfad}\}.$$

$$P_i(v) = \bigcup_{x \in N(v)} \{R \cup \{\gamma(v)\} \mid R \in P_{i-1}(x) \text{ und } \gamma(v) \notin R\}.$$

DP Algorithmus

Regenbogen(G, γ) — G Graph, γ k-Färbung

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

Bunt(G, i) — G ein γ-gefärbter Graph

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

Laufzeit

Regenbogen(G, γ) — G Graph, γ k-Färbung

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

Bunt(G, i) — G ein γ-gefärbter Graph

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

Laufzeit

Regenbogen(G, γ)	G Graph, γ k-Färbung
for all $v \in V$ **do** $P_0(v) \leftarrow \{\{\gamma(v)\}\}$	$O(n)$
for $i = 1..k-1$ **do** Bunt(G, i)	???
return $\bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$	$O(n)$

Bunt(G, i)	G ein γ-gefärbter Graph
for all $v \in V$ **do**	
$\quad P_i(v) \leftarrow \emptyset$	
$\quad$ **for all** $x \in N(v)$ **do**	$\deg(v)$
$\quad\quad$ **for all** $R \in P_{i-1}(x)$ mit $\gamma(v) \notin R$ **do**	$\lvert P_{i-1}(x) \rvert \cdot i$
$\quad\quad\quad P_i(v) \leftarrow P_i(v) \cup \{R \cup \{\gamma(v)\}\}$	

Laufzeit

Bunt(G, i) — G ein γ-gefärbter Graph

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

$\deg(v)$

$|P_{i-1}(x)| \cdot i$

Wir haben also $O\left(\sum_{v \in V} \deg(v) \cdot |P_{i-1}(v)| \cdot i\right)$ pro Bunt(G, i) Runde.

Nach dem Handshake Lemma und weil $P_{i-1}(v) \subseteq \binom{[k]}{i}$ und daher $|P_{i-1}(v)| \leq \binom{k}{i}$ bekommen wir pro Bunt(G, i) Runde:

$$O\left(\sum_{v \in V} \deg(v) \cdot |P_{i-1}(v)| \cdot i\right) = O\left(m \cdot \binom{k}{i} \cdot i\right).$$

Laufzeit

Regenbogen(G, γ)	G Graph, γ k-Färbung
1: **for all** $v \in V$ **do** $P_0(v) \leftarrow \{\{\gamma(v)\}\}$	$O(n)$
2: **for** $i = 1..k-1$ **do** Bunt(G, i)	$O\left(m \cdot \binom{k}{i} \cdot i\right)$
3: **return** $\bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$	$O(n)$

Bunt(G, i)	G ein γ-gefärbter Graph
1: **for all** $v \in V$ **do**	
2: $\quad P_i(v) \leftarrow \emptyset$	
3: $\quad$ **for all** $x \in N(v)$ **do**	$\deg(v)$
4: $\quad\quad$ **for all** $R \in P_{i-1}(x)$ mit $\gamma(v) \notin R$ **do**	$\lvert P_{i-1}(x) \rvert \cdot i$
5: $\quad\quad\quad P_i(v) \leftarrow P_i(v) \cup \{R \cup \{\gamma(v)\}\}$	

Laufzeit

Regenbogen(G, γ)	G Graph, γ k-Färbung	
for all $v \in V$ **do** $P_0(v) \leftarrow \{\{\gamma(v)\}\}$	$O(n)$	
for $i = 1..k-1$ **do** Bunt(G, i)		$O\left(m \cdot \binom{k}{i} \cdot i\right)$
return $\bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$	$O(n)$	

$$\sum_{i=1}^{k-1} m \cdot \binom{k}{i} \cdot i \leq \sum_{i=1}^{k} m \cdot \binom{k}{i} \cdot i \leq O(2^k \cdot k \cdot m).$$

Where we used that:

$$\sum_{i=1}^{k} \binom{k}{i} \cdot i = 2^{k-1} k.$$

Laufzeit

Regenbogen(G, γ)	G Graph, γ k-Färbung
1: **for all** $v \in V$ **do** $P_0(v) \leftarrow \{\{\gamma(v)\}\}$	$O(n)$
2: **for** $i = 1..k-1$ **do** Bunt(G, i)	$O\left(m \cdot \binom{k}{i} \cdot i\right)$
3: **return** $\bigcup_{v \in V} P_{k-1}(v) \neq \emptyset$	$O(n)$

Since we have $O(2^k \cdot k \cdot m)$, for $k \leq O(\log n)$ we have a polynomial algorithm.

Colorful-Path und Lange Pfade

Lange Pfade: (G, B), G ein Graph und $B \in \mathbb{N}_0$.

Für $k = B + 1$, färbe G **zufällig** mit k Farben, und suche einen bunten Pfad mit k Knoten.

Bei zufälliger Färbung, was ist die **Erfolgswahrscheinlichkeit**?

Möglichkeiten einen Pfad der Länge k mit k Farben zu Färben?

$$k^k.$$

Möglichkeiten einen Pfad der Länge k mit k Farben zu färben, so dass dieser bunt ist?

$$k!.$$

Colorful-Path und Lange Pfade

Bei zufälliger Färbung, was ist die **Erfolgswahrscheinlichkeit**?

Taylor series for e^k.

$$p_{\text{Erfolg}} := \Pr[\exists \text{ bunter Pfad der Länge } k-1] \geq \Pr[P \text{ ist bunt}] = \frac{k!}{k^k} \geq e^{-k}.$$

Möglichkeiten einen Pfad der Länge k mit k Farben zu Färben?

$$k^k.$$

Möglichkeiten einen Pfad der Länge k mit k Farben zu färben, so dass dieser bunt ist?

$$k!.$$

Ein Versuch:

- Laufzeit $O(2^k km)$. $\quad p_{\text{Erfolg}} \geq e^{-k}$.

$\lceil \lambda e^k \rceil$ Versuche:

- Laufzeit $O(\lambda(2e)^k km)$.
- W'keit, dass der Algorithmus den Pfad nicht findet ist

$$\leq \left(1-e^{-k}\right)^{\lceil \lambda e^k \rceil} \leq \left(e^{-e^{-k}}\right)^{\lceil \lambda e^k \rceil} \leq e^{-\lambda}.$$

Konvexe Menge, Konvexe Hülle

Sei $d \in \mathbb{N}$.

- Für $v_0, v_1 \in \mathbb{R}^d$ sei

$$\overline{v_0 v_1} := \{(1-\lambda)v_0 + \lambda v_1 \mid \lambda \in \mathbb{R}, 0 \leq \lambda \leq 1\} ,$$

  das v_0 und v_1 verbindende **Liniensegment**.

- Eine Menge $C \subseteq \mathbb{R}^d$ heisst **konvex**, falls

$$\forall v_0, v_1 \in C : \ \overline{v_0 v_1} \subseteq C .$$

- Die **konvexe Hülle**, $\mathrm{conv}(S)$, einer Menge $S \subseteq \mathbb{R}^d$ ist der Schnitt aller konvexen Mengen, die S enthalten, d.h.

$$\mathrm{conv}(S) := \bigcap_{S \subseteq C \subseteq \mathbb{R}^d,\ C \text{ konvex}} C .$$

Desmos Graphing Calculator.

Für jede endliche Punktemenge P gilt, dass $x \in \operatorname{conv}(P)$ für alle $x \in P$.

- Für $v_0, v_1 \in \mathbb{R}^d$ sei

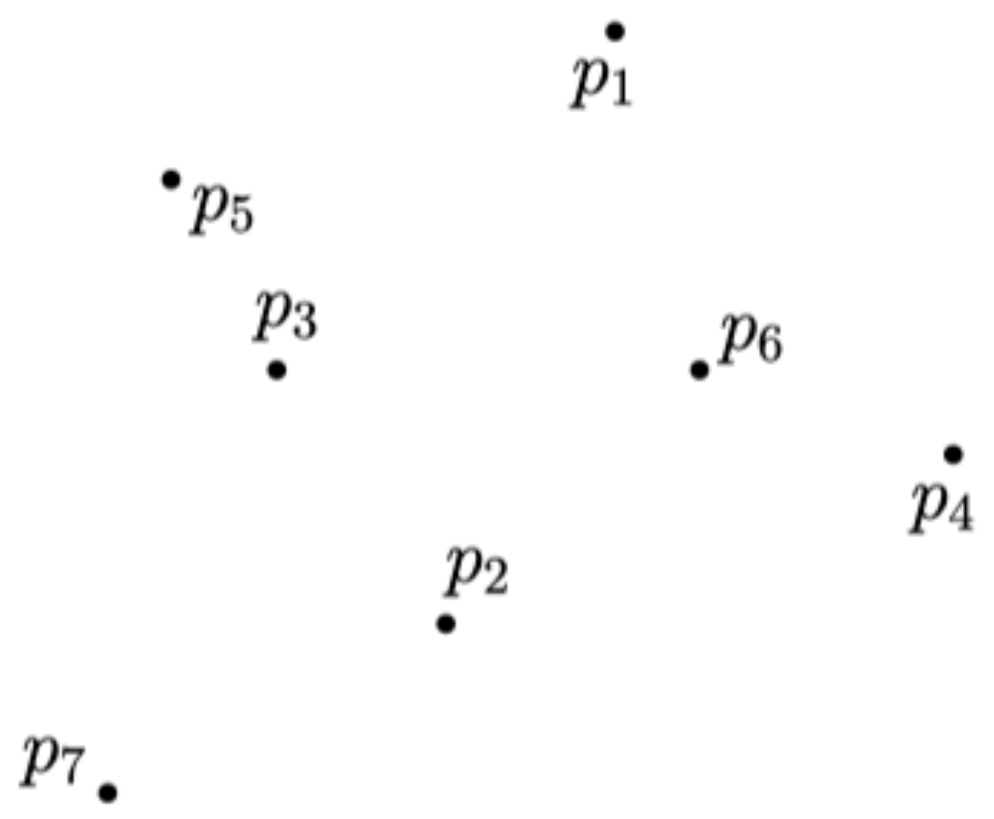
$$\overline{v_0 v_1} := \{(1-\lambda)v_0 + \lambda v_1 \mid \lambda \in \mathbb{R}, 0 \leq \lambda \leq 1\} ,$$

  das v_0 und v_1 verbindende **Liniensegment**.
- Eine Menge $C \subseteq \mathbb{R}^d$ heisst **konvex**, falls

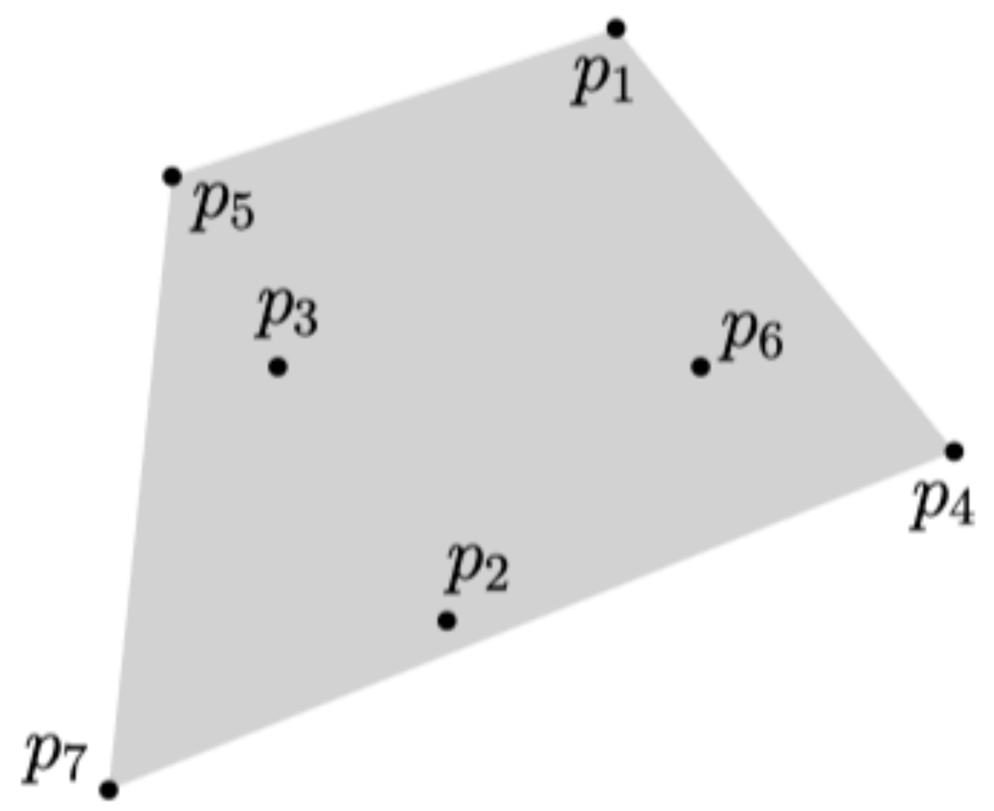
$$\forall v_0, v_1 \in C: \ \overline{v_0 v_1} \subseteq C .$$

Setze λ gleich 0 oder 1...

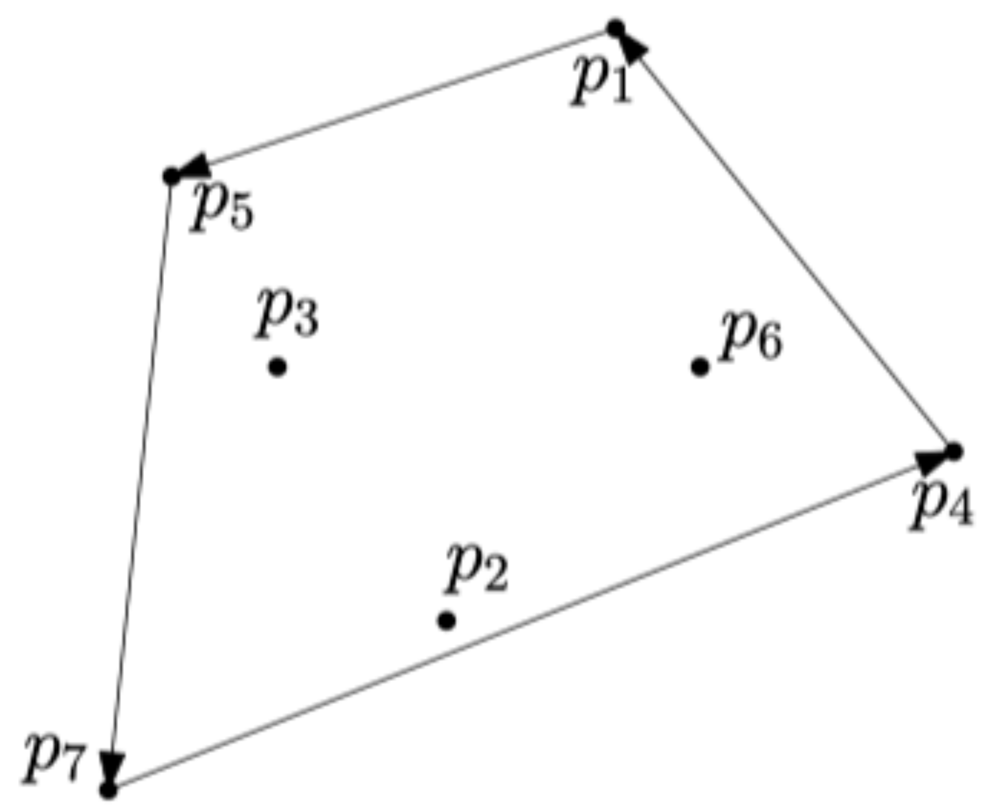
Konvexe Hülle: Darstellung



Punktemenge P,



konvexe Hülle $\mathrm{conv}(P)$,



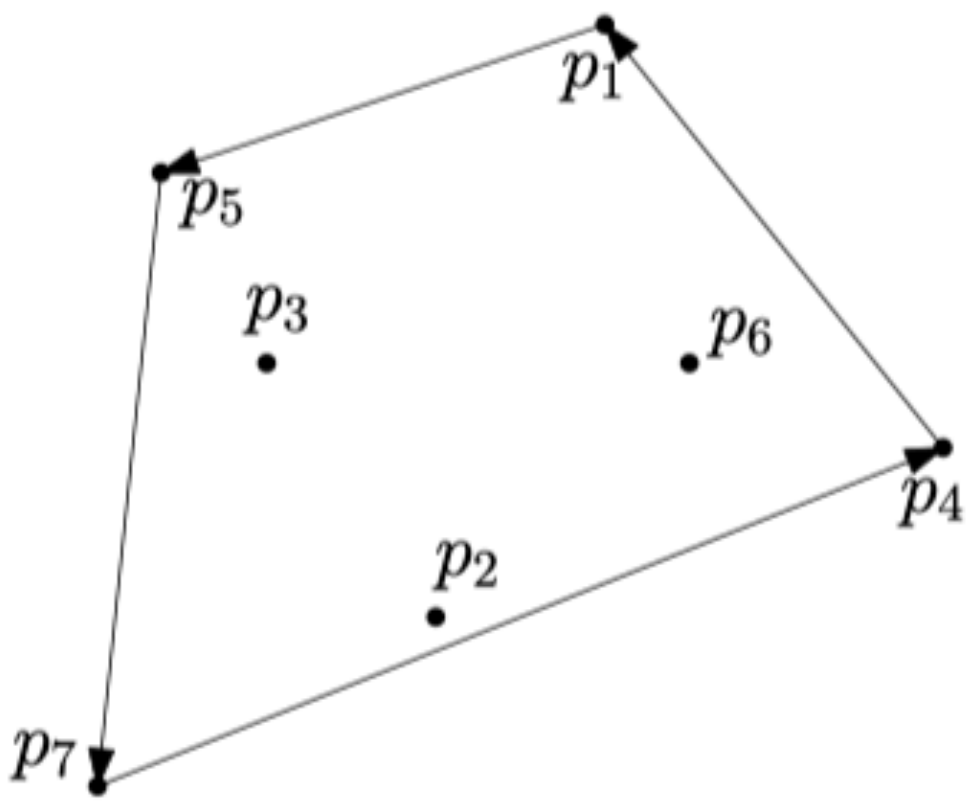
Polygon (p_4, p_1, p_5, p_7).

Vereinfachende Annahme: **Allgemeine Lage**, d.h. keine 3 Punkte auf einer gemeinsamen Geraden, keine 2 Pkt gleiche x-Koordinate.

Convex Hull

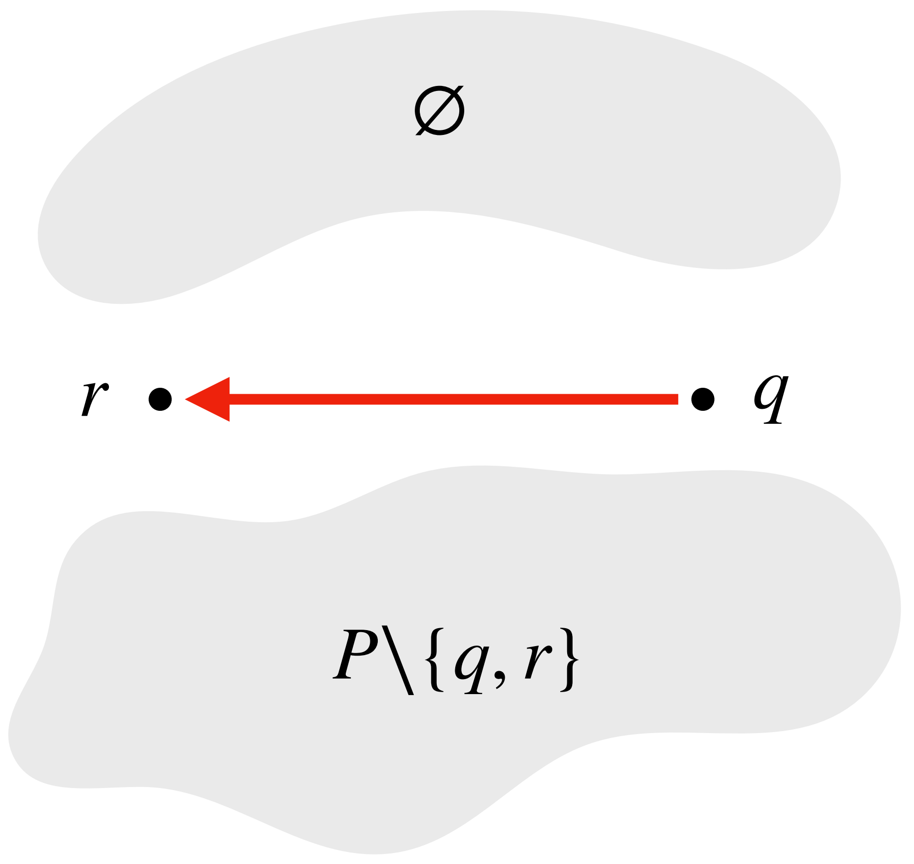
Gegeben: endliche Punktemenge $P \subseteq \mathbb{R}^2$.

Gesucht: die Ecken des $\mathrm{conv}(P)$ umrandenden Polygons, in der Reihenfolge gegen den Uhrzeigersinn.

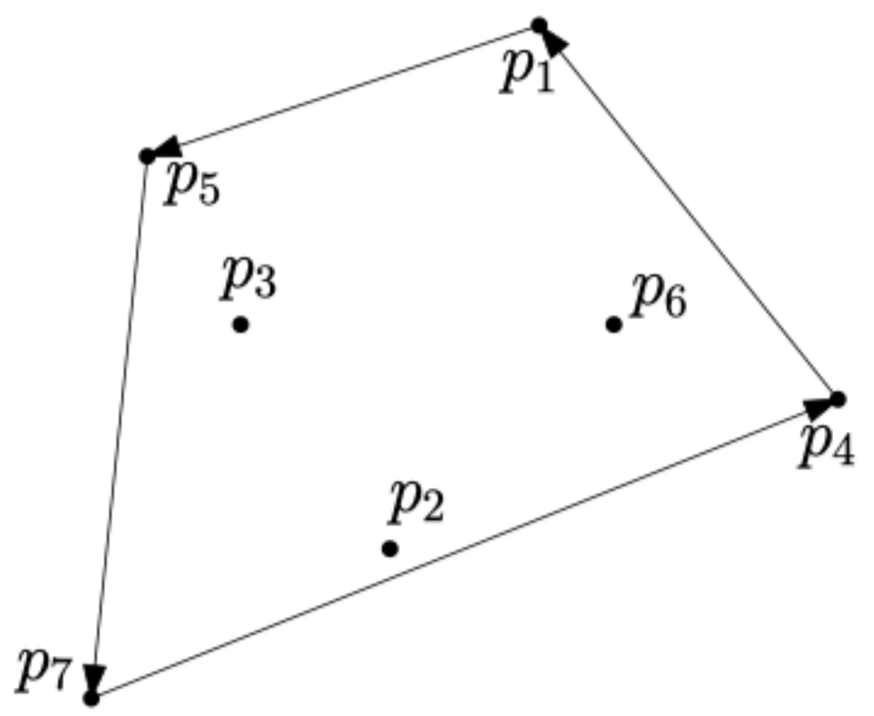


Polygon (p_4, p_1, p_5, p_7).

Randkanten



Restkanten: Alle Punkte links von qr.



Ein Paar $qr \in P^2$, $q \neq r$, heisst **Randkante** von P, falls alle Punkte in $P \setminus \{q, r\}$ links von qr liegen, d.h. auf der linken Seite der gerichteten Geraden durch q und r, gerichtet von q nach r, liegen.

Lemma

$(q_0, q_1, \ldots, q_{h-1})$ ist die Eckenfolge des $\mathrm{conv}(P)$ umschliessenden Polygons gegen den Uhrzeigersinn genau dann wenn alle Paare (q_{i-1}, q_i), $i = 1, 2, \ldots, h$, Randkanten von P sind (Indizes $\mathrm{mod}\, h$).

Jarvis Wrap

JarvisWrap(P)

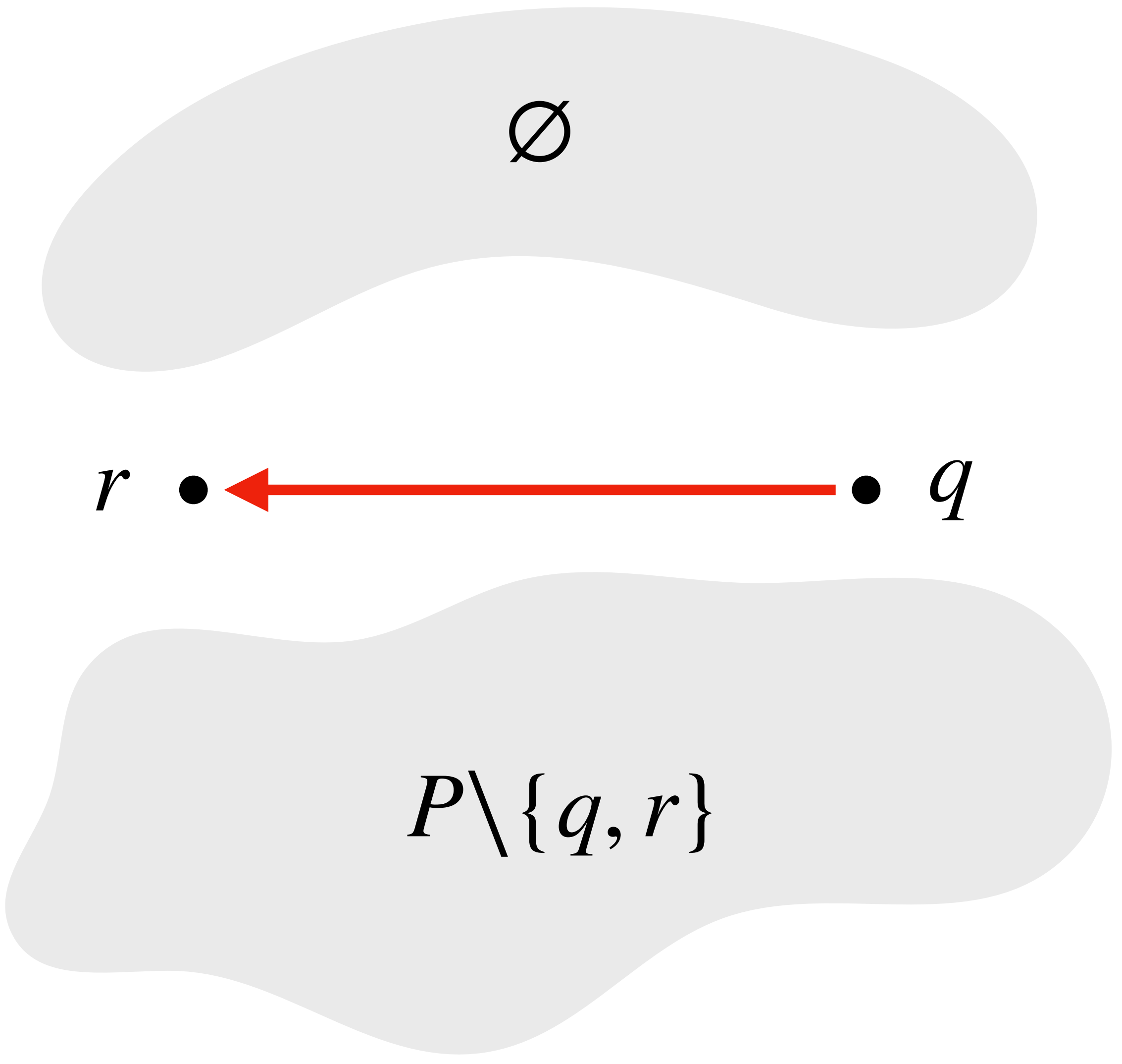
$h \leftarrow 0$
$p_{\text{now}} \leftarrow$ Punkt in P mit kleinster x-Koordinate
repeat
    $q_h \leftarrow p_{\text{now}}$
    $p_{\text{now}} \leftarrow$ FindNext(q_h)
    $h \leftarrow h + 1$
until $p_{\text{now}} = q_0$
return $(q_0, q_1, \ldots, q_{h-1})$

FindNext(q)

Wähle $p_0 \in P \setminus \{q\}$ beliebig
$q_{\text{next}} \leftarrow p_0$
for all $p \in P \setminus \{q, p_0\}$ **do**
    if p rechts von qq_{next} **then**
        $q_{\text{next}} \leftarrow p$
return q_{next}

Jarvis Wrap

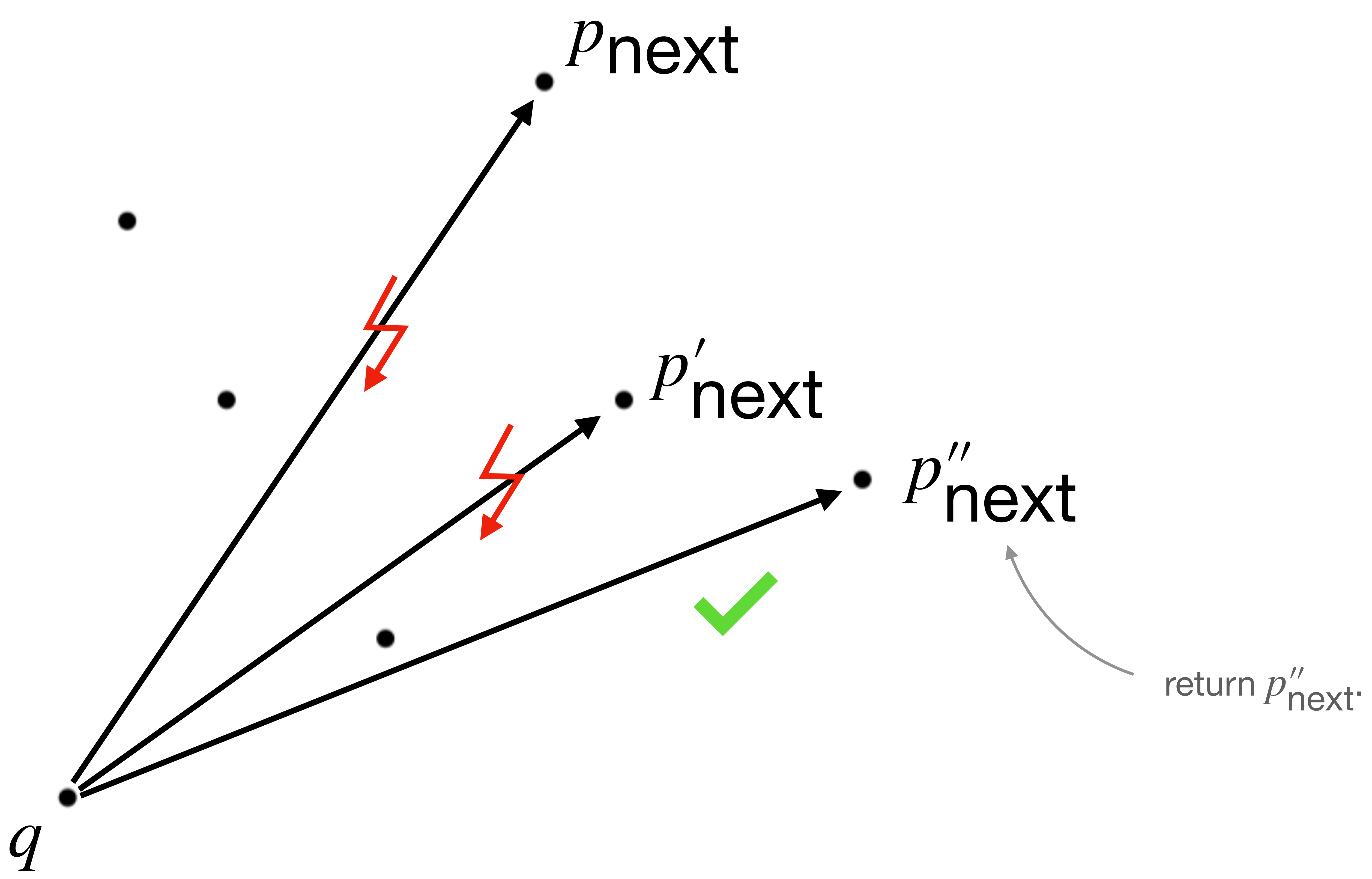
Lemma: Seiten des Polygons sind Restkanten.



Restkanten: Alle Punkte links von qr.

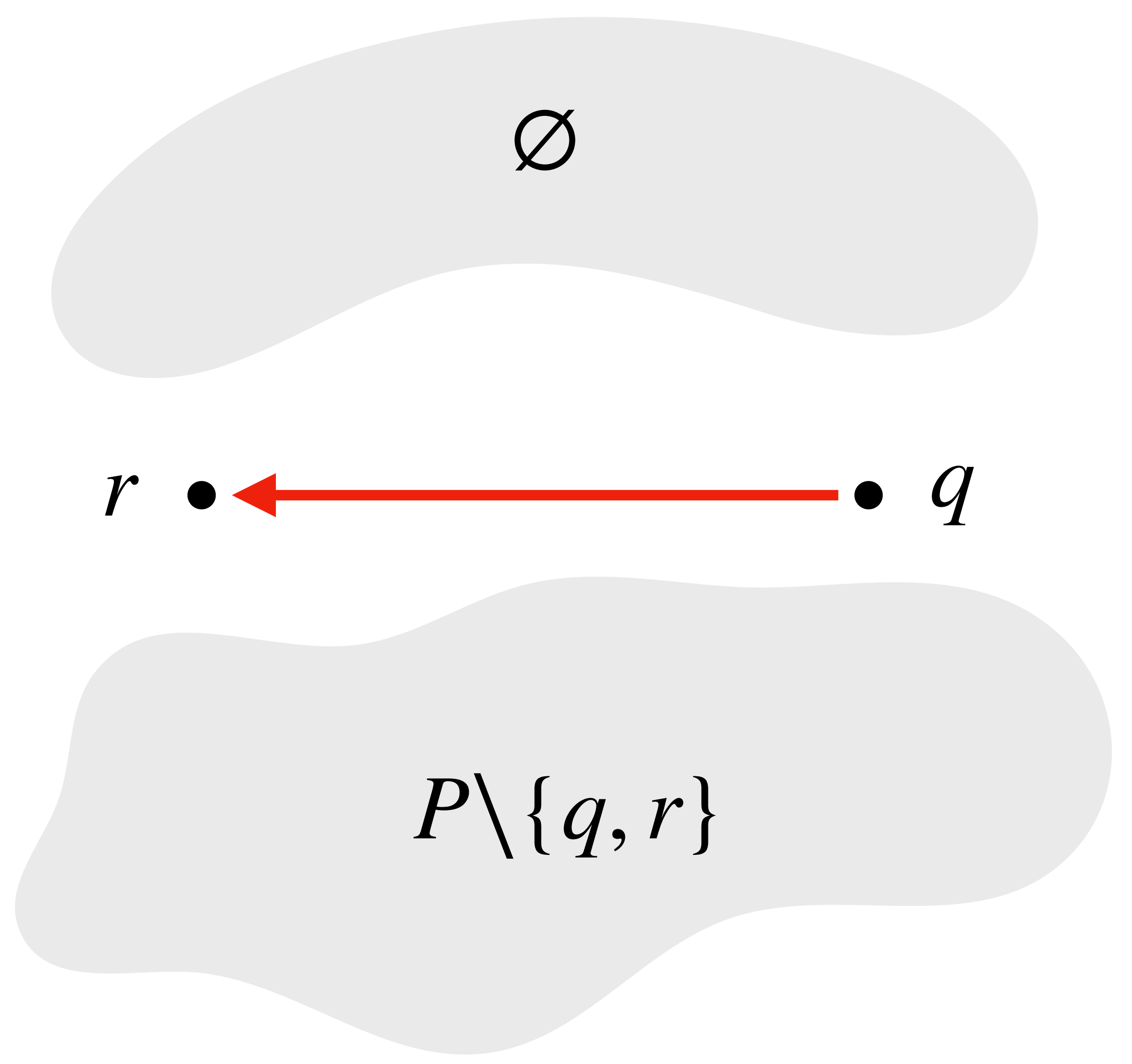
FindNext(q)

Wähle $p_0 \in P \setminus \{q\}$ beliebig
$q_{\text{next}} \leftarrow p_0$
for all $p \in P \setminus \{q, p_0\}$ **do**
  if p rechts von qq_{next} **then**
    $q_{\text{next}} \leftarrow p$
return q_{next}



Jarvis Wrap

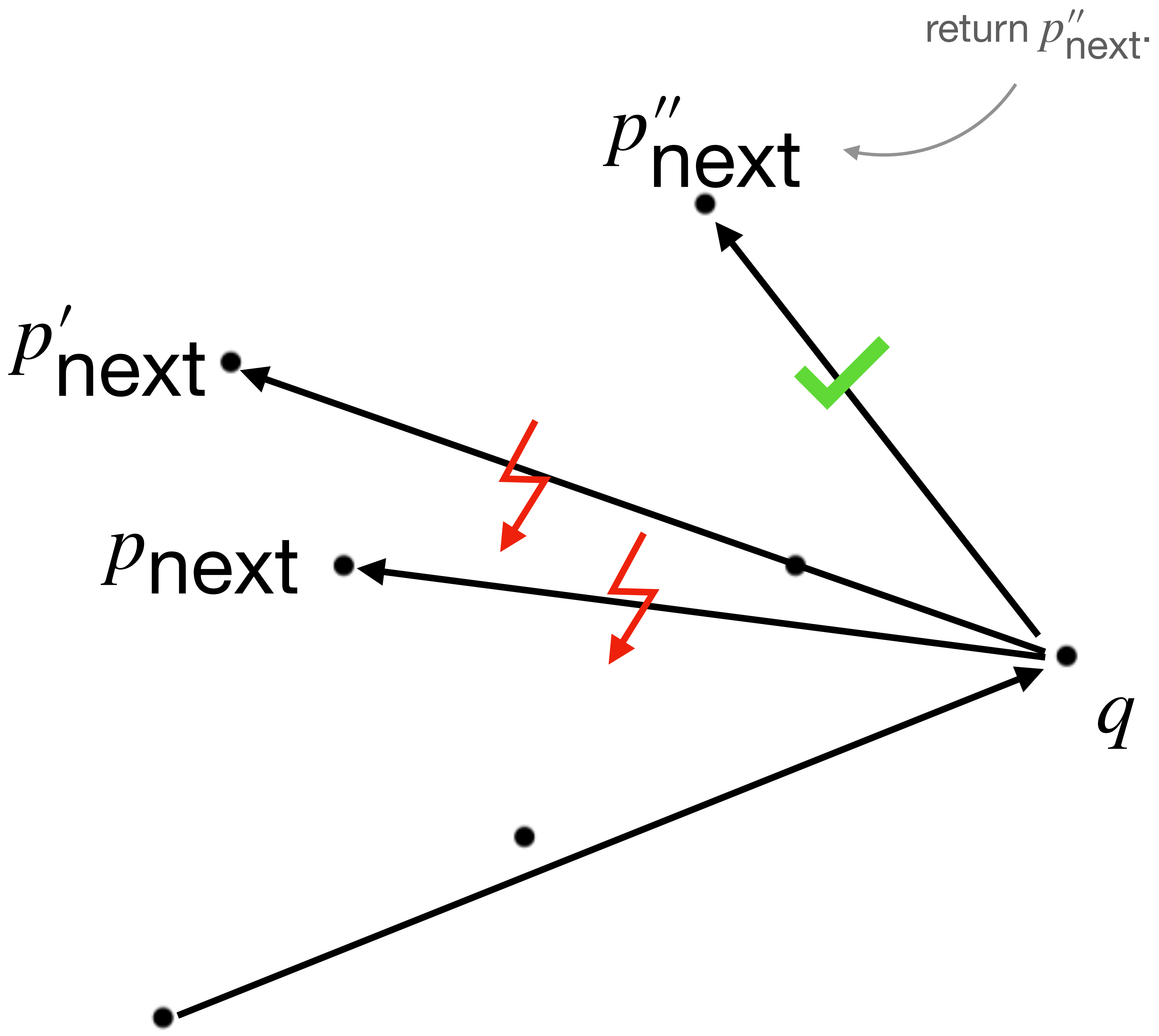
Lemma: Seiten des Polygons sind Restkanten.



Restkanten: Alle Punkte links von qr.

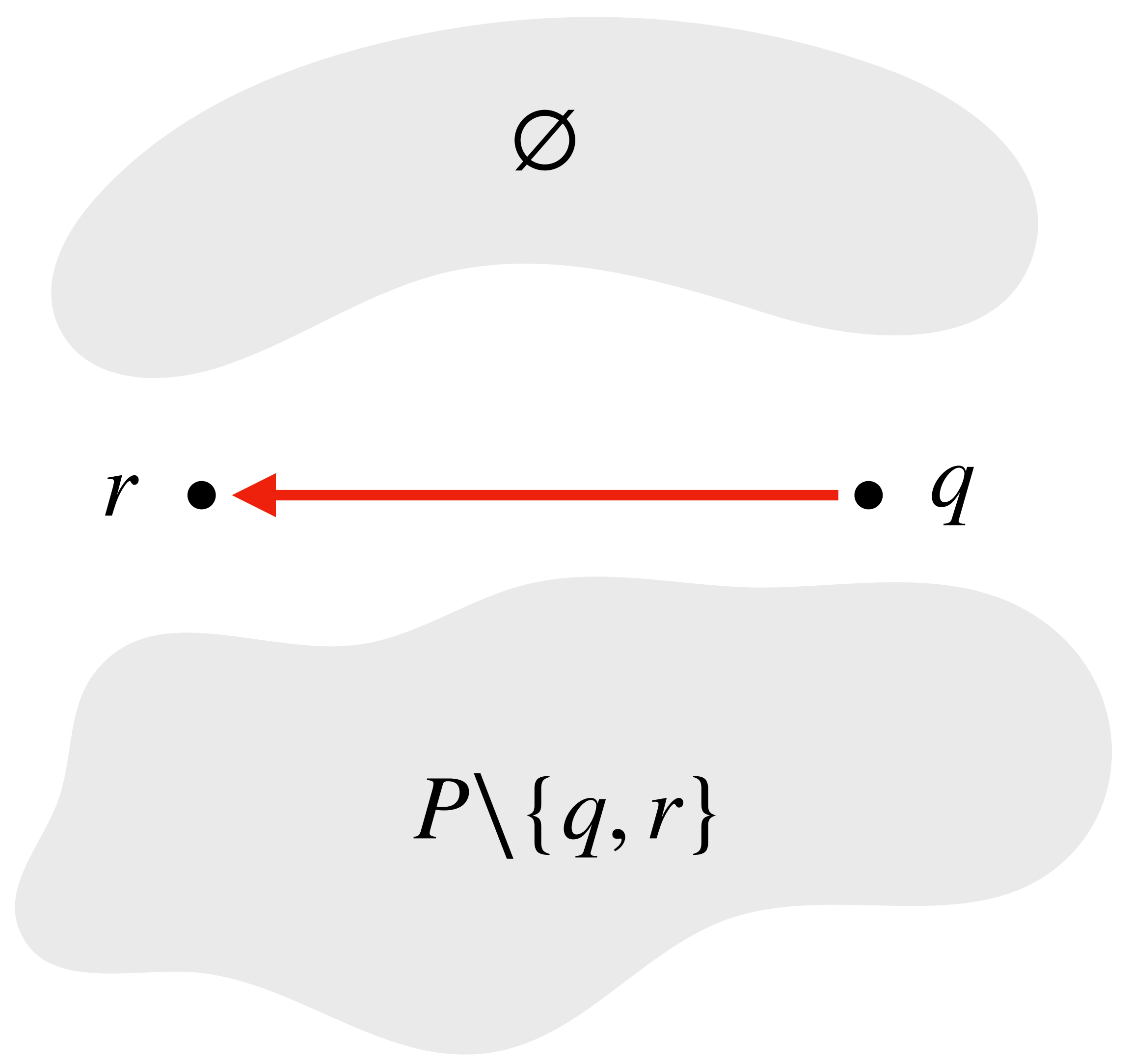
FindNext(q)

- Wähle $p_0 \in P \setminus \{q\}$ beliebig
- $q_{\text{next}} \leftarrow p_0$
- **for all** $p \in P \setminus \{q, p_0\}$ **do**
  - **if** p rechts von qq_{next} **then**
    - $q_{\text{next}} \leftarrow p$
- **return** q_{next}



Jarvis Wrap

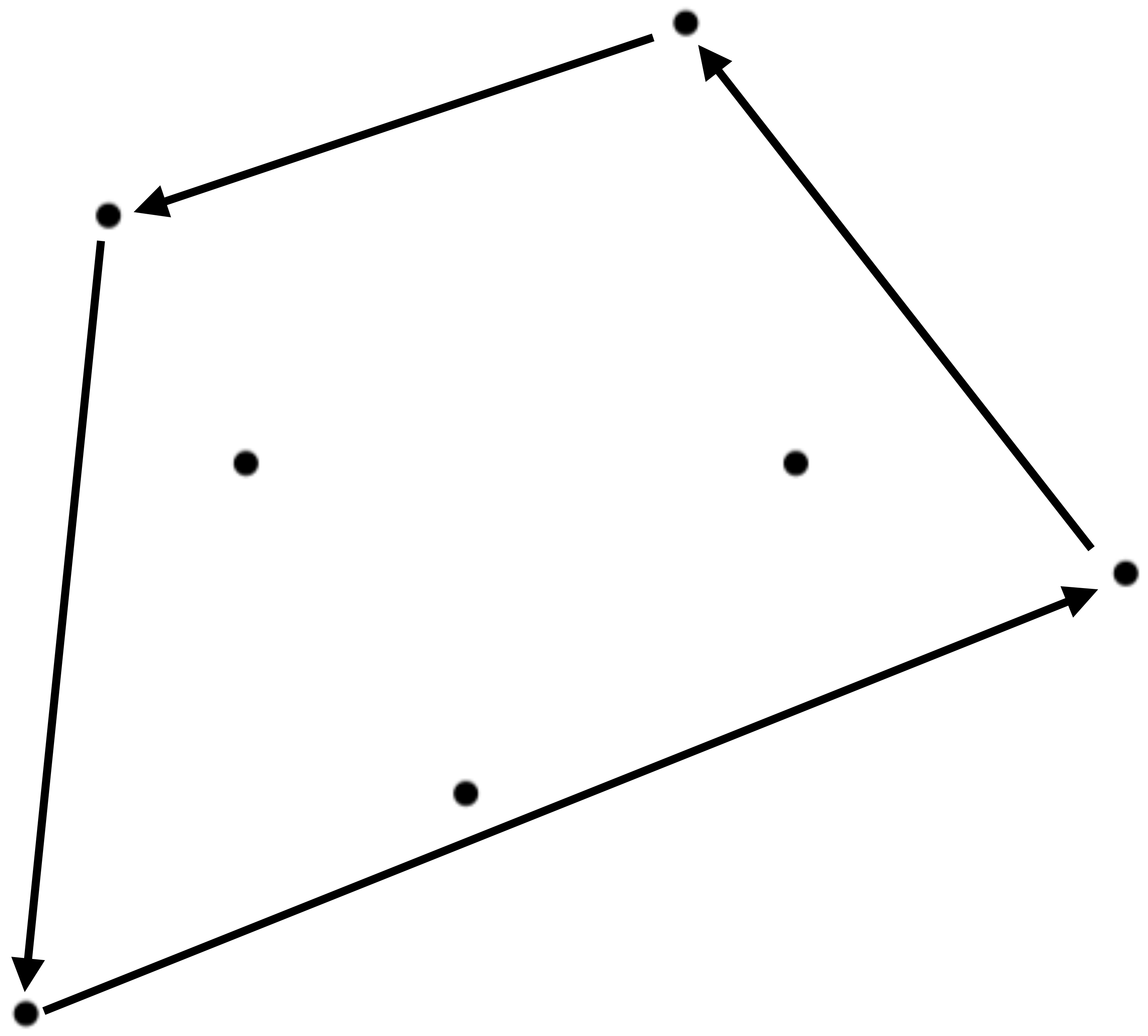
Lemma: Seiten des Polygons sind Restkanten.



Restkanten: Alle Punkte links von qr.

FindNext(q)

- Wähle $p_0 \in P \setminus \{q\}$ beliebig
- $q_{\text{next}} \leftarrow p_0$
- **for all** $p \in P \setminus \{q, p_0\}$ **do**
  - **if** p rechts von qq_{next} **then**
    - $q_{\text{next}} \leftarrow p$
- **return** q_{next}



Laufzeit

```
JarvisWrap(P)
h ← 0
p_now ← Punkt in P mit kleinster x-Koordinate
repeat
    q_h ← p_now
    p_now ← FindNext(q_h)
    h ← h + 1
until p_now = q_0
return (q_0, q_1, ..., q_{h-1})
```

```
FindNext(q)
Wähle p_0 ∈ P \ {q} beliebig
q_next ← p_0
for all p ∈ P \ {q, p_0} do
    if p rechts von q q_next then
        q_next ← p
return q_next
```

Jeder Aufruf der Funktion FindNext benötigt $O(n)$ Zeit, und wir rufen diese genau h-mal auf.

wobei h die Anzahl an Seiten des Polygons.

Satz 3.37. Gegeben eine Menge P von n Punkten in allgemeiner Lage in $\mathbb{R}^2$, berechnet der Algorithmus JarvisWrap die konvexe Hülle in Zeit $O(nh)$, wobei h die Anzahl der Ecken der konvexen Hülle von P ist.

Sei P eine endiche Punktemenge in allgemeiner Lage mit $|P| = n$. Dann ist es möglich in $O(n)$ Zeit zu prüfen, ob $\operatorname{conv}(P)$ ein Dreieck ist.

Satz 3.37. Gegeben eine Menge P von n Punkten in allgemeiner Lage in $\mathbb{R}^2$, berechnet der Algorithmus JARVISWRAP die konvexe Hülle in Zeit $O(nh)$, wobei h die Anzahl der Ecken der konvexen Hülle von P ist.

Sei P eine endliche Menge von Punkten in allgemeiner Lage und sei p der Punkt mit der grössten x-Koordinate. Dann ist p eine Ecke der konvexen Hülle von P.