AnW – Woche 10

02.05.24
Nicolas Wehrli

Heute:

I. Minitest
II. Kurze Kommentare zur Theorieaufgabe 4
    Quizaufgabe
III. Theory Recap
    - Jarvis Wrap
    - Local Repair
IV. Aufgabe

I. Minitest

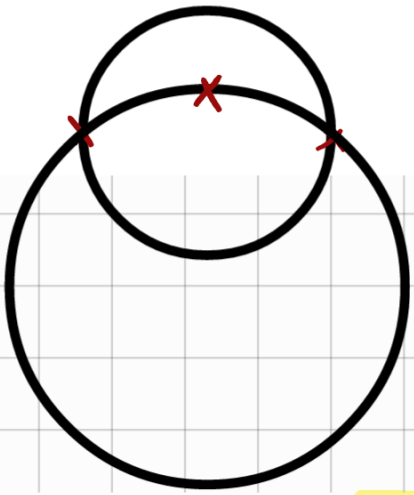
Passwort: enhance

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.

Bitte wählen Sie eine Antwort:

Wahr ○

Falsch ⊗

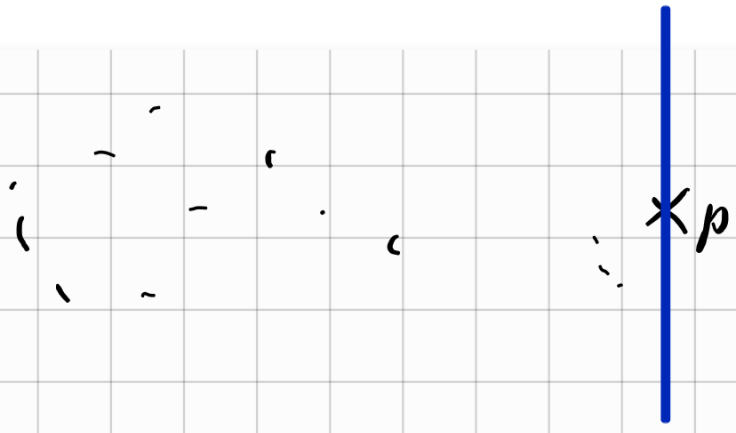


Sei P eine endliche Menge von Punkten in allgemeiner Lage und sei p der Punkt mit der grössten x-Koordinate. Dann ist p eine Ecke der konvexen Hülle von P.

Bitte wählen Sie eine Antwort:

Wahr ⊗

Falsch ○



Da keine 2 Punkte die gleiche x-Koordinate haben können, muss p eine Ecke sein, sonst wäre p nicht in der konvexen Hülle. ↯

Für jede endliche Punktemenge P gilt, dass $x \in \text{conv}(P)$ für alle $x \in P$. Bitte wählen Sie eine Antwort:

Wahr ⊗

Falsch ○

Aus der Definition von $\text{conv}(P)$ folgt $P \subseteq \text{conv}(P)$.

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

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

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.

Bitte wählen Sie eine Antwort:

Wahr ○

Falsch ✗

Randomised_PrimitiveVersion(P)

```
1: repeat forever
2:     wähle Q ⊆ P mit |Q| = 3 zufällig und gleichverteilt
3:     bestimme C(Q)
4:     if P ⊆ C•(Q) then
5:         return C(Q)
```

erwartete Laufzeit $O(n^4)$ (nicht schneller als CompleteEnumeration)

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?

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

Reihenfolge wird gespeichert ✗

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

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

Muss man auswendig können.
Von den Notizen Woche 09:

Für $v \in V$ und $i \in \mathbb{N}_0$: $DP[v][i] = P_i(v) =$

$$\left\{ S \in \binom{[k]}{i+1} \mid \exists \text{ ein mit } S \text{ gefärbter bunter Pfad der in } v \text{ endet} \right\}$$

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

Bitte wählen Sie eine Antwort:

- Wahr ⊗
- Falsch ○

Eine Iteration von JarvisWrap läuft in $O(n)$ und liefert den nächsten Ecken von $\mathrm{conv}(P)$.

Wir können die ersten 3 Iterationen von JarvisWrap laufen lassen. (Wenn es nach 3 Iterationen noch nicht fertig ist, wissen wir dass es kein Dreieck ist.) $\Longrightarrow 3 \cdot O(n) = O(n)$

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'

True	False	
○	☒	Die Ausgabe 'Primzahl' ist immer richtig
☒	○	Die Ausgabe 'keine Primzahl' ist immer richtig

$ggT(a,n) > 1$, $a^{n-1} \not\equiv_n 1$ sind Zertifikate für "keine Primzahl"

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

Berechne $Var(X-Y)$.

$$Var(X-Y) = Var(X + \underbrace{(-Y)}_{Z}) \overset{X \text{ und } Z=-Y \text{ unabh.}}{=} Var(X) + Var(-Y)$$

$$= Var(X) + (-1)^2 \cdot Var(Y)$$

$$= 2 + 4 = 6$$

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.

Richtig	Falsch		
☑	☐	$\mathbb{E}[X] = 100\|U\|/\|S\|$	✔
☐	☑	Falls $X = 100$, dann $\|S\| = \|U\|$.	✔
☑	☐	Falls $\|S\| = \|U\|$, dann $X = 100$.	✔
☐	☑	X ist geometrisch verteilt.	✔

- X ist negativ binomialverteilt mit $n=100$ und $p = \frac{|S|}{|U|}$.

1. $\mathbb{E}[X] = \frac{n}{p} = 100 \cdot \frac{|U|}{|S|}$
2. Falsch, kann Zufall sein.
3. $|S|=|U| \Rightarrow p = \frac{|S|}{|U|} = 1 \Rightarrow Pr[X=100] = 1$
4. Nein, es ist die Summe von 100 geometrisch verteilten Zufallsvariablen.

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

Wahr **Falsch**

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

$$\begin{aligned}\Pr[(A \cup B) \cap C] &= \Pr[(A \cap C) \cup (B \cap C)] \\ &= \Pr[A \cap C] + \Pr[B \cap C] - \Pr[A \cap B \cap C] \\ &= \Pr[C] \cdot (\Pr[A] + \Pr[B] - \Pr[A \cap B]) = \Pr[A \cup B] \cdot \Pr[C],\end{aligned}$$

☑ ☐ Falls $Pr[A \cup B] = \Pr[A] + \Pr[B]$, dann $Pr[A \cap B] = 0$. ✔

Folgt aus der Siebformel.

☑ ☐ Falls $\Pr[A \cap B] = Pr[A] \cdot Pr[B]$, dann sind A und B unabhängig. ✔

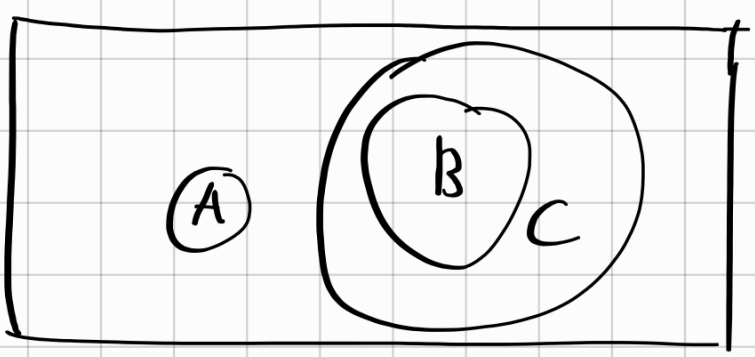
Per Definition:

Definition 2.18. Die Ereignisse A und B heissen *unabhängig*, wenn gilt

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

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

Intuitiv:



I. Kurze Kommentare zu T4

Generell sehr gut gelöst. ~

Definiert die Zufallsvariablen, die ihr verwendet!

Rechnet nicht einfach sondern **schreibt hin was ihr macht!**
Verhindert einfache Fehler.

↳ $\mathbb{E}(X) = \frac{k}{2}$ gibt keine Punkte!

Bsp:

(d) Nimm an, dass $k = 1000 \log_2 n$ und $n \geq 2$. Zeige, dass mit Wahrscheinlichkeit mindesten 0.99 alle Spaziergänge ein Vergnügen sein werden.

Sei X = "Anzahl Strassen mit Blumen".

$$\Pr\left[X \geq \mathbb{E}[X] + \frac{k}{4}\right] \leq \dots$$
$$\vdots$$
$$\leq n^{-40}$$

1. Abschätzung pro Spaziergang

Sei $Z = Z_1 + \dots + Z_n$ die Anzahl Spaziergänge die kein Vergnügen sind. Wobei $Z_i \sim \text{Ber}(p)$ mit $p \leq n^{-40}$

$$\mathbb{E}[Z] = n \cdot n^{-40} = n^{-39}$$ ← noch nicht das Resultat!

$$\Pr[Z \geq 1] \overset{\text{Markov}}{\leq} \frac{\mathbb{E}[Z]}{1} = n^{-39} \leq 2^{-39} < 0.01$$

$$\Rightarrow 1 - \Pr[Z \geq 1] \geq 0.99$$

$a \leq b \not\Rightarrow 1-a \leq 1-b$

$a \leq b \wedge c \leq d \not\Rightarrow \frac{a}{c} \leq \frac{b}{d}$

$\mathbb{P}(Z=0) \geq 1 - n \cdot e^{-40} \neq 1 - 2 \cdot e^{-40}$

$X = \sum_{i=1}^{n} X_i \sim Bin(n,p)$

$\Leftrightarrow \forall i \in [n]: X_i \sim Ber(p)$

und X_i unabhängig !

Quizaufgabe

Es sei $X = X_1 + \cdots + X_n$ die Summe von n unabhängigen Bernoulli-verteilten Zufallsvariablen mit Parameter p. Dann gilt für alle $\delta \in [0,1]$, dass

$\Pr[|X - np| \geq \delta np] \leq 2e^{-\delta^2 np/3}$.

Da $X_i \sim Ber(p)$ und unabh. gilt $X \sim Bin(n,p)$ und somit

alles was rot ist, müsst ihr nicht hinschreiben dient zur Erklärung

$\mathbb{E}(X) = n \cdot p$ per Binomialverteilung.

$\Rightarrow \Pr(|X - np| \geq \delta np) = \Pr(|X - \mathbb{E}(X)| \geq \delta \mathbb{E}(X))$

Platzhalter für Ereignis E

$E = \{\omega \in \Omega \mid |X(\omega) - \underbrace{\mathbb{E}(X)}_{\text{Zahl}}| \geq \delta \underbrace{\mathbb{E}(X)}_{\text{Zahl}}\} \subseteq \Omega$

$$\Pr[E] = \Pr\left(\underbrace{'X - \mathbb{E}(X) \geq \delta\mathbb{E}(X)'}_{\text{Ereignis } A} \cup \underbrace{'X - \mathbb{E}(X) \leq -\delta\mathbb{E}(X)'}_{\text{Ereignis } B}\right)$$

$$E = \{\omega \in \Omega \mid |X(\omega) - \mathbb{E}(X)| \geq \delta\mathbb{E}(X)\}$$

$$= \{\omega \in \Omega \mid (X(\omega) - \mathbb{E}(X) \geq \delta\mathbb{E}(X)) \vee (X(\omega) - \mathbb{E}(X) \leq -\delta\mathbb{E}(X))\}$$

$$= \underbrace{\{\omega \in \Omega \mid X(\omega) - \mathbb{E}(X) \geq \delta\mathbb{E}(X)\}}_{A} \cup \underbrace{\{\omega \in \Omega \mid X(\omega) - \mathbb{E}(X) \leq -\delta\mathbb{E}(X)\}}_{B}$$

(Union Bound)

$$\leq \Pr[X - \mathbb{E}(X) \geq \delta\mathbb{E}(X)] + \Pr[X - \mathbb{E}(X) \leq -\delta\mathbb{E}(X)]$$

$$= \Pr[X \geq (1+\delta)\mathbb{E}(X)] + \Pr[X \leq (1-\delta)\mathbb{E}(X)]$$

Da X eine Summe von unabhängigen Bernoullivariablen ist und $\underline{0 \leq \delta \leq 1}$, können wir Chernoff anwenden.

Satz 2.70 (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 $t \geq 2e\mathbb{E}[X]$.

$$\leq e^{\underbrace{-\frac{1}{3}\delta^2\mathbb{E}(X)}_{a}} + e^{\underbrace{-\frac{1}{2}\delta^2\mathbb{E}(X)}_{b}}$$

$$x \leq y \Rightarrow e^x \leq e^y$$

$$-b \leq -a \Rightarrow e^{-b} \leq e^{-a}$$

$$\leq 2 \cdot e^{-\frac{1}{3}\delta^2\mathbb{E}(X)} = 2 \cdot e^{-\delta^2 \frac{np}{3}}$$

III. Theory Recap

Konvexe Hülle

ConvexHull-Problem. Gegeben eine endliche Punktemenge $P \subseteq \mathbb{R}^2$, bestimme die konvexe Hülle von P.

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

- Für $v_0, v_1 \in \mathbb{R}^d$ sei

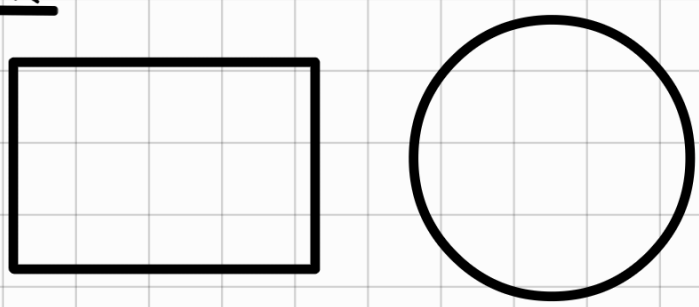
$$\overline{v_0 v_1} := \{(1-\lambda)v_0 + \lambda v_1 \mid \lambda \in \mathbb{R}, 0 \leq \lambda \leq 1\} ,$$

das v_0 und v_1 verbindende **Liniensegment**.

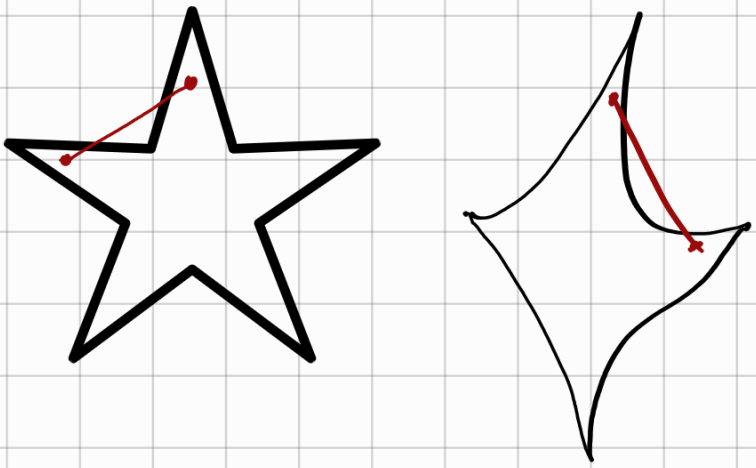
- Eine Menge $C \subseteq \mathbb{R}^d$ heisst **konvex**, falls

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

Beispiele in $\mathbb{R}^2$:



Konvex



nicht konvex

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

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

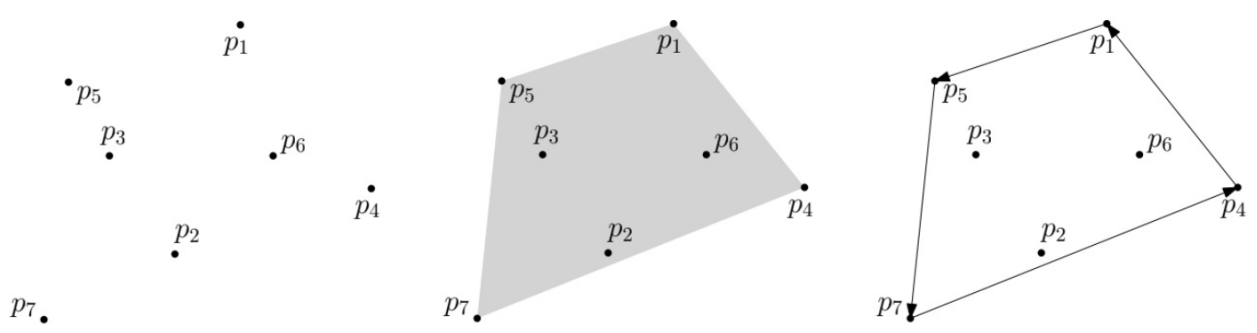
conv(S) = "kleinste konvexe Menge, die S enthält."

Wir beschränken uns auf $\mathbb{R}^2$ und endliche Punktemengen P.

Wir suchen die Folge $(q_0, \ldots, q_{h-1})$, $h \leq n$ der Ecken des Polygons, dass P begrenzt.

q_0 ist eine beliebige Ecke und der Rest folgt dem Gegenuhrzeigersinn.

Beachte: $Q := \{q_0, q_1, \ldots, q_{h-1}\} \subseteq P$ ist die kleinste Teilmenge von P mit $\text{conv}(Q) = \text{conv}(P)$.

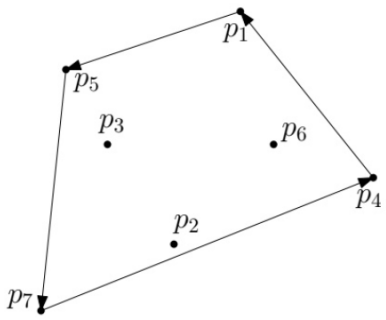


Punktemenge P, konvexe Hülle $\text{conv}(P)$, Polygon (p_4, p_1, p_5, p_7).

Vereinfachende Annahme für die Herleitung:

Allgemeine Lage: keine 3 Punkte auf derselben Geraden

und keine 2 Punkte haben die gleiche x-Koordinate.
(Behandeln wir später)



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 $\bmod h$).

Wir können mit ein wenig Lineare Algebra folgenden Test finden:

Lemma

Seien $p = (p_x, p_y)$, $q = (q_x, q_y)$, und $r = (r_x, r_y)$ Punkte in $\mathbb{R}^2$. Es gilt $q \neq r$ und p liegt links von qr genau dann wenn

$$\det(p, q, r) := \begin{vmatrix} p_x & p_y & 1 \\ q_x & q_y & 1 \\ r_x & r_y & 1 \end{vmatrix} = \begin{vmatrix} q_x - p_x & q_y - p_y \\ r_x - p_x & r_y - p_y \end{vmatrix} > 0$$

$$\Leftrightarrow \quad (q_x - p_x)(r_y - p_y) > (q_y - p_y)(r_x - p_x)$$

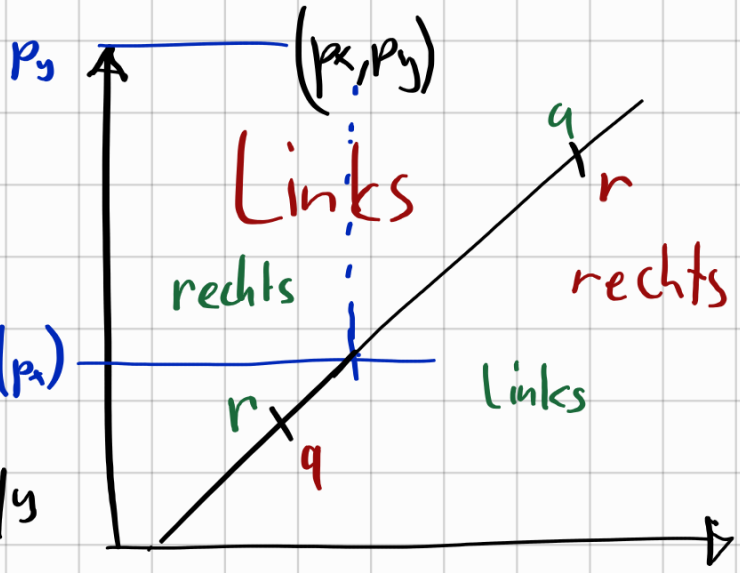
p links von qr ?

$\det(p, q, r) = 0$ zeigt an, dass die drei Punkte auf einer gemeinsamen Gerade liegen.

Intuition: $(r_x - q_x) > 0$

$(r_x - q_x) < 0$

Funktion für Gerade durch qr



$$f(x) = \left(\frac{r_y - q_y}{r_x - q_x}\right) \cdot (x - q_x) + q_y$$

(p_x, p_y) links von qr $\iff f(p_x) \cdot (r_x - q_x) < p_y \cdot (r_x - q_x)$

Den naiven Ansatz ($O(n^3)$) skippen wir.

Einwickeln (Jarvis Wrap)

$q_0 :=$ Punkt mit kleinster x-Koordinate in P.

q_0 ist sicher eine Ecke der konvexen Hülle, also Teil der gesuchten Folge und wir können insbesondere die Folge auch mit q_0 beginnen.

Gegeben $q \in P$, sei $\prec_q$ eine Relation auf $P \setminus \{q\}$ mittels

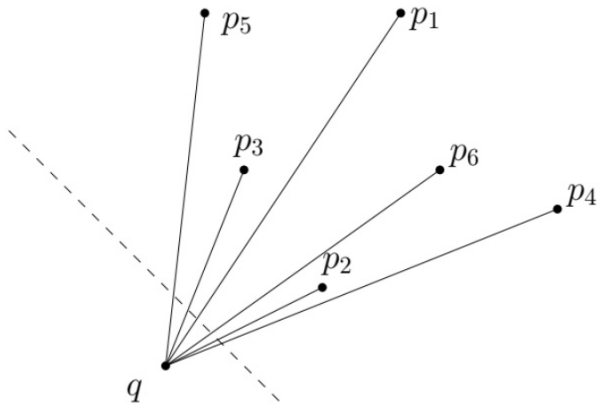
$$p_1 \prec_q p_2 \quad :\Leftrightarrow \quad p_1 \text{ rechts von } qp_2$$

rechts (nicht links wie oben!) $\Rightarrow$ Test: $\det(p_x, q, p_2) < 0$

Lemma

Ist q eine Ecke der konvexen Hülle von P, so ist die Relation $\prec_q$ eine totale Ordnung auf $P \setminus \{q\}$. Für das Minimum $p_{\min}$ dieser Ordnung gilt, dass $qp_{\min}$ eine Randkante ist.

Beispiel:



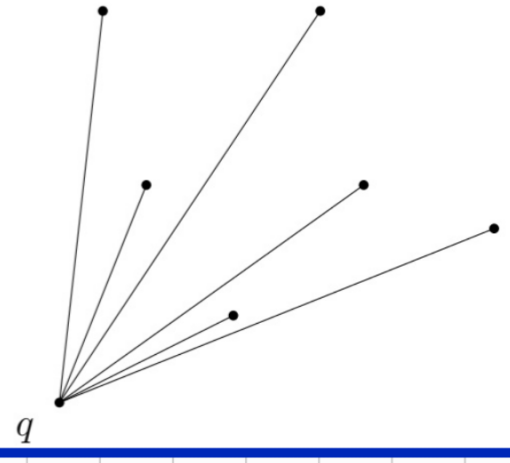
$p_4 \prec_q p_2 \prec_q p_6 \prec_q p_1 \prec_q p_3 \prec_q p_5$

qp_4 ist Randkante.

Nun können wir von einer Ecke ausgehend, die nächste Ecke bestimmen.

FindNext(q)

1: Wähle $p_0 \in P \setminus \{q\}$ beliebig
2: $q_{\text{next}} \leftarrow p_0$
3: **for all** $p \in P \setminus \{q, p_0\}$ **do** O(n)
4: **if** p rechts von qq_{next} **then**
5: $q_{\text{next}} \leftarrow p$
6: **return** q_{next}



Mit dem können wir von q_0 aus das ganze Polygon bestimmen.

JarvisWrap(P)

1: $h \leftarrow 0$
2: $p_{\text{now}} \leftarrow$ Punkt in P mit kleinster x-Koordinate
3: **repeat**
4: $q_h \leftarrow p_{\text{now}}$
5: $p_{\text{now}} \leftarrow$ FindNext(q_h) O(n)
6: $h \leftarrow h + 1$
7: **until** $p_{\text{now}} = q_0$ ← O(h)-mal
8: **return** $(q_0, q_1, \ldots, q_{h-1})$

Analyse:

Satz
Für 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.*

- Da $h \leq n$, läuft JarvisWrap in $O(n^2)$ (statt $O(n^3)$).
- Ist $h = O(1)$, z.B. conv(P) ist ein Dreieck, läuft der Algorithmus in $O(n)$ Zeit.

Nun behandeln wir die vereinfachenden Annahmen.

Für Mengen P mit Kollinearitäten (mehr als 2 Punkte auf einer Gerade) oder mehreren Punkten gleicher x-Koordinate, ist folgendes zu beachten:

1. Anfangspunkt q_0 als den Punkt mit lexikographisch kleinster Koordinate (unter allen mit kleinster x-Koordinate, den mit kleinster y-Koordinate). (Adaption der x-Ordnung der Punkte)

2. Der Test "p rechts von qq_{next}" in FindNext(q) muss ersetzt werden durch (p rechts von qq_{next}) oder (p auf der Geraden durch qq_{next} und $|qp| > |qq_{next}|$):

$$(\det(p, q, q_{next}) < 0) \vee (\det(p, q, q_{next}) = 0 \wedge |qp| > |qq_{next}|) .$$

3. In der Regel können wir nicht einmal annehmen, dass die Punkte verschieden sind (z.B. gegeben in einem Feld)!

↳ Wir können einen Punkt mehrmals haben.

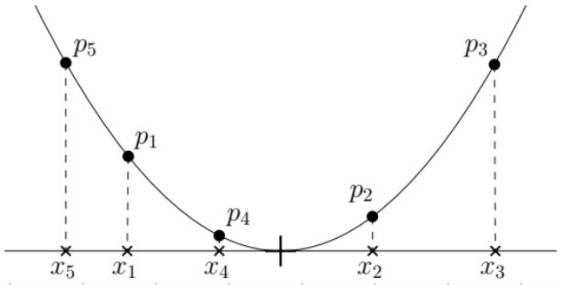
Mit Floating Point Values treten numerische Probleme auf. (Siehe NumCS nächstes Semester)

Es geht oft nicht um die absolute Genauigkeit des Ergebnisses (z.B. bei Eingaben, die selbst schon mit Fehlern behaftet sind). Vielmehr kann der Algorithmus völlig falsche Ergebnisse liefern oder in eine unendliche Schleife laufen.

Wegen diesen Ungenauigkeiten könnte z.B. der Algo am Startpunkt vorbeilaufen.

Untere Schranke durchs Sortieren

Betrachte eine Folge $(x_1, x_2, \ldots, x_n)$ von Zahlen in $\mathbb{R}$. Wir setzen $p_i = (x_i, x_i^2)$, $i = 1, 2, \ldots, n$ (vertikale Projektion von der x-Achse im $\mathbb{R}^2$ auf die Einheitsparabel $y = x^2$).



Aus der Folge der Ecken der konvexen Hülle von $P := \{p_1, p_2, \ldots, p_n\}$ ergibt sich (in linearer Zeit) auch die aufsteigend sortierte Reihenfolge der x_i's.

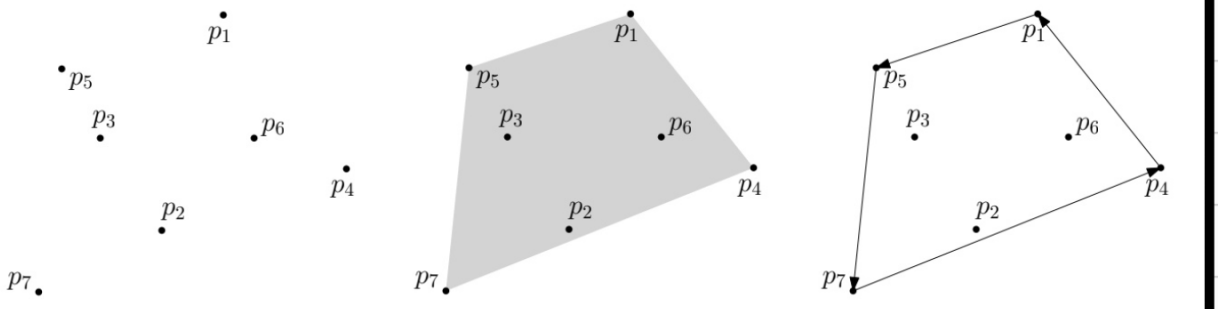
Wir haben eine sogenannte **Reduktion** gezeigt: Kann man ConvexHull in $t(n)$ lösen, so kann man in $t(n)+O(n)$ Zeit sortieren.

T1

Lokales Verbessern

Gleiche Problemstellung wie bei Jarvis Wrap:

ConvexHull-Problem. Gegeben eine endliche Punktemenge $P \subseteq \mathbb{R}^2$, bestimme die Ecken des $\text{conv}(P)$ umrandenden Polygons, in der Reihenfolge gegen den Uhrseigersinn.



Punktemenge P, konvexe Hülle $\text{conv}(P)$, Polygon (p_4, p_1, p_5, p_7).

Wir verwenden wieder die Annahme der Allgemeinen Lage. (keine 3 Punkte auf der gleichen Gerade, keine 2 Punkte mit der gleichen x-Koordinate.)

Bis jetzt haben wir immer folgende Bedingung überprüft, um die nächste Kante des Polygons zu finden.

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. $O(n)$

Dies kostet jedes Mal um den nächsten Eckpunkt zu finden $O(n)$.

Deshalb schlagen wir etwas neues vor:

Lokal konvex:

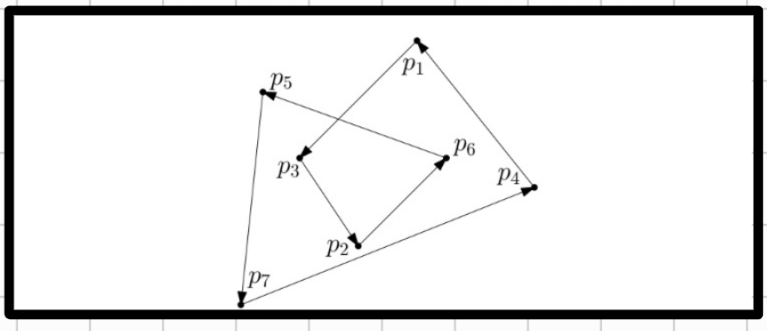
Lokale Prüfung in Polygon $(q_0, q_1, \ldots, q_{h-1})$.

$$\forall i, 1 \leq i \leq h: \; q_{i+1} \text{ links von } q_{i-1}q_i$$

$O(1)$

Diese Bedingung reicht aber nicht aus.
Sie hat ein paar Defizite.

- $\{q_0, q_1, \ldots, q_{h-1}\}$ muss nicht Teilmenge von P sein.
- Das Polygon muss nicht alle anderen Punkte im Inneren haben.
- Das Polygon kann sich selbst kreuzen.

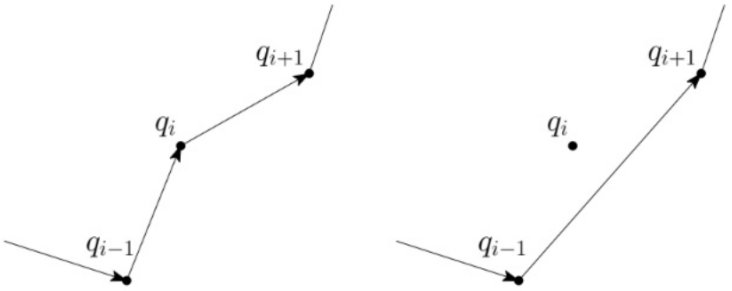


Wir können solche nicht-konvexe Polygone
Lokal Verbessern.

Gegeben $(q_0, q_1, \ldots, q_{k-1})$, falls

$$q_i \text{ links von } q_{i-1}q_{i+1} \text{ liegt}$$

dann entferne q_i aus der Folge.



Local-Repair Algorithmus

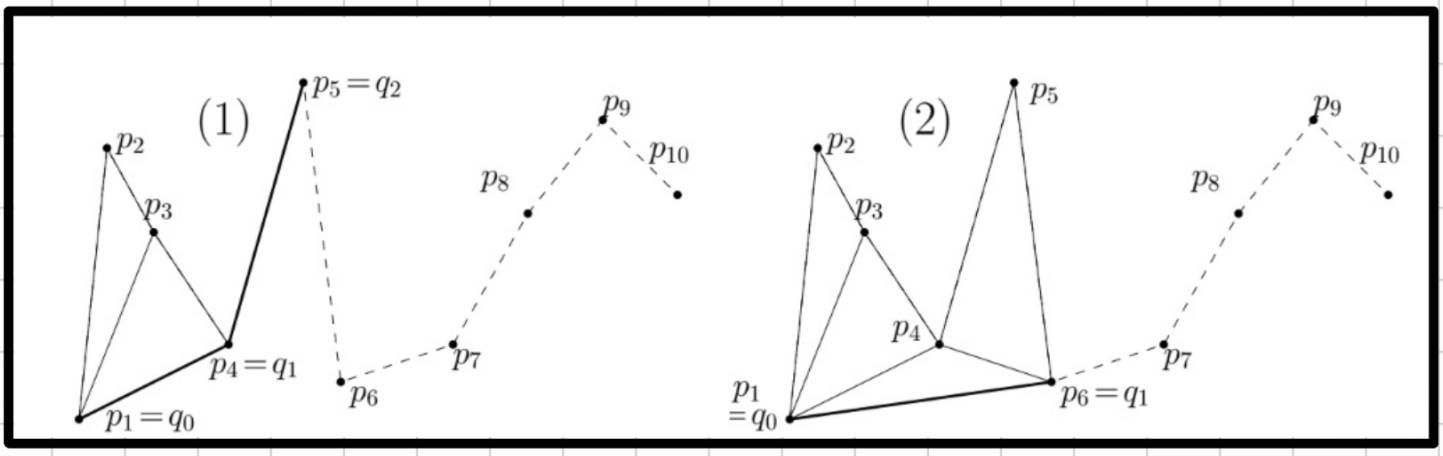
Sortiere P nach x-Koordinate (aufsteigend): $(p_1, \dots, p_n)$.

Dann betrachten wir $(p_1, \dots, p_n, p_{n-1}, \dots, p_2)$ als Startpolygon.

```
LocalRepair(p_1, p_2, ..., p_n)                    (p_1, p_2, ..., p_n) sortiert
 1: q_0 ← p_1; h ← 0
 2: for i ← 2 to n do                  ▷ unterer Rand, links nach rechts
 3:     while h > 0 und q_h links von q_{h-1}p_i do
 4:         h ← h − 1
 5:     h ← h + 1; q_h ← p_i
 6:             ▷ (q_0, ..., q_h) untere konvexe Hülle von {p_1, ..., p_i}
 7: h' ← h
 8: for i ← n − 1 downto 1 do          ▷ oberer Rand, rechts nach links
 9:     while h > h' und q_h links von q_{h-1}p_i do
10:         h ← h − 1
11:     h ← h + 1; q_h ← p_i
12: return (q_0, q_1, ..., q_{h-1})
```

Intuitiv: Nach der i-ten Iteration, ist $(q_0, \dots, q_h)$ die untere konvexe Hülle (Invariante) von $\{p_1, \dots, p_i\}$ wobei $q_h = p_i$.

In der nächsten Iteration erweitern wir diese Folge um p_{i+1} und verbessern die Folge lokal, in dem wir sie von q_h aus rückwärts durchgehen, bis wir die Tangente an $(q_1, \dots, q_h)$, die durch p_{i+1} geht, gefunden haben.



etc.

Analyse:

Wir beginnen mit einem Polygon mit $2(n-1)$ Ecken und haben am Ende h Ecken. Wir verbessern also genau $2(n-1)-h = O(n)$ Mal.

Es gibt also $O(n)$ erfolgreiche Tests „q_h links von $q_{h-1}p_i$“, und für jeden Punkt p_i zwei erfolglose (einmal unten, und einmal oben).

Der Algorithmus hat also nach dem anfänglichen Sortieren in $O(n \log n)$ eine Laufzeit von $O(n)$.

Satz 3.42. Gegeben eine Folge $p_1, p_2, \ldots, p_n$ nach x-Koordinate sortierter Punkte in allgemeiner Lage in $\mathbb{R}^2$, berechnet der Algorithmus LocalRepair die konvexe Hülle von $\{p_1, p_2, \ldots, p_n\}$ in Zeit $O(n)$.

Wir haben letztes Mal besprochen, dass Sortieren eine untere Schranke vom ConvexHull-Problem ist.

Da die Laufzeit von LocalRepair vom Sortieren dominiert wird, ist der Algorithmus optimal.

Weitere Kommentare:

- Annahme der 'Allgemeinen Lage'
  - → lexikographisch sortieren
  - → Test "q_n links von $q_{n-1}p_i$" anpassen.
  - → Duplikate nach sortieren entfernen.
- Numerisch robuster als JarvisWrap.
  - → kann nicht in einer unendlichen Schlaufe laufen.
- Liefert Triangulierungen als Webenprodukt.

Aufgaben

Aufgabe 1 – *Dreiecks Überdeckung*

Sei P eine endliche Menge von Punkten in der Ebene, die nicht alle auf einer Gerade liegen. Wir sagen, ein Dreieck überdeckt einen Punkt p, falls p im Innern des Dreiecks, auf dessen Kanten oder auf einem der Eckpunkte liegt. Wir nennen ein Dreieck *erlaubt*, wenn *alle* seine Eckpunkte Punkte in P sind.

Sei $\Delta(P)$ die minimale Anzahl erlaubter Dreiecke, sodass jeder Punkt in P von (mindestens) einem dieser Dreiecke überdeckt ist. Ziel dieser Aufgabe ist es, eine Approximation für $\Delta(P)$ zu finden.

(a) Sei $Q = (x_1, x_2, \dots x_n)$ eine Sequenz von n Punkten, die die Eckpunkte eines konvexen Polygons bilden. Das heisst, dass für jedes $i = 1, \dots, n$ alle Punkte ausser x_i und x_{i+1} links von der gerichteten Geraden $x_i x_{i+1}$ liegen (wobei wir x_{n+1} als x_1 interpretieren).
Bestimme den Wert von $\Delta(Q)$ in Abhängigkeit von n (mit Beweis).

Hinweis: Sie dürfen ohne Beweis annehmen, dass jedes erlaubte Dreieck keine Punkte von Q ausser dessen Eckpunkte überdeckt.

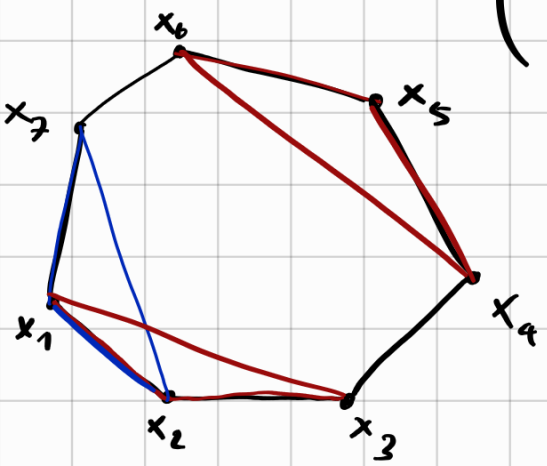
Per Hinweis überdeckt ein Dreieck maximal 3 Punkte in Q.

$$\Longrightarrow \Delta(Q) = \left\lceil \frac{n}{3} \right\rceil$$

Beispielsweise Dreiecke mit Eckpunkten

$\{x_1, x_2, x_3\}, \{x_4, x_5, x_6\}, \dots$ (Wiederverwendung von x_1, x_2 falls $n \bmod 3 \neq 0$)

(deshalb $\lceil \dots \rceil$)



(b) Sei P nun eine beliebige Punktmenge und sei k die Anzahl der Eckpunkte der konvexen Hülle von P. Beweise, dass $\Delta(P) \geq k/3$.

Sei $Q = \{x_1, x_2, \dots, x_k\} \subseteq P$ die Menge der Eckpunkte.

Wir wissen von (a) $\Delta(Q) \geq \frac{k}{3}$.

Wir zeigen nun $\Delta(P) \geq \Delta(Q)$.

Nehmen wir per Widerspruch $\Delta(P) < \Delta(Q)$ an.

$\Rightarrow$ Wir können mit Eckpunkten aus P mind. 1 Dreieck finden, dass mehr als 3 Punkte von Q überdeckt.

$\Rightarrow$ $\exists x_i \in Q$, so dass x_i kein Eckpunkt dieses Dreiecks ist.

$\Rightarrow$ Dieses x_i kann dann aber kein Eckpunkt der konvexen Hülle von P sein! ↯

$\longrightarrow$ Per Widerspruch folgt $\Delta(P) \geq \Delta(Q) \geq \frac{k}{3}$ ☐

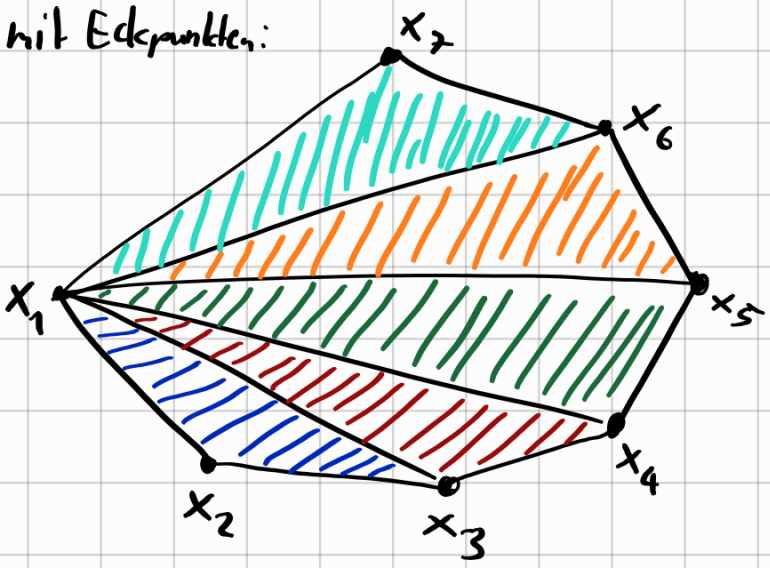
(c) Seien P und k wie in (b) definiert. Beweise $\Delta(P) < k$.

<u>Existenzbeweis:</u>

Wir zeigen, dass für jede Menge P $k-2$ Dreiecke gibt, die conv(P) überdecken.

Wir definieren die Dreiecke mit Eckpunkten:

d_1: $\{x_1, x_2, x_3\}$
d_2: $\{x_1, x_3, x_4\}$
$\vdots$ $\quad\quad$ $\vdots$
d_{k-2}: $\{x_1, x_{k-1}, x_k\}$



Sei $P(i) :=$ "Die Dreiecke $d_1, \dots, d_{i-2}$ überdecken die konvexe Hülle $\text{conv}(\{x_1, \dots, x_i\})$"

Wir zeigen $P(k)$ per Induktion.

Base Case:

Wir zeigen $P(3)$.

$d_1 = \text{conv}(\{x_1, x_2, x_3\})$ ✓

(identisch)

Induction Hypothesis:

For $u \in \{3, \dots, k-1\}$
Assume $P(u)$.

Induction Step: $u \to u+1$

Then show $P(u+1)$:

Wir betrachten die Kante (x_1, x_u).
Sei $p \in \text{conv}(\{x_1, \dots, x_{i+1}\})$ beliebig.

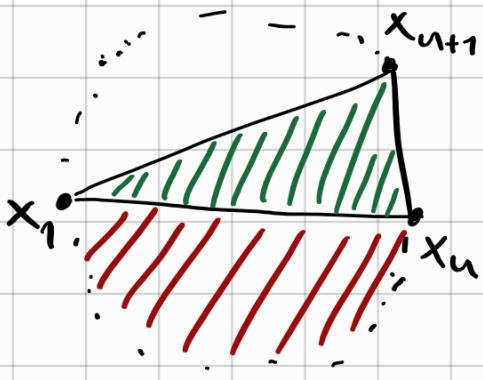
Case Distinction:

1. p rechts von (x_1, x_u) oder auf (x_1, x_u):

$\Rightarrow p \in \text{conv}(\{x_1, \dots, x_u\})$

$P(u) \Rightarrow$ p wird von $d_1, \dots, d_{u-2}$ überdeckt. (by I.H.)

2. p links von (x_1, x_u):



Da $p \in \text{conv}(\{x_1, \dots, x_{u+1}\})$ gilt auch:
- p links von (x_u, x_{u+1})
- p links von (x_{u+1}, x_1)

$\Rightarrow$ p wird von $d_{(u+1)-2}$ mit Eckpunkten $\{x_1, x_u, x_{u+1}\}$ überdeckt.

$\Rightarrow$ Jedes beliebige $p \in \text{conv}(\{x_1, \dots, x_{u+1}\})$ wird von den Dreiecken $d_1, \dots, d_{(u+1)-2}$ überdeckt.

$\Rightarrow P(u+1)$

$\square$

(d) Konstruiere einen Algorithmus, der eine 3-Approximation von $\Delta(P)$ findet. Zeige die Korrektheit des Algorithmus und gib seine Laufzeit (in Abhänigikeit von $|P|$) an.

Hinweis: Sie dürfen die Aussagen von (a) (b) und (c) für die Lösung von (d) verwenden, auch wenn sie diese Teilaufgaben nicht gelöst haben.

Dreieck (P):

```
Q ← ∅ // Liste für Eckpunkte der konvexen Hülle      O(1)
Q ← LocalRepair(P)                                  O(n log n)
k ← |Q|                                             O(1)
return k-2                                          O(1)
```

Laufzeit: $O(n \log n)$

Korrektheit:

(b) $\Longrightarrow \quad \Delta(P) \geq \frac{k}{3} \quad | \cdot 3$

$3 \cdot \Delta(P) \geq k$

$\Rightarrow \quad k-2 < 3 \cdot \Delta(P)$ ($k-2$ ist eine 3-Approx. von $\Delta(P)$)

Per (c) gibt es für alle P eine $k-2$ Dreiecke, die die konvexe Hülle $conv(P)$ überdecken.

□