# Vorlesung Algorithmische Kartografie

## Proportional Symbol Maps

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

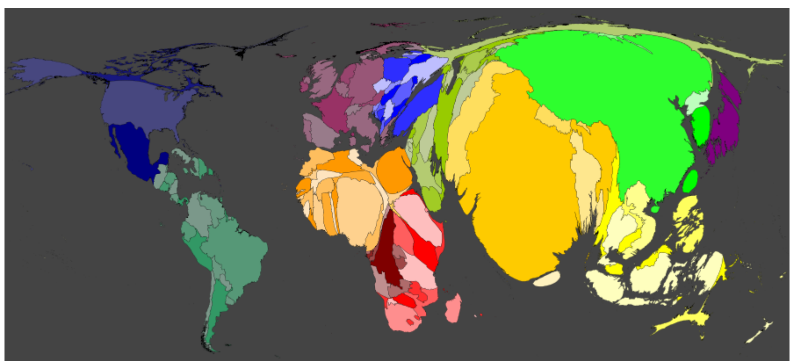
**Benjamin Niedermann** · Martin Nöllenburg

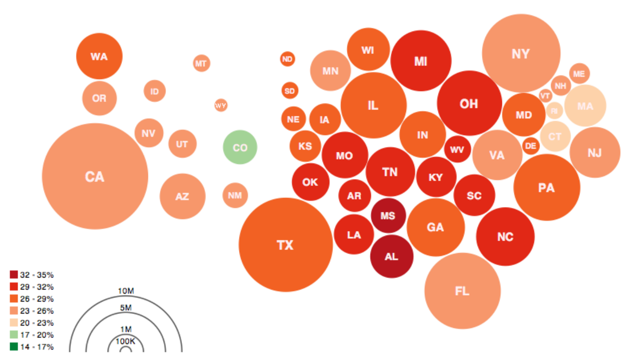
25.06.2015

# Statistische Visualisierung in Karten

**Thematische Karten** sind Kartendarstellungen bestimmter Themen, z.B. Bevölkerungsstruktur, Wirtschaftsdaten, Geologie, usw. Die Daten sind oft punkt- oder flächenbezogen und können über Farben, Größen, Label etc. visualisiert werden.

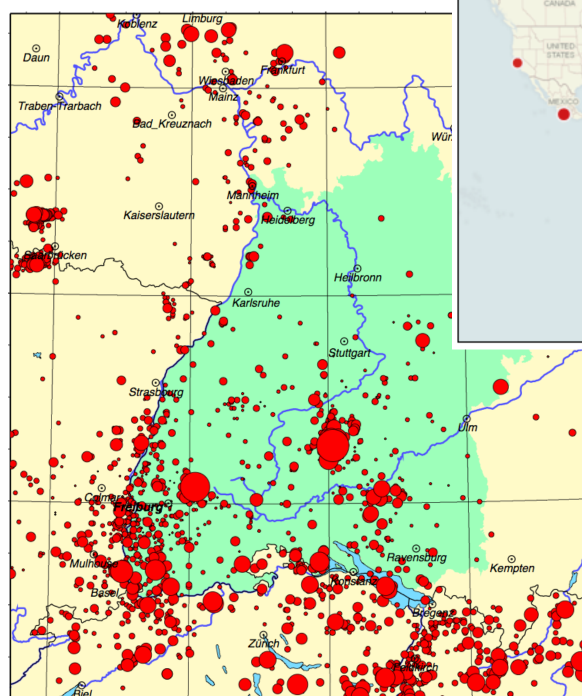
**Beispiele:**

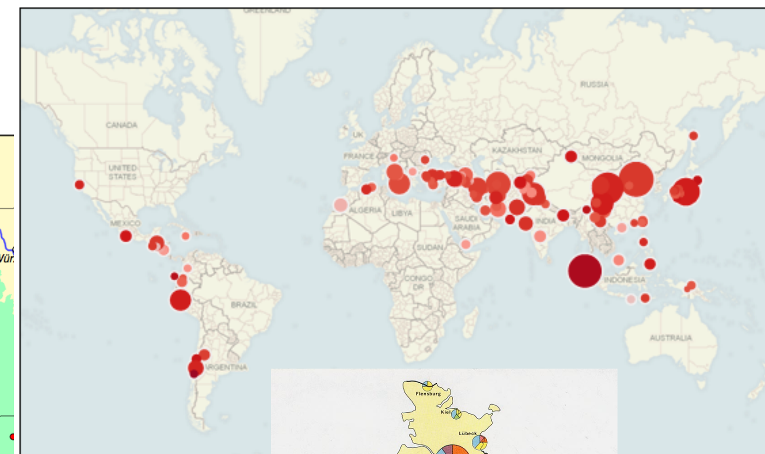


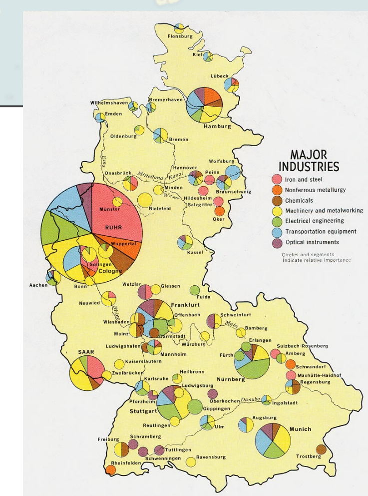



Flächendaten

Punktdaten

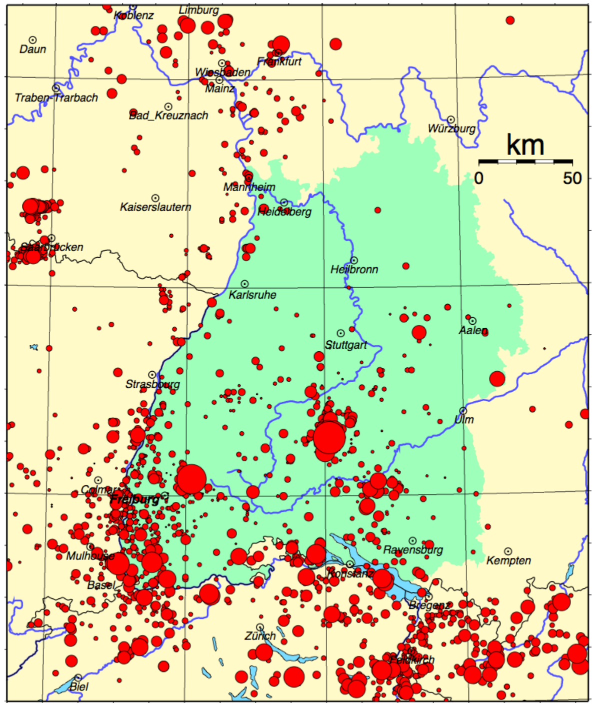




# Proportional Symbol Maps für Punktdaten

[Cabello et al. 2010]



# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

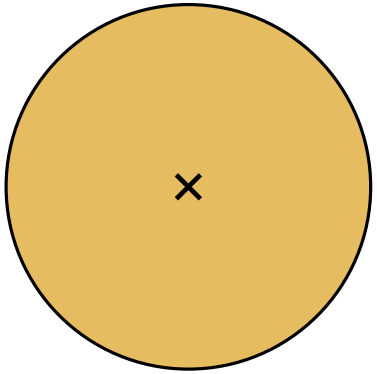
Wo ist dabei das Problem?

# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

Wo ist dabei das Problem?

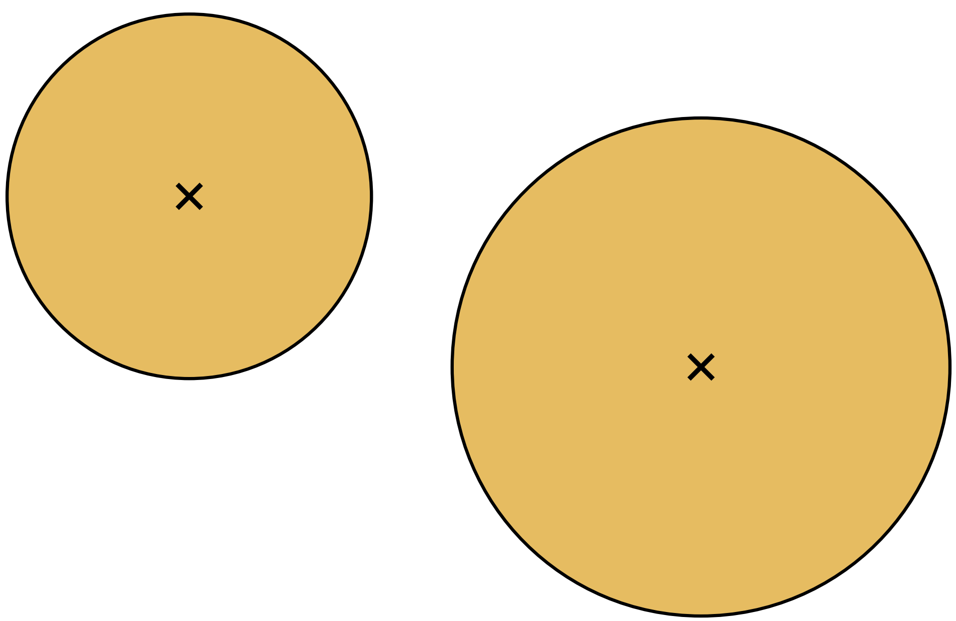


# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

Wo ist dabei das Problem?

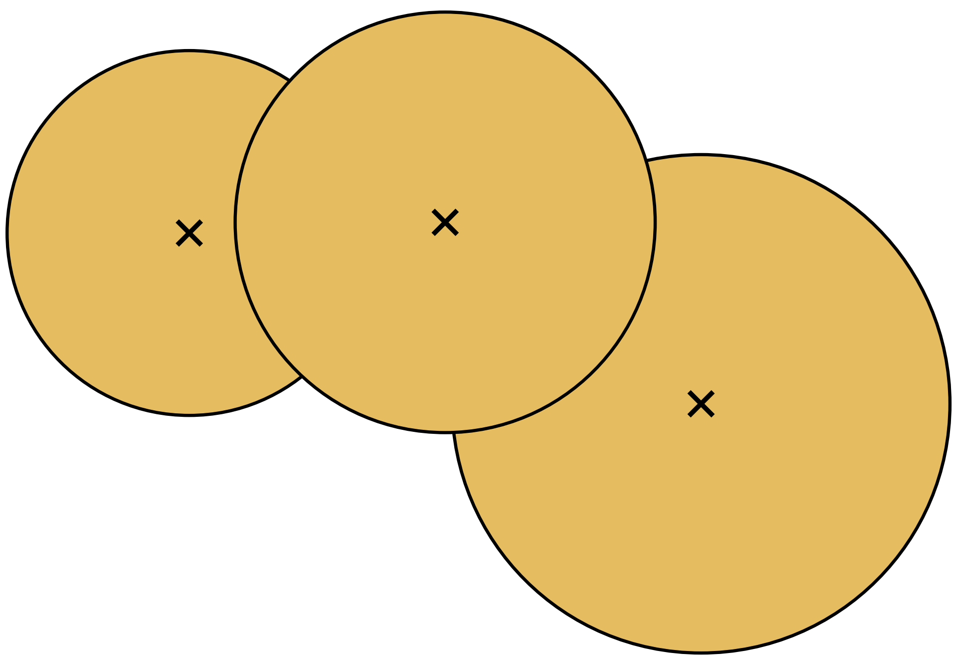


# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

Wo ist dabei das Problem?

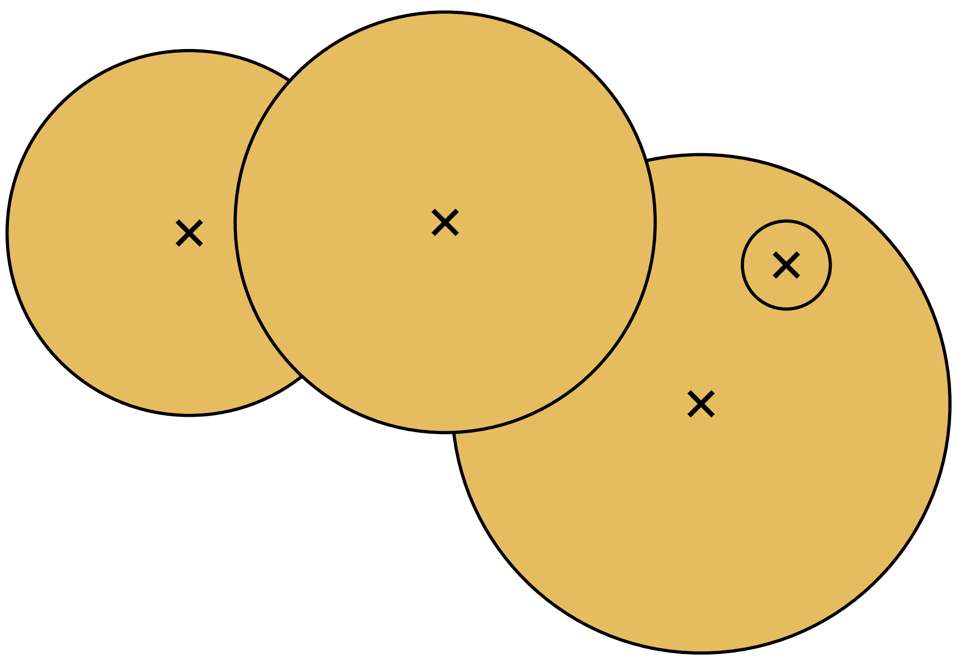


# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

Wo ist dabei das Problem?

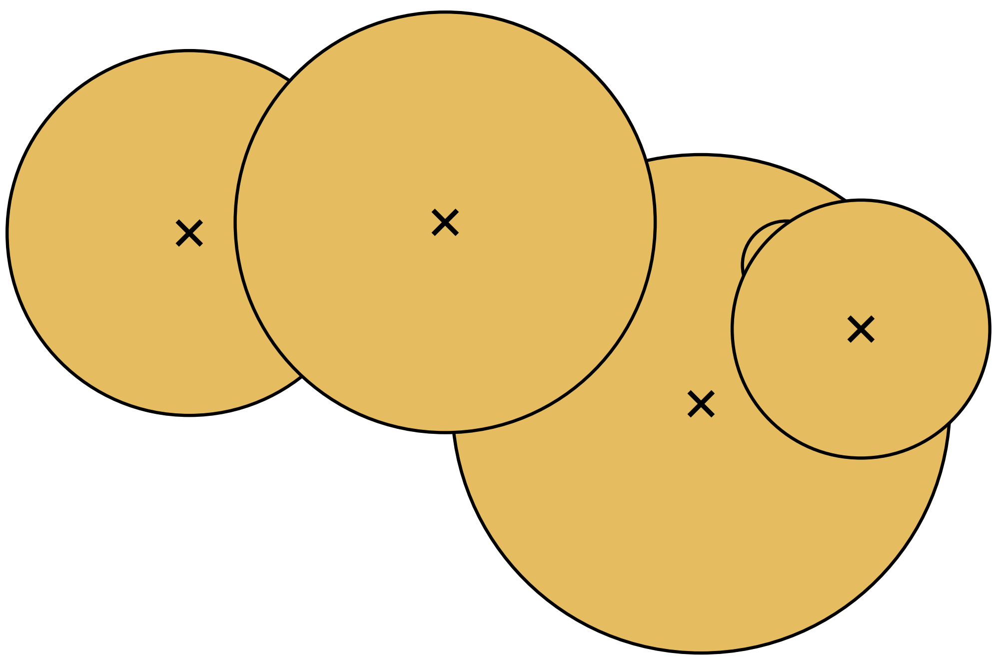


# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

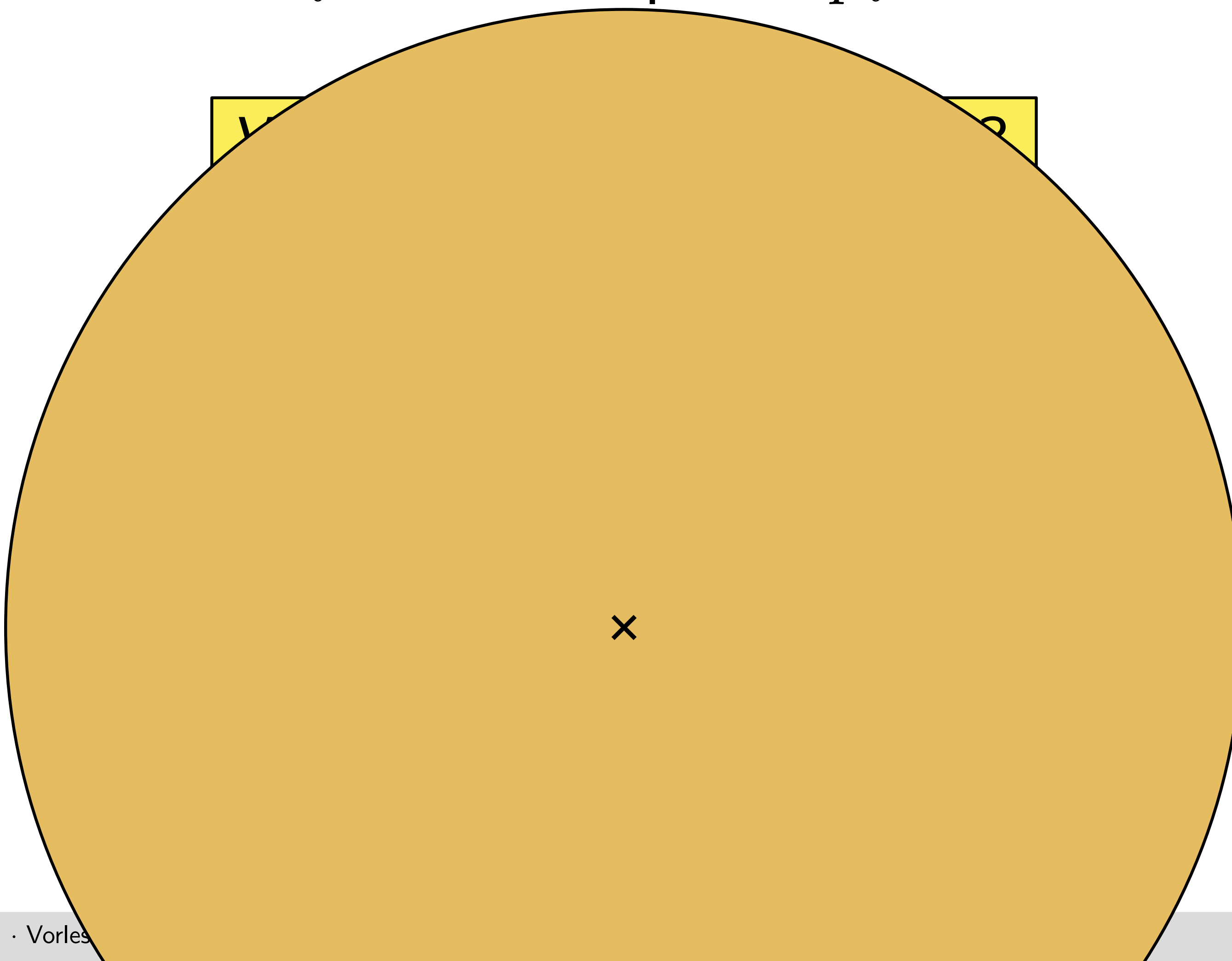
Wo ist dabei das Problem?



# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

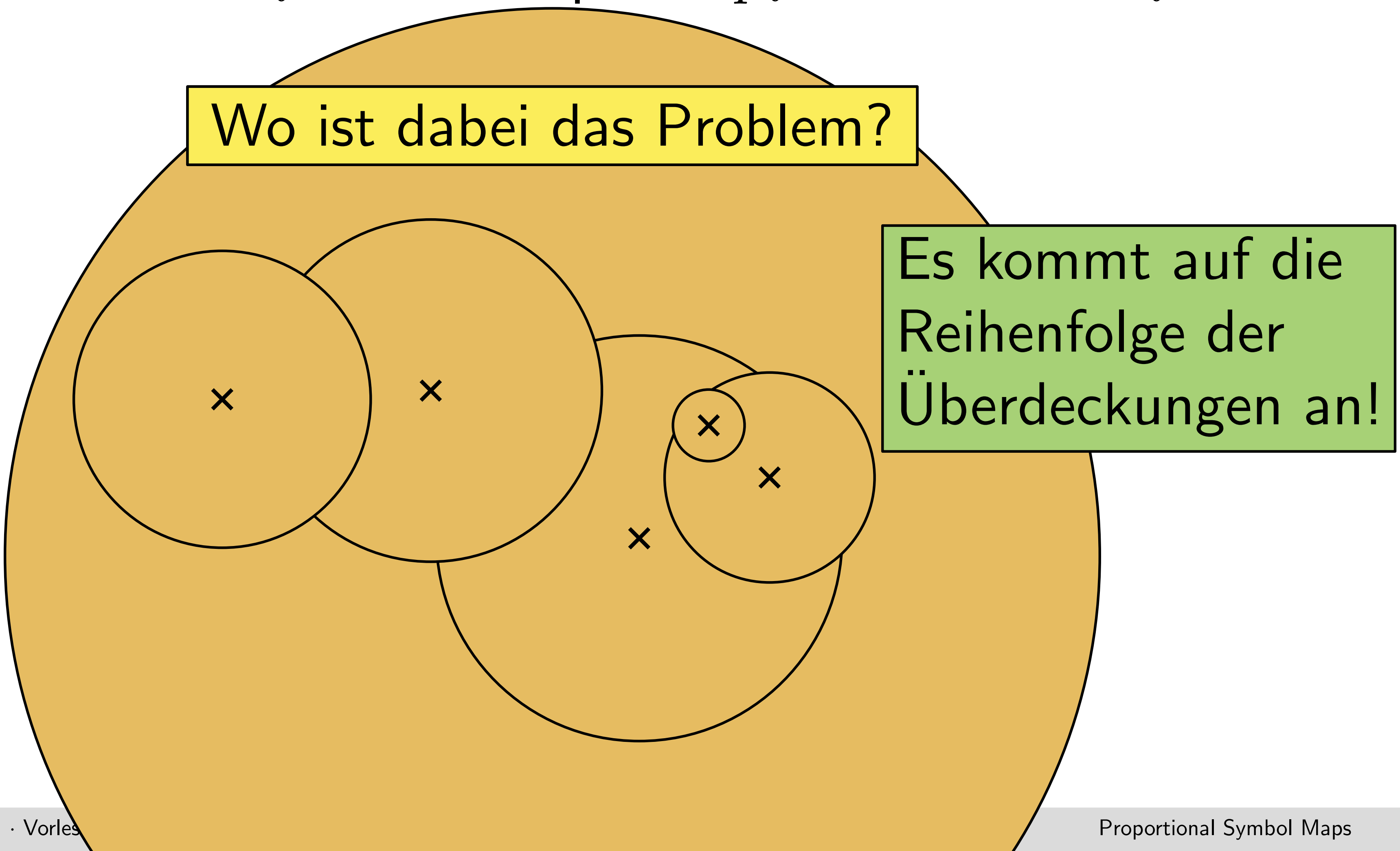
**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.



# Problemstellung

**Geg:** Menge $P = \{p_1, \dots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \dots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.



# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

**Qualitätskriterien:**

Ideen?

# Problemstellung

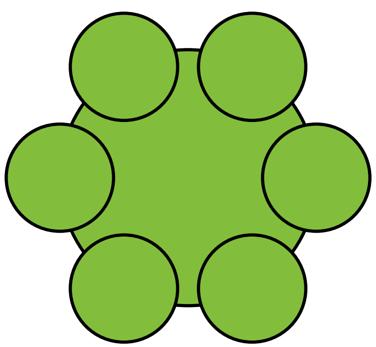
**Geg:** Menge $P = \{p_1, \dots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \dots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

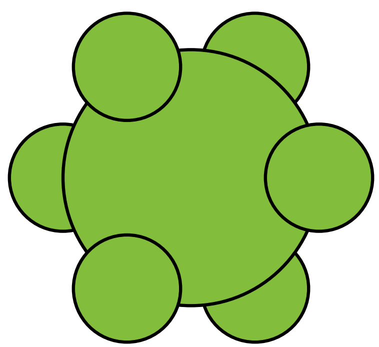
relativ oder absolut?

## Qualitätskriterien:

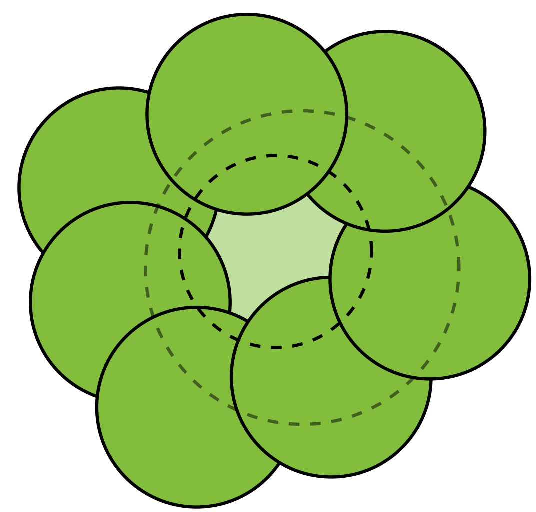
- maximiere die Länge des minimal sichtbaren Randes aller Scheiben
- maximiere die Gesamtlänge des sichtbaren Randes



MaxTotal



MaxMin



sichtbare Fläche problematisch

# Problemstellung

**Geg:** Menge $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ von Punkten, Menge $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}^+$ von zugehörigen Werten

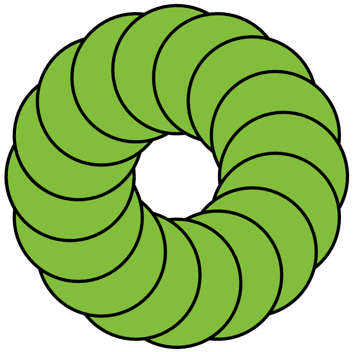
**Ges:** *gute* Visualisierung von $P$ und $W$, wobei $(p_i, w_i)$ Kreisscheibe $D_i$ mit Mittelpunkt $p_i$ und Fläche $w_i$ def.

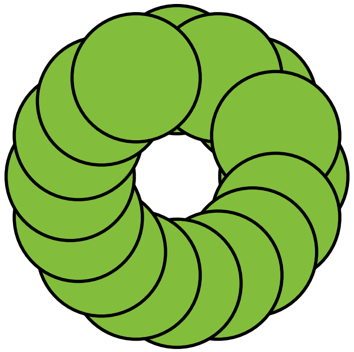
## Qualitätskriterien:

- maximiere die Länge des minimal sichtbaren Randes aller Scheiben
- maximiere die Gesamtlänge des sichtbaren Randes

## Modelle:

- Physisch realisierbares Modell: mit Münzen nachbaubar
- Stacking Modell: Totalordnung der Scheiben





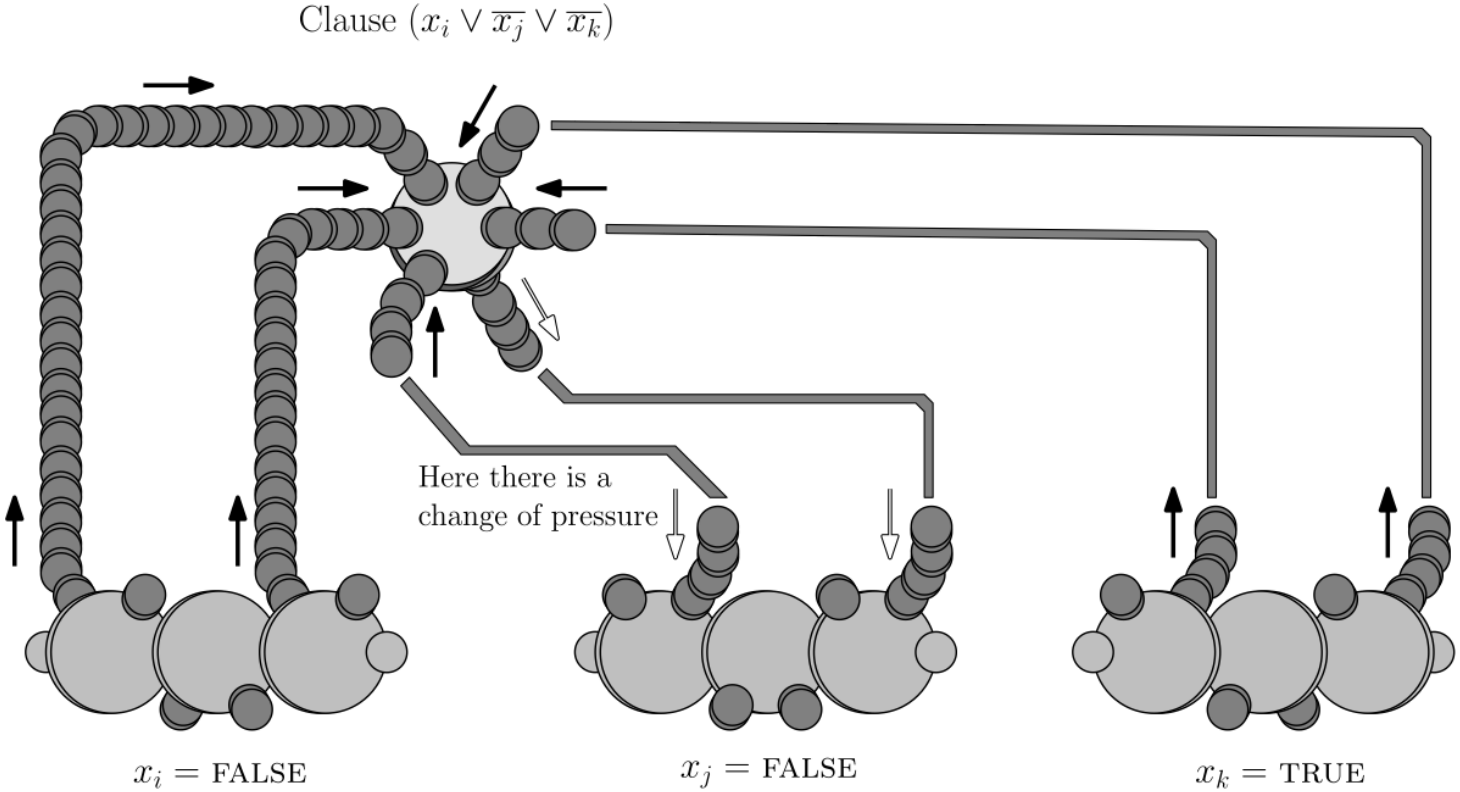
# Physisch Realisierbares Modell: Komplexität

**Satz 1:** Sowohl MaxMin als auch MaxTotal im physisch realisierbaren Modell sind NP-schwer.

# Physisch Realisierbares Modell: Komplexität

**Satz 1:** Sowohl MaxMin als auch MaxTotal im physisch realisierbaren Modell sind NP-schwer.

**Beweis:** wieder durch Reduktion von planarem 3-Sat

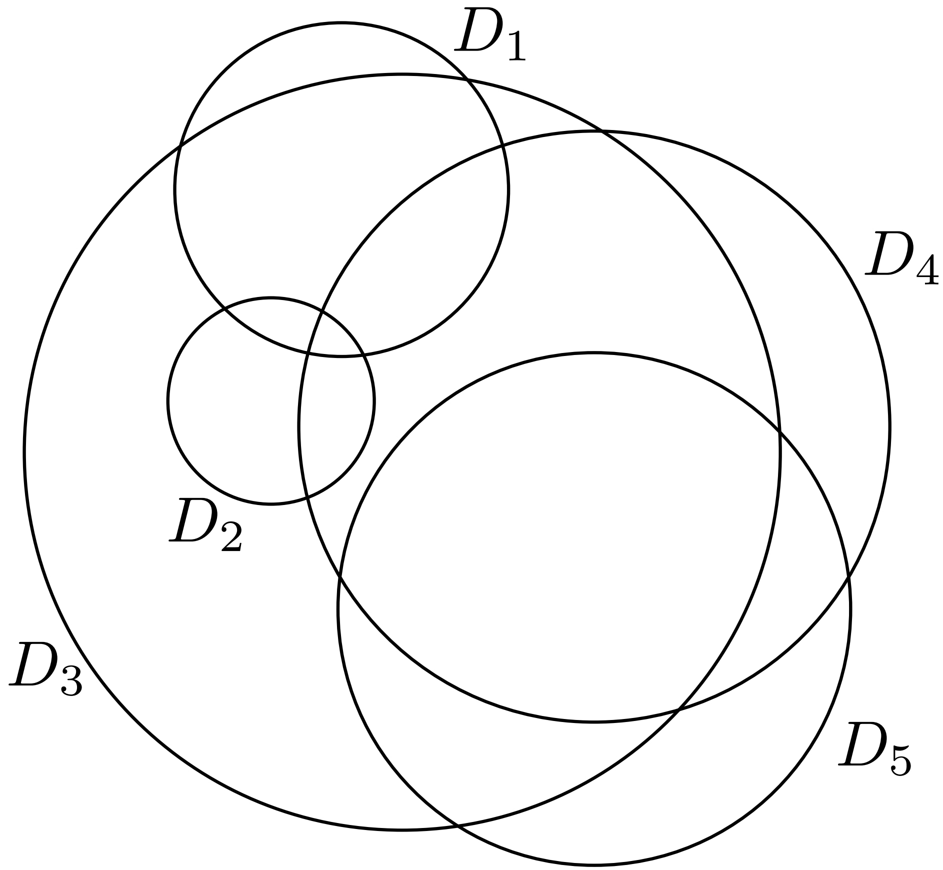



Werden wir aber hier nicht betrachten!

# MaxMin im Stacking Modell

**Geg:** Menge von Kreisscheiben $\mathcal{D} = \{D_1, \ldots, D_n\}$ in $\mathbb{R}^2$

**Ges:** Permutation $\pi$ von $\{1, \ldots, n\}$, so dass die Proportional Symbol Map mit Stacking Ordnung $D_{\pi(1)} < \cdots < D_{\pi(n)}$ den minimal sichtbaren Rand von $\mathcal{D}$ maximiert

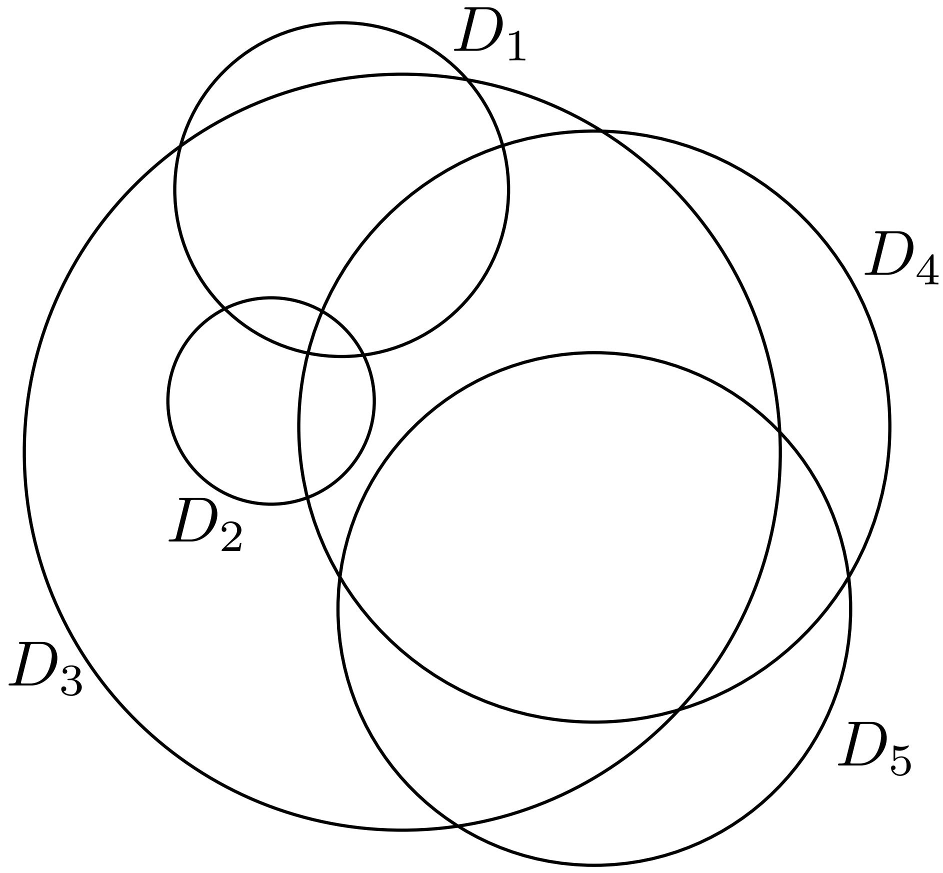


# MaxMin im Stacking Modell

**Geg:** Menge von Kreisscheiben $\mathcal{D} = \{D_1, \ldots, D_n\}$ in $\mathbb{R}^2$

**Ges:** Permutation $\pi$ von $\{1, \ldots, n\}$, so dass die Proportional Symbol Map mit Stacking Ordnung $D_{\pi(1)} < \cdots < D_{\pi(n)}$ den minimal sichtbaren Rand von $\mathcal{D}$ maximiert

Überlegen Sie sich einen Algorithmus zur Optimierung der Stacking Ordnung.

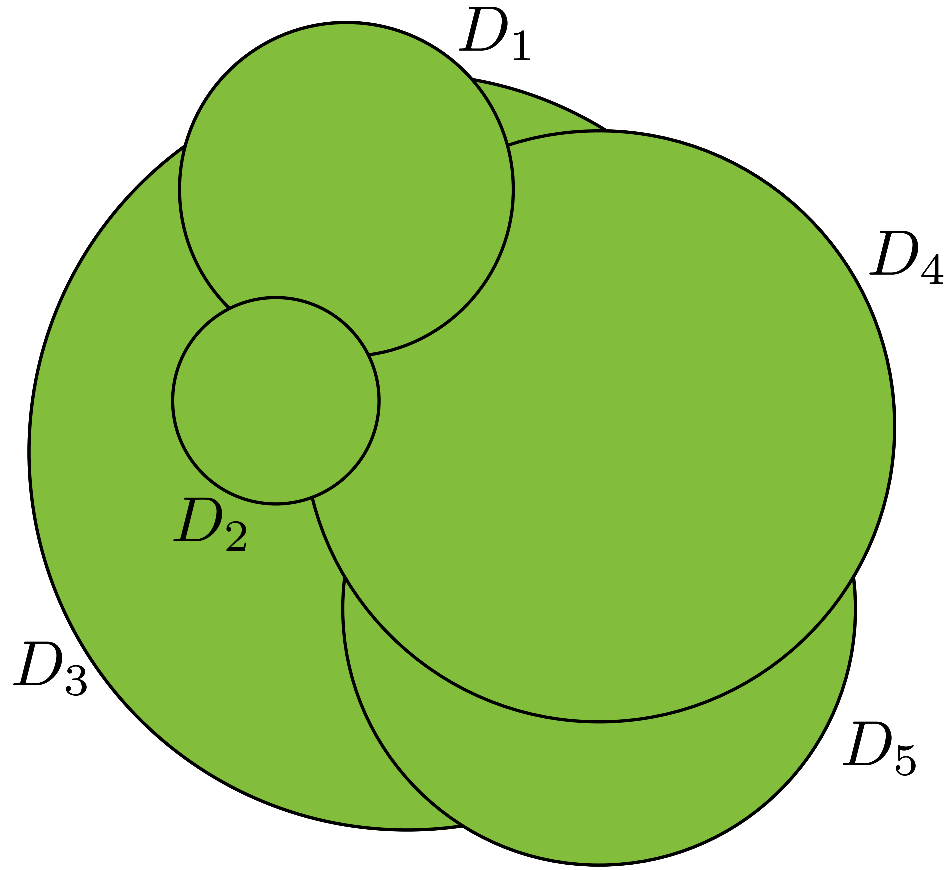


# MaxMin im Stacking Modell

**Geg:** Menge von Kreisscheiben $\mathcal{D} = \{D_1, \ldots, D_n\}$ in $\mathbb{R}^2$

**Ges:** Permutation $\pi$ von $\{1, \ldots, n\}$, so dass die Proportional Symbol Map mit Stacking Ordnung $D_{\pi(1)} < \cdots < D_{\pi(n)}$ den minimal sichtbaren Rand von $\mathcal{D}$ maximiert

Überlegen Sie sich einen Algorithmus zur Optimierung der Stacking Ordnung.



# Greedy-Algorithmus MaxMin Stacking

**Input**: Menge von Kreisscheiben $\mathcal{D} = \{D_1, \ldots, D_n\}$
**Output**: Permutation $\pi$ als optimale Stacking Ordnung

---

initialisiere $\mathcal{S} \leftarrow \mathcal{D}$
**for** $i = 1, \ldots, n$ **do**
  **foreach** $D_j \in \mathcal{S}$ **do**
    vis($D_j$) $\leftarrow$ sichtb. Rand von $D_j$ bzgl. $\mathcal{S}$ falls $\pi(i) = j$
  $D_k \leftarrow \arg\max_{D_j \in \mathcal{S}}$ vis($D_j$)
  set $\pi(i) \leftarrow k$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{D_k\}$
**return** $\pi$

# Greedy-Algorithmus MaxMin Stacking

**Input**: Menge von Kreisscheiben $\mathcal{D} = \{D_1, \ldots, D_n\}$
**Output**: Permutation $\pi$ als optimale Stacking Ordnung

---

initialisiere $\mathcal{S} \leftarrow \mathcal{D}$
**for** $i = 1, \ldots, n$ **do**
  **foreach** $D_j \in \mathcal{S}$ **do**
    vis$(D_j) \leftarrow$ sichtb. Rand von $D_j$ bzgl. $\mathcal{S}$ falls $\pi(i) = j$
  $D_k \leftarrow \arg\max_{D_j \in \mathcal{S}}$ vis$(D_j)$
  set $\pi(i) \leftarrow k$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{D_k\}$
**return** $\pi$

Korrektheit?
Laufzeit?

# MaxMin Stacking: Ergebnis

**Satz 2:** Gegeben $n$ Kreisscheiben in der Ebene berechnet der Greedy-Algorithmus eine Stacking Ordnung, die den minimal sichtbaren Rand aller Kreisscheiben maximiert. Der Algorithmus lässt sich in $O(n^2 \log n)$ Laufzeit implementieren.

# MaxMin Stacking: Ergebnis

**Satz 2:** Gegeben $n$ Kreisscheiben in der Ebene berechnet der Greedy-Algorithmus eine Stacking Ordnung, die den minimal sichtbaren Rand aller Kreisscheiben maximiert. Der Algorithmus lässt sich in $O(n^2 \log n)$ Laufzeit implementieren.

**Beweis:**

- Korrektheit: Vergleiche Lösung $\mathcal{A}$ des Algorithmus mit optimaler Lösung $\mathcal{O}$ und nutze Austauschargument
- Laufzeit: nutze Variante von Segment-Bäumen

  [de Berg et al. '08, Kap. 10.3]

# Korrektheit

Betrachte Scheiben in aufsteigener Stacking-Reihenfolge.

| | $\mathcal{A}$ | $\mathcal{O}$ |
|---|---|---|
| 7 | $D_7$ | $D_7$ |
| 6 | $D_6$ | $D_2$ |
| 5 | $D_4$ | $D_1$ |
| 4 | $D_2$ | $D_6$ |
| 3 | $D_1$ | $D_4$ |
| 2 | $D_3$ | $D_3$ |
| 1 | $D_5$ | $D_5$ |

**Vergleiche:**
$\mathcal{A} =$ Lösung des Algorithmus
$\mathcal{O} =$ Beliebige optimale Lösung

# Korrektheit

Betrachte Scheiben in aufsteigener Stacking-Reihenfolge.

| | $\mathcal{A}$ | $\mathcal{O}$ |
|---|---|---|
| 7 | $D_7$ | $D_7$ |
| 6 | $D_6$ | $D_2$ |
| 5 | $D_4$ | $D_1$ |
| 4 | $D_2$ | $D_6$ |
| 3 | $D_1$ | $D_4$ |
| 2 | $D_3$ | $D_3$ |
| 1 | $D_5$ | $D_5$ |

**Vergleiche:**
$\mathcal{A}$ = Lösung des Algorithmus
$\mathcal{O}$ = Beliebige optimale Lösung

Sei $i$ kleinster Wert, sodass $\mathcal{A}[i] \neq \mathcal{O}[i]$

$\mathcal{A}[i] = \text{argmax}_{D \in S}\text{vis}(D)$

# Korrektheit

Betrachte Scheiben in aufsteigener Stacking-Reihenfolge.

| | $\mathcal{A}$ | $\mathcal{O}$ |
|---|---|---|
| 7 | $D_7$ | $D_7$ |
| 6 | $D_6$ | $D_2$ |
| 5 | $D_4$ | $D_1$ |
| 4 | $D_2$ | $D_6$ |
| 3 | $D_1$ | $D_4$ |
| 2 | $D_3$ | $D_3$ |
| 1 | $D_5$ | $D_5$ |

**Vergleiche:**
$\mathcal{A}$ = Lösung des Algorithmus
$\mathcal{O}$ = Beliebige optimale Lösung

Sei $i$ kleinster Wert, sodass $\mathcal{A}[i] \neq \mathcal{O}[i]$

$\mathcal{A}[i] = \text{argmax}_{D \in S} \text{vis}(D)$

Ziehe $\mathcal{A}[i]$ in $\mathcal{O}$ an Stelle $\mathcal{O}[i]$:

# Korrektheit

Betrachte Scheiben in aufsteigener Stacking-Reihenfolge.

| | $\mathcal{A}$ | $\mathcal{O}$ |
|---|---|---|
| 7 | $D_7$ | $D_7$ |
| 6 | $D_6$ | $D_2$ |
| 5 | $D_4$ | $\boxed{D_1}$ |
| 4 | $D_2$ | $D_6$ |
| 3 | $D_1$ | $D_4$ |
| 2 | $D_3$ | $D_3$ |
| 1 | $D_5$ | $D_5$ |

**Vergleiche:**
$\mathcal{A}$ = Lösung des Algorithmus
$\mathcal{O}$ = Beliebige optimale Lösung

Sei $i$ kleinster Wert, sodass $\mathcal{A}[i] \neq \mathcal{O}[i]$

$\mathcal{A}[i] = \text{argmax}_{D \in S} \text{vis}(D)$

Ziehe $\mathcal{A}[i]$ in $\mathcal{O}$ an Stelle $\mathcal{O}[i]$:

- verändert den Wert min-vis$(\mathcal{O})$ an Stelle $i$ nicht.

# Korrektheit

Betrachte Scheiben in aufsteigener Stacking-Reihenfolge.

| | $\mathcal{A}$ | $\mathcal{O}$ |
|---|---|---|
| 7 | $D_7$ | $D_7$ |
| 6 | $D_6$ | $D_2$ |
| 5 | $D_4$ | $\boxed{D_1}$ |
| 4 | $D_2$ | $D_6$ |
| 3 | $D_1$ | $D_4$ |
| 2 | $D_3$ | $D_3$ |
| 1 | $D_5$ | $D_5$ |

**Vergleiche:**
$\mathcal{A}$ = Lösung des Algorithmus
$\mathcal{O}$ = Beliebige optimale Lösung

Sei $i$ kleinster Wert, sodass $\mathcal{A}[i] \neq \mathcal{O}[i]$

$\mathcal{A}[i] = \text{argmax}_{D \in S} \text{vis}(D)$

Ziehe $\mathcal{A}[i]$ in $\mathcal{O}$ an Stelle $\mathcal{O}[i]$:

- verändert den Wert min-vis($\mathcal{O}$) an Stelle $i$ nicht.
- Übersprungene Scheiben können nur besser werden.

# Korrektheit

Betrachte Scheiben in aufsteigener Stacking-Reihenfolge.

| | $\mathcal{A}$ | $\mathcal{O}$ |
|---|---|---|
| 7 | $D_7$ | $D_7$ |
| 6 | $D_6$ | $D_2$ |
| 5 | $D_4$ | $\boxed{D_1}$ |
| 4 | $D_2$ | $D_6$ |
| 3 | $D_1$ | $D_4$ |
| 2 | $D_3$ | $D_3$ |
| 1 | $D_5$ | $D_5$ |

**Vergleiche:**
$\mathcal{A}$ = Lösung des Algorithmus
$\mathcal{O}$ = Beliebige optimale Lösung

Sei $i$ kleinster Wert, sodass $\mathcal{A}[i] \neq \mathcal{O}[i]$

$\mathcal{A}[i] = \text{argmax}_{D \in S}\text{vis}(D)$

Ziehe $\mathcal{A}[i]$ in $\mathcal{O}$ an Stelle $\mathcal{O}[i]$:

- verändert den Wert min-vis($\mathcal{O}$) an Stelle $i$ nicht.
- Übersprungene Scheiben können nur besser werden.
- Restliche Scheiben bleiben unverändert.

# Korrektheit

Betrachte Scheiben in aufsteigener Stacking-Reihenfolge.

| | $\mathcal{A}$ | $\mathcal{O}$ |
|---|---|---|
| 7 | $D_7$ | $D_7$ |
| 6 | $D_6$ | $D_2$ |
| 5 | $D_4$ | $D_1$ |
| 4 | $D_2$ | $D_6$ |
| 3 | $D_1$ | $D_4$ |
| 2 | $D_3$ | $D_3$ |
| 1 | $D_5$ | $D_5$ |

**Vergleiche:**
$\mathcal{A} =$ Lösung des Algorithmus
$\mathcal{O} =$ Beliebige optimale Lösung

Sei $i$ kleinster Wert, sodass $\mathcal{A}[i] \neq \mathcal{O}[i]$

$\mathcal{A}[i] = \text{argmax}_{D \in S} \text{vis}(D)$

Ziehe $\mathcal{A}[i]$ in $\mathcal{O}$ an Stelle $\mathcal{O}[i]$:

- verändert den Wert min-vis$(\mathcal{O})$ an Stelle $i$ nicht.
- Übersprungene Scheiben können nur besser werden.
- Restliche Scheiben bleiben unverändert.

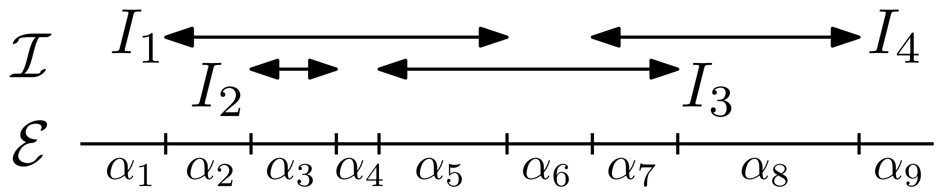
→ Modifikationen erhalten Wert von $O$

→ Wiederhole Modifikation bis $\mathcal{A} = \mathcal{O}$.

# Segment-Bäume

Datenstruktur zum Speichern einer Menge von Intervallen in $\mathbb{R}$ in $O(n \log n)$ Platz für Punkt-in-Intervall Anfragen in $O(\log n + k)$ Zeit („Gegeben Punkt $p$, welche Intervalle enthalten $p$?“)

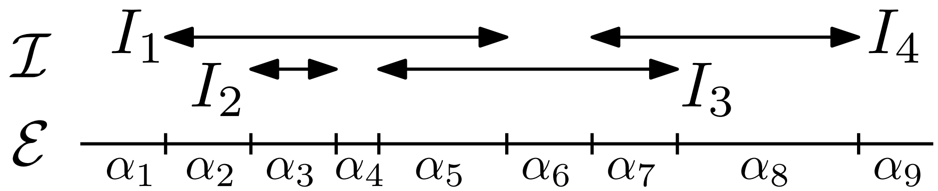
- Menge der Intervalle $\mathcal{I} = \{I_1, \ldots, I_k\}$ definieren Menge von Elementarintervallen $\mathcal{E} = \{\alpha_1, \ldots, \alpha_m\}$



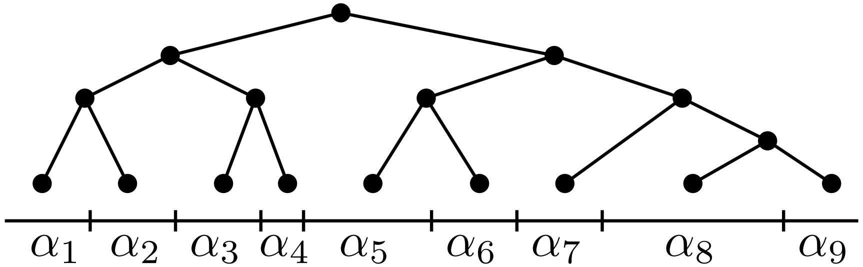
# Segment-Bäume

Datenstruktur zum Speichern einer Menge von Intervallen in $\mathbb{R}$ in $O(n \log n)$ Platz für Punkt-in-Intervall Anfragen in $O(\log n + k)$ Zeit („Gegeben Punkt $p$, welche Intervalle enthalten $p$?")

- Menge der Intervalle $\mathcal{I} = \{I_1, \ldots, I_k\}$ definieren Menge von Elementarintervallen $\mathcal{E} = \{\alpha_1, \ldots, \alpha_m\}$




- Elementarintervalle $\alpha_i$ def. Blätter $\mu_i$ eines balancierten Binärbaums, innere Knoten $v$ entsprechen Vereinigung der Kindintervalle



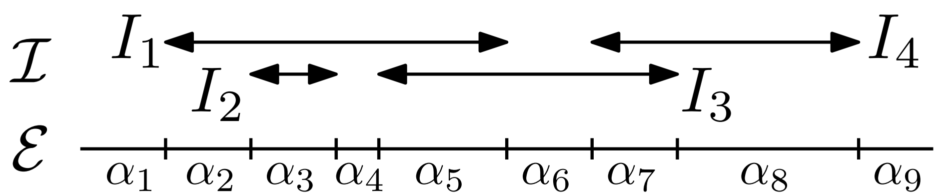

$\mathsf{Int}(\mu_i) = \alpha_i$
$\mathsf{Int}(v) = \mathsf{Int}(\mathsf{lc}(v)) \cup \mathsf{Int}(\mathsf{rc}(v))$

# Segment-Bäume

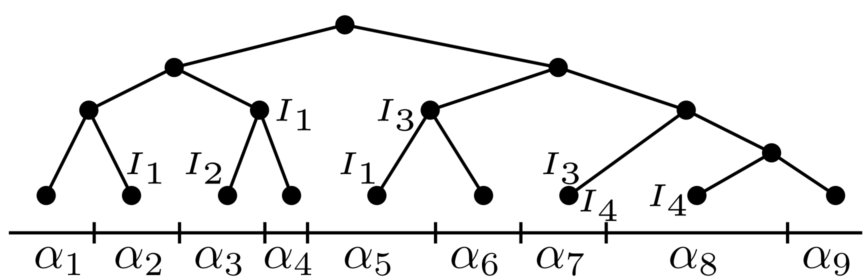


Datenstruktur zum Speichern einer Menge von Intervallen in $\mathbb{R}$ in $O(n \log n)$ Platz für Punkt-in-Intervall Anfragen in $O(\log n + k)$ Zeit („Gegeben Punkt $p$, welche Intervalle enthalten $p$?“)

- Menge der Intervalle $\mathcal{I} = \{I_1, \ldots, I_k\}$ definieren Menge von Elementarintervallen $\mathcal{E} = \{\alpha_1, \ldots, \alpha_m\}$




- Elementarintervalle $\alpha_i$ def. Blätter $\mu_i$ eines balancierten Binärbaums, innere Knoten $v$ entsprechen Vereinigung der Kindintervalle



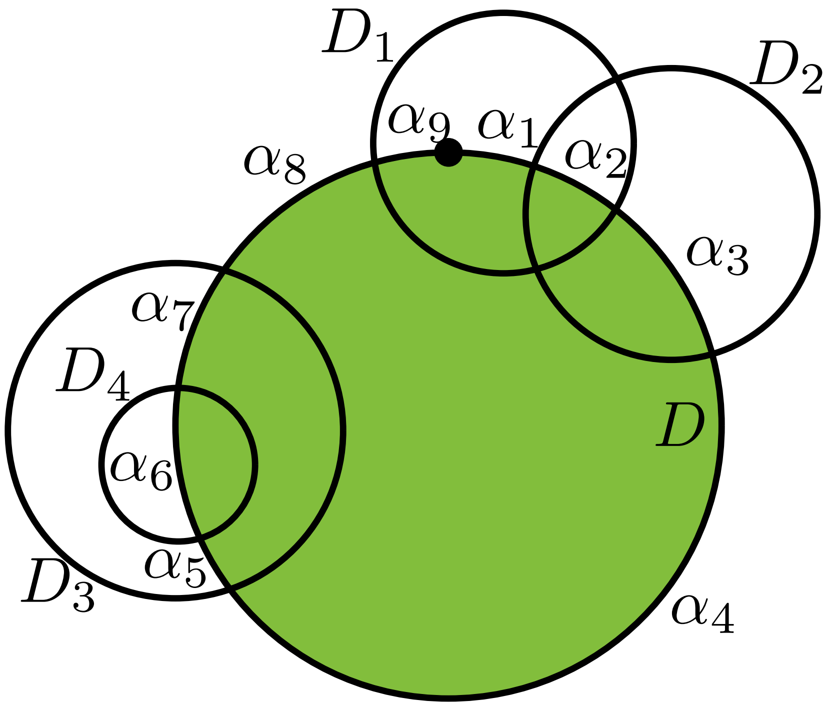

$\mathrm{Int}(\mu_i) = \alpha_i$
$\mathrm{Int}(v) = \mathrm{Int}(\mathrm{lc}(v)) \cup \mathrm{Int}(\mathrm{rc}(v))$

- Knoten $v$ speichern Menge $I(v)$ von Eingabeintervallen $I_j$ mit $\mathrm{Int}(v) \subseteq I_j$ und $\mathrm{Int}(\mathrm{parent}(v)) \not\subseteq I_j$

($I_j$ wird pro Level max. 2-mal gespeichert)

# Segment-Bäume für Kreisränder

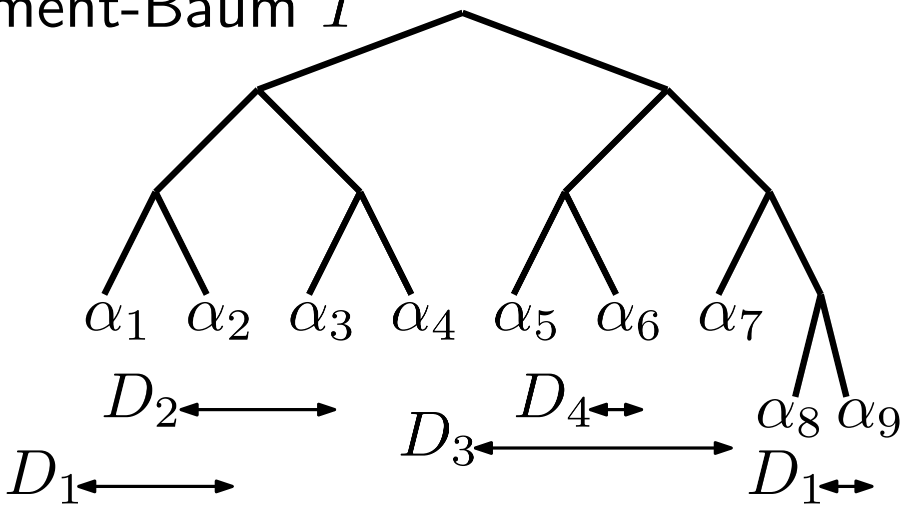
- betrachte Rand eines Kreises $D$ von oben im UZS als lineares Intervall
- definiere Elementarintervalle aus Schnitten mit anderen Kreisen in $\mathcal{D}$
- jeder Kreis $D' \neq D$ schneidet $D$ in 0, 1 oder 2 Intervallen

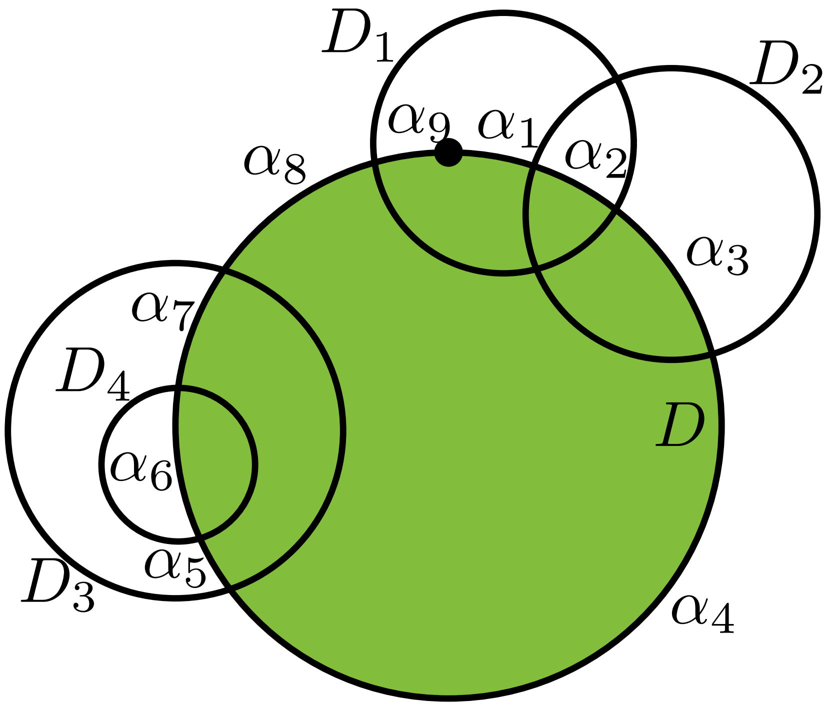


# Segment-Bäume für Kreisränder

- betrachte Rand eines Kreises $D$ von oben im UZS als lineares Intervall
- definiere Elementarintervalle aus Schnitten mit anderen Kreisen in $\mathcal{D}$
- jeder Kreis $D' \neq D$ schneidet $D$ in 0, 1 oder 2 Intervallen

Segment-Baum $T$

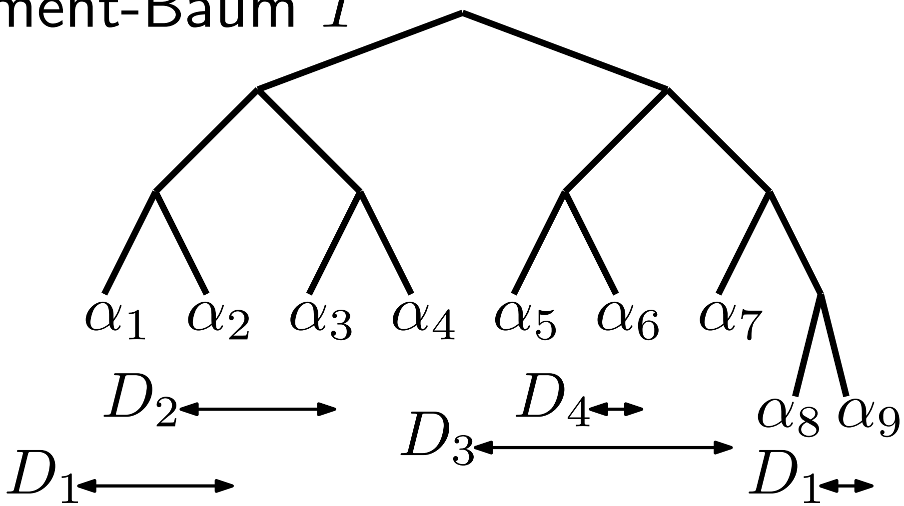


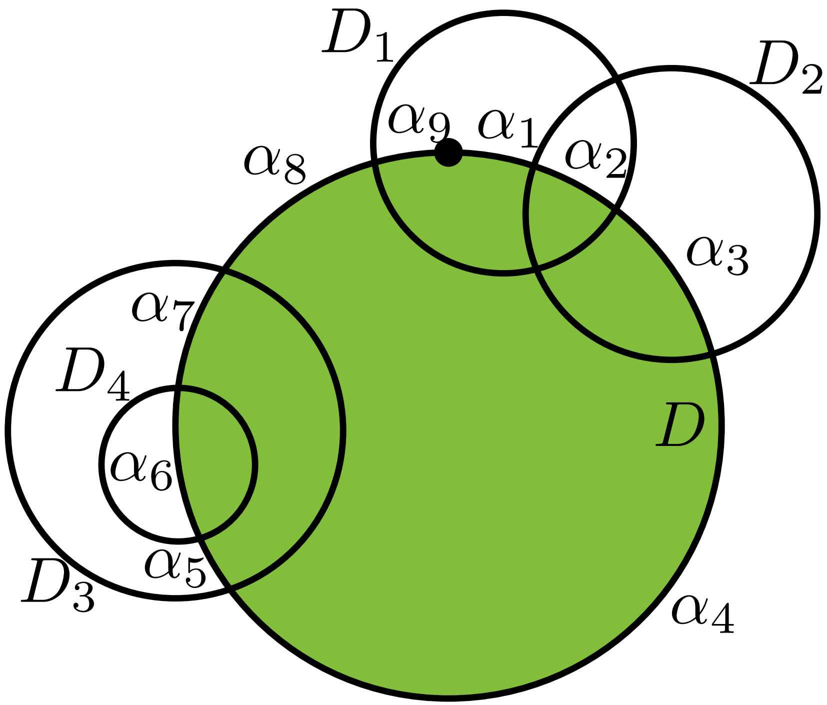



# Segment-Bäume für Kreisränder

- betrachte Rand eines Kreises $D$ von oben im UZS als lineares Intervall
- definiere Elementarintervalle aus Schnitten mit anderen Kreisen in $\mathcal{D}$
- jeder Kreis $D' \neq D$ schneidet $D$ in 0, 1 oder 2 Intervallen

Segment-Baum $T$



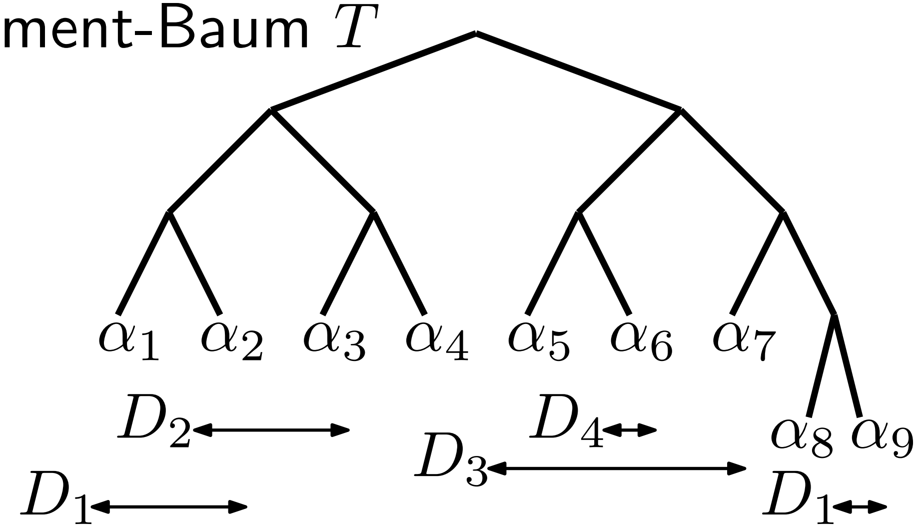

- speichere zusätzlich zu $\mathsf{Int}(v)$ und $I(v)$ an jedem Knoten noch $\mathsf{vis}(v)$

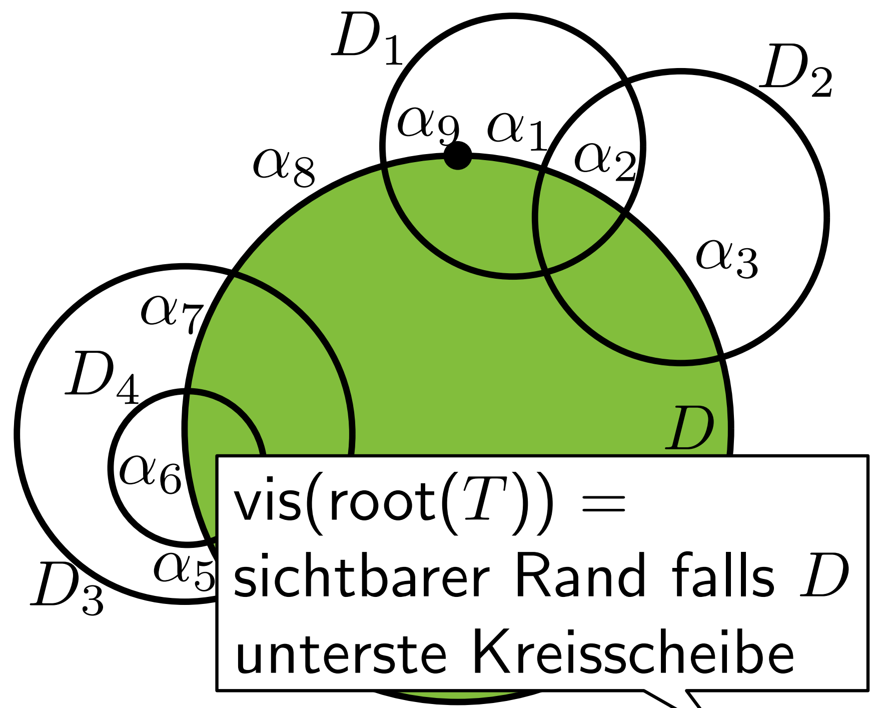
$$\mathsf{vis}(v) = \begin{cases} 0 & \text{falls } |I(v)| \geq 1 \\ \mathsf{vis}(\mathsf{lc}(v)) + \mathsf{vis}(\mathsf{rc}(v)) & \text{falls } v \text{ innerer Knoten} \\ |\mathsf{Int}(v)| & \text{falls } v \text{ Blatt} \end{cases}$$

# Segment-Bäume für Kreisränder

- betrachte Rand eines Kreises $D$ von oben im UZS als lineares Intervall
- definiere Elementarintervalle aus Schnitten mit anderen Kreisen in $\mathcal{D}$
- jeder Kreis $D' \neq D$ schneidet $D$ in 0, 1 oder 2 Intervallen

Segment-Baum $T$

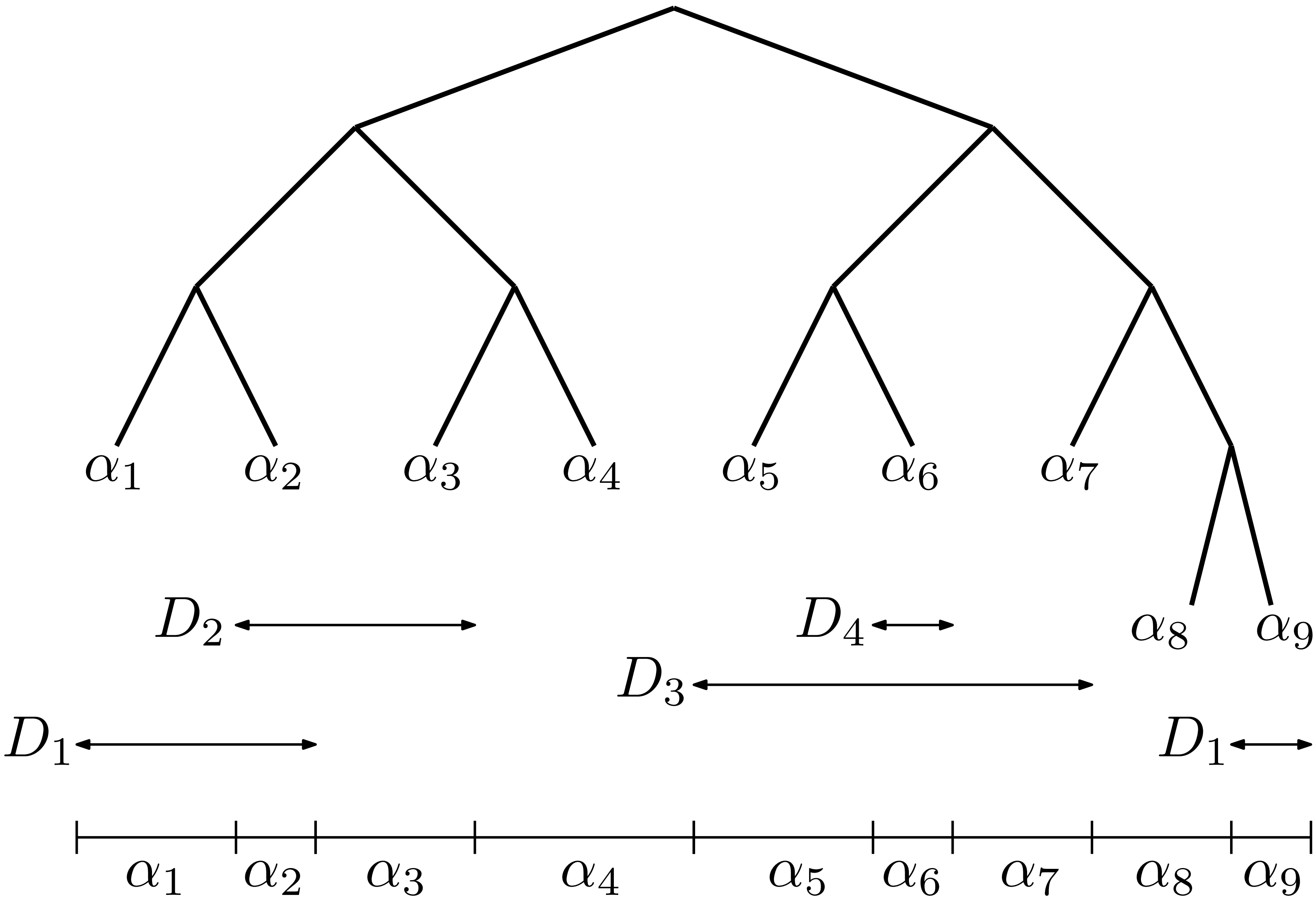


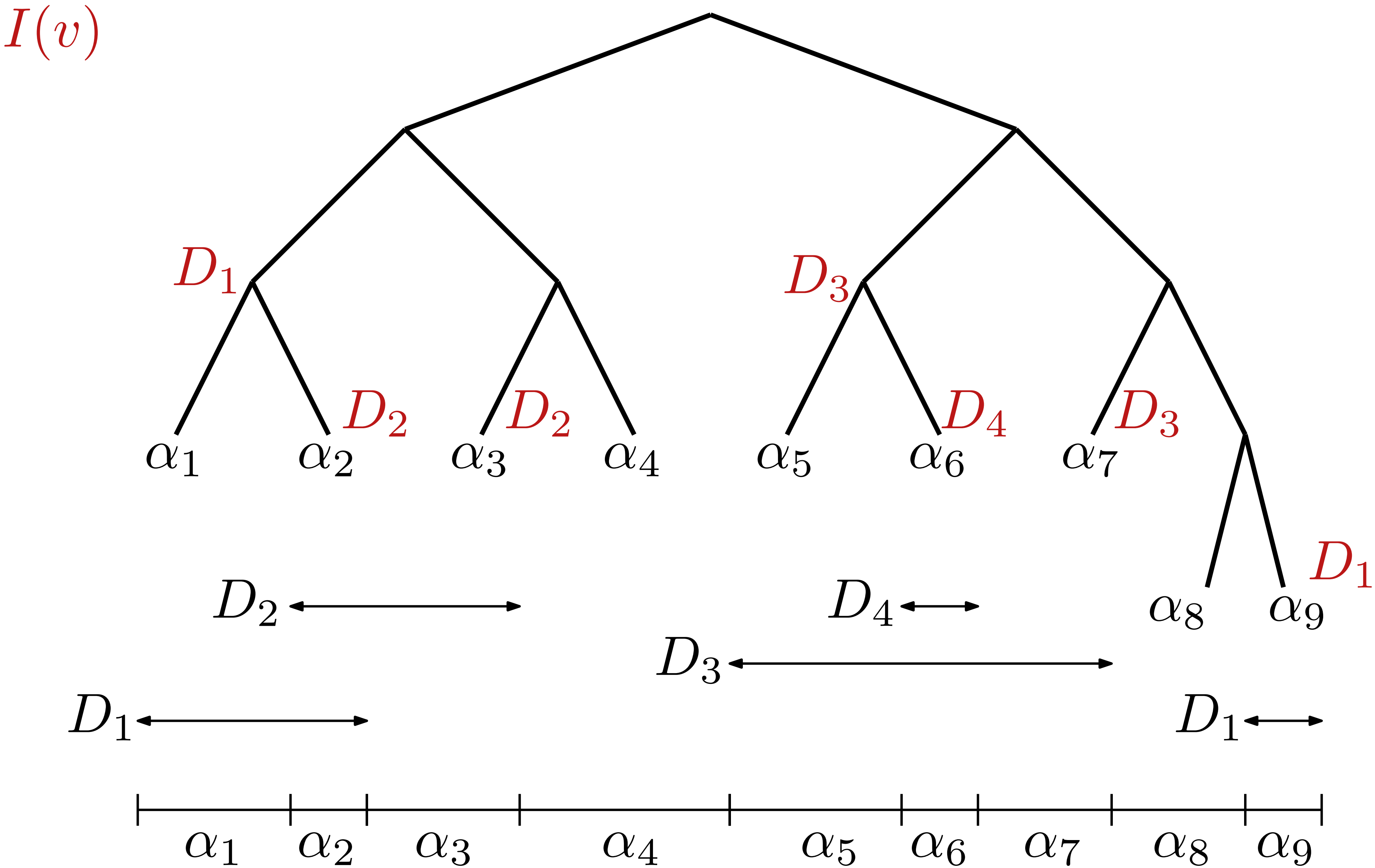

- speichere zusätzlich zu Int$(v)$ und $I(v)$ an jedem Knoten noch vis$(v)$

$$\text{vis}(v) = \begin{cases} 0 & \text{falls } |I(v)| \geq 1 \\ \text{vis}(\text{lc}(v)) + \text{vis}(\text{rc}(v)) & \text{falls } v \text{ innerer Knoten} \\ |\text{Int}(v)| & \text{falls } v \text{ Blatt} \end{cases}$$

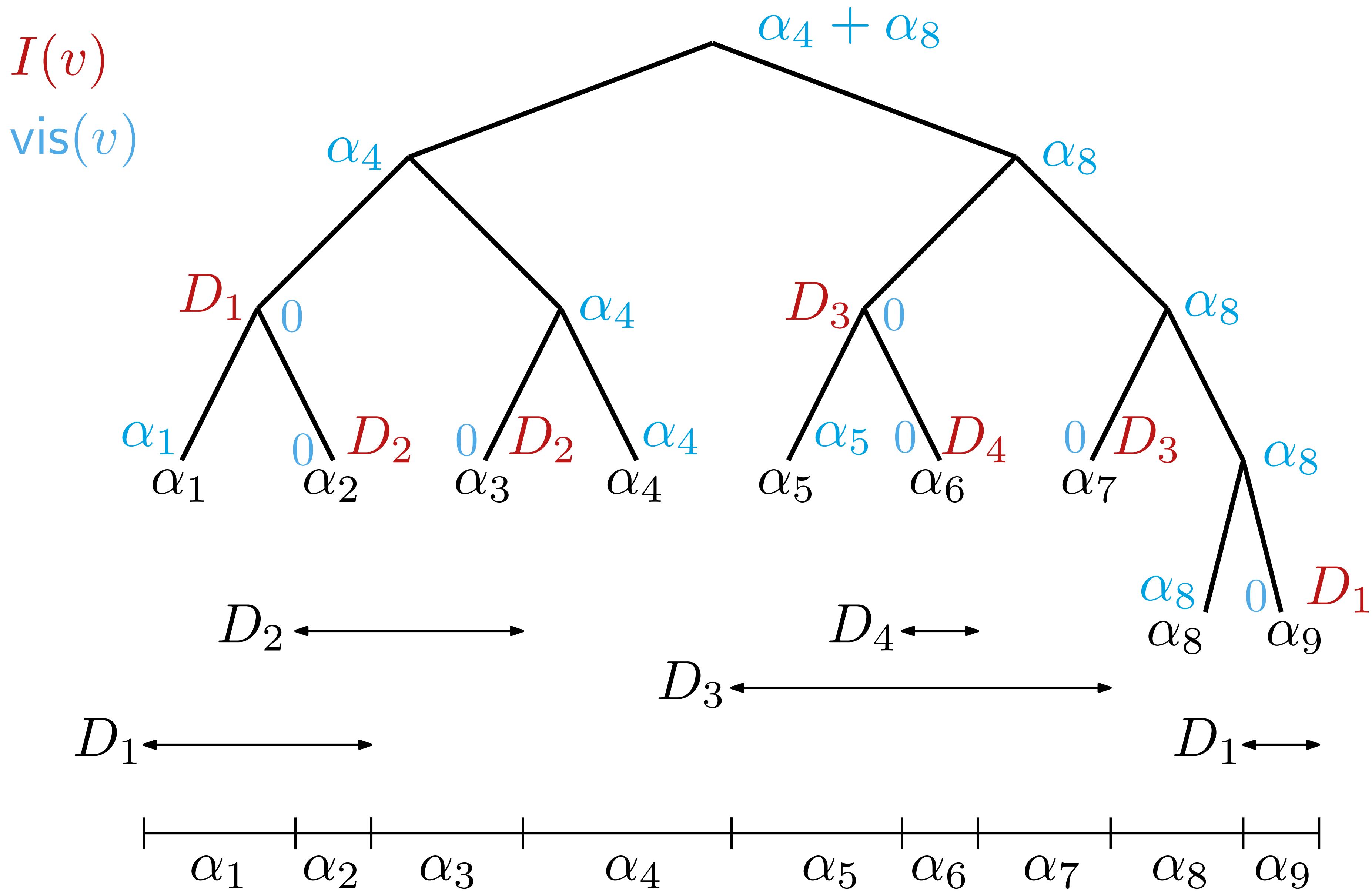
# Segment-Bäume für Kreisränder: Beispiel



# Segment-Bäume für Kreisränder: Beispiel



# Segment-Bäume für Kreisränder: Beispiel



# Implementierung Greedy Algorithmus

(1) erstelle modifizierten Segment-Baum $T_i$ für jede Kreisscheibe $D_i \in \mathcal{S}$
(2) bestimme $D_k = \arg\max_{D_i \in \mathcal{S}}$ vis(root($T_i$))
(3) lösche $T_k$
(4) lösche $D_k$ aus $\mathcal{S}$ und verbleibenden $T_i$
(5) falls $\mathcal{S} \neq \emptyset$ gehe zu (2)

# Implementierung Greedy Algorithmus

| | |
|---|---|
| (1) erstelle modifizierten Segment-Baum $T_i$ für jede Kreisscheibe $D_i \in \mathcal{S}$ | $O(n^2 \log n)$ |
| (2) bestimme $D_k = \arg\max_{D_i \in \mathcal{S}}$ vis(root($T_i$)) | $O(n)$ |
| (3) lösche $T_k$ | $O(1)$ |
| (4) lösche $D_k$ aus $\mathcal{S}$ und verbleibenden $T_i$ | $O(n \log n)$ |
| (5) falls $\mathcal{S} \neq \emptyset$ gehe zu (2) | |
| | $O(n^2 \log n)$ |

$\square$

# Implementierung Greedy Algorithmus

| | | |
|---|---|---|
| (1) | erstelle modifizierten Segment-Baum $T_i$ für jede Kreisscheibe $D_i \in \mathcal{S}$ | $O(n^2 \log n)$ |
| (2) | bestimme $D_k = \arg\max_{D_i \in \mathcal{S}}$ vis(root($T_i$)) | $O(n)$ |
| (3) | lösche $T_k$ | $O(1)$ |
| (4) | lösche $D_k$ aus $\mathcal{S}$ und verbleibenden $T_i$ | $O(n \log n)$ |
| (5) | falls $\mathcal{S} \neq \emptyset$ gehe zu (2) | |
| | | $O(n^2 \log n)$ |

$\square$

**Satz 2:** Gegeben $n$ Kreisscheiben in der Ebene berechnet der Greedy-Algorithmus eine Stacking Ordnung, die den minimal sichtbaren Rand aller Kreisscheiben maximiert. Der Algorithmus lässt sich in $O(n^2 \log n)$ Laufzeit implementieren.

**Satz 3:** Falls kein Punkt in mehr als $O(1)$ Kreisen enthalten ist, kann die optimale Stacking Ordnung in $O(n \log n)$ Zeit berechnet werden.

# Implementierung Greedy Algorithmus

(1) erstelle modifizierten Segment-Baum $T_i$ für jede Kreisscheibe $D_i \in \mathcal{S}$ $\quad O(n^2 \log n)$

(2) bestimme $D_k = \arg\max_{D_i \in \mathcal{S}}$ vis(root($T_i$)) $\quad O(n)$

(3) lösche $T_k$ $\quad O(1)$

(4) lösche $D_k$ aus $\mathcal{S}$ und verbleibenden $T_i$ $\quad O(n \log n)$

(5) falls $\mathcal{S} \neq \emptyset$ gehe zu (2)

$O(n^2 \log n)$

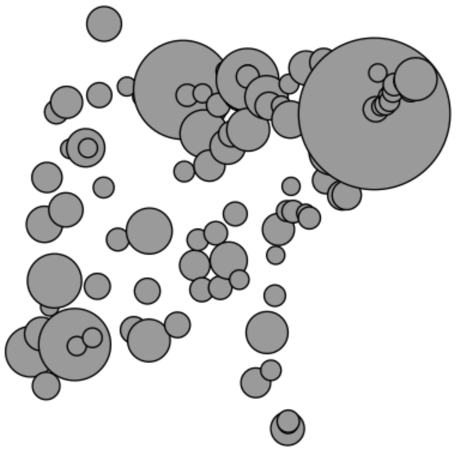
Für MaxTotal im Stacking Modell ist die Komplexität noch offen!

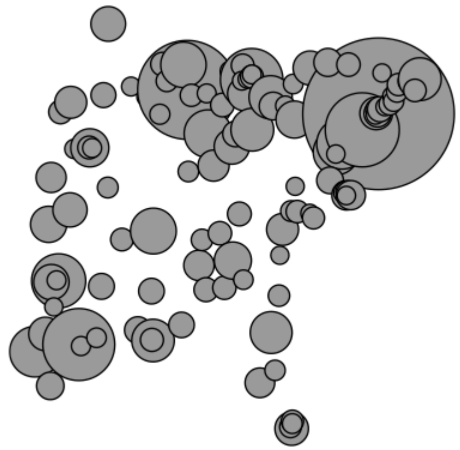
$\square$

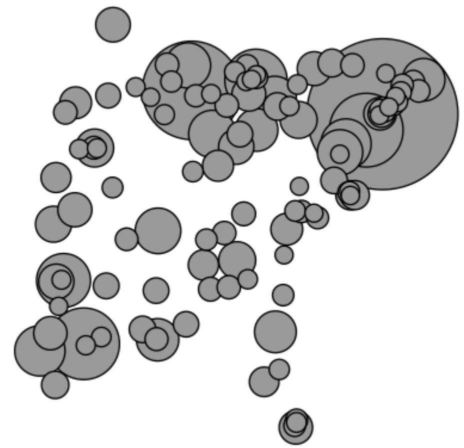
**Satz 2:** Gegeben $n$ Kreisscheiben in der Ebene berechnet der Greedy-Algorithmus eine Stacking Ordnung, die den minimal sichtbaren Rand aller Kreisscheiben maximiert. Der Algorithmus lässt sich in $O(n^2 \log n)$ Laufzeit implementieren.

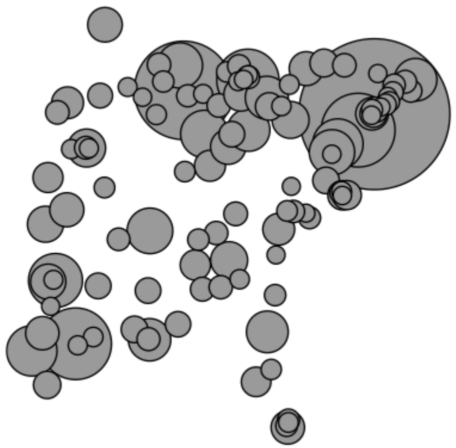
**Satz 3:** Falls kein Punkt in mehr als $O(1)$ Kreisen enthalten ist, kann die optimale Stacking Ordnung in $O(n \log n)$ Zeit berechnet werden.

# Proportional Symbol Maps in der Praxis


(a) Left-to-right by center.


(b) Left-to-right by leftmost.


(c) Large-to-small.


(d) Max-Min.

| Data sets | | Top-10 average | | Total boundary | |
|---|---|---|---|---|---|
| | | Absolute | (Relative) | Absolute | (Relative) |
| City 156 | Left-to-right by center | 0.00 | (0.00%) | 1405 | (57.76%) |
| | Left-to-right by leftmost | 2.14 | (21.48%) | 1711 | (74.84%) |
| | Large-to-small | 2.72 | (25.48%) | 1730 | (78.21%) |
| | Max-Min | 4.42 | (43.03%) | 1759 | (79.33%) |