Karlsruher Institut für Technologie

Algorithmische Geometrie

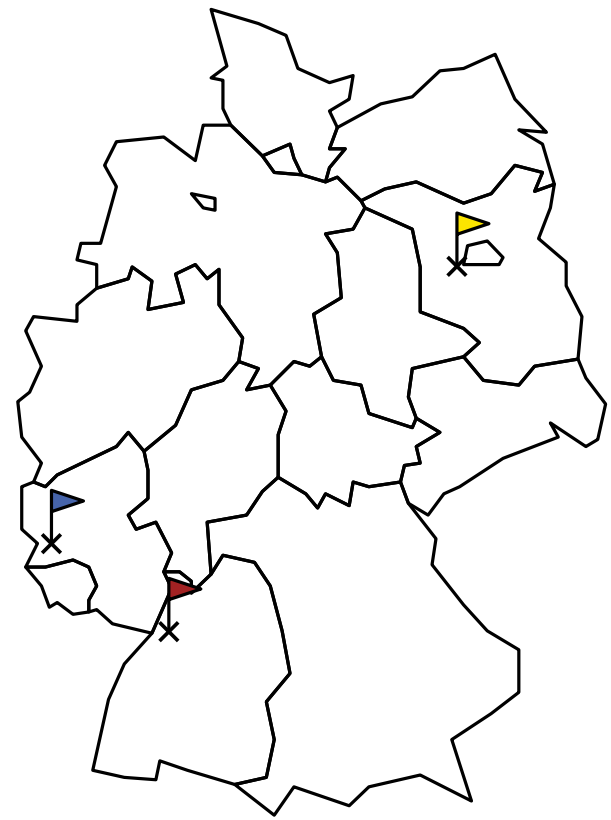
Aktivsession 4

Wo bin ich?

Problem: Point-Location
In welcher Facette eines geom. Graphen liegt ein gegebener Punkt p?

Statische Variante

- Graph G ist fest
- beantworte Anfragen für viele Punkte $p \in \mathbb{R}^2$

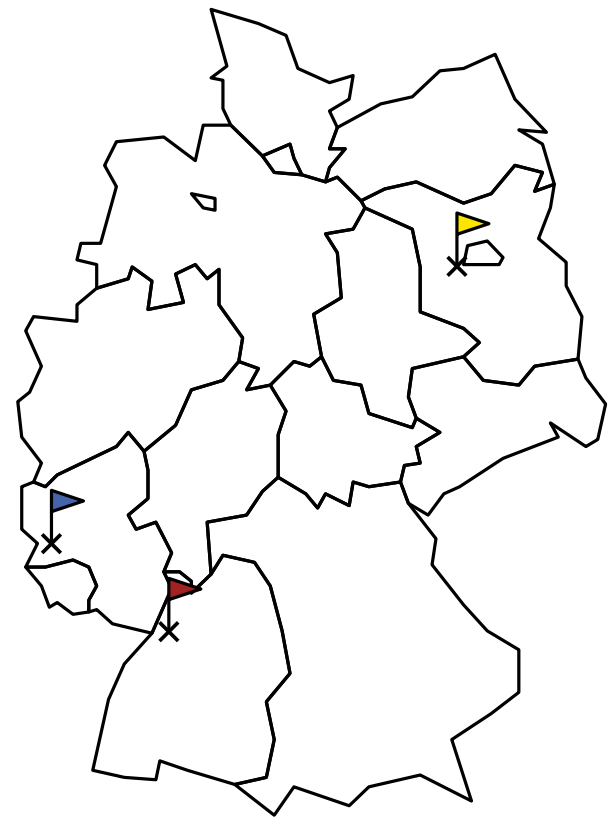


Wo bin ich?

Problem: Point-Location
In welcher Facette eines geom. Graphen liegt ein gegebener Punkt p?

Statische Variante

- Graph G ist fest
- beantworte Anfragen für viele Punkte $p \in \mathbb{R}^2$



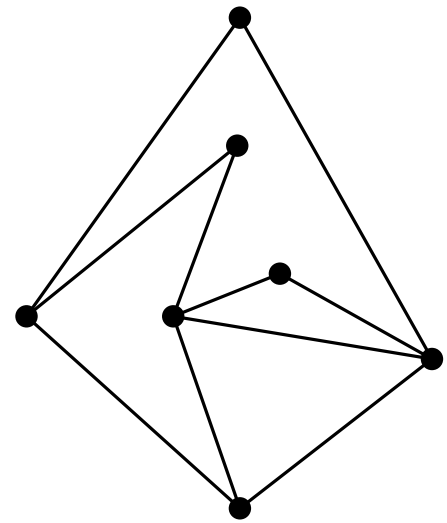
Freitag gesehen

- Sweep-Line Algo mit persistentem Suchbaum
  - aus „Wo ist $p = (p_x, p_y)$?“
  - wird „Wo war p_x zum Zeitpunkt p_y?“

Vorberechnung: $O(n \log n)$ **Anfragen:** $O(\log n)$

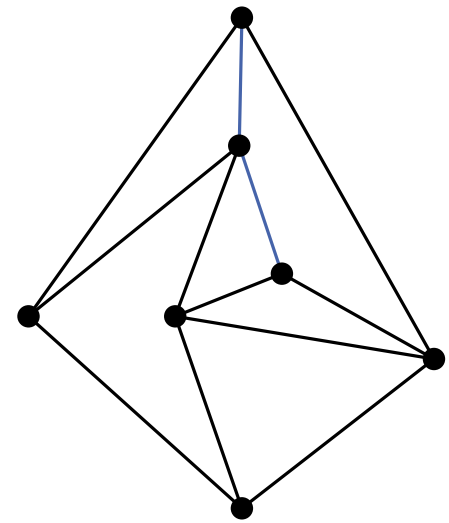
Speicherplatz: $O(n)$

Das geht bestimmt auch anders!



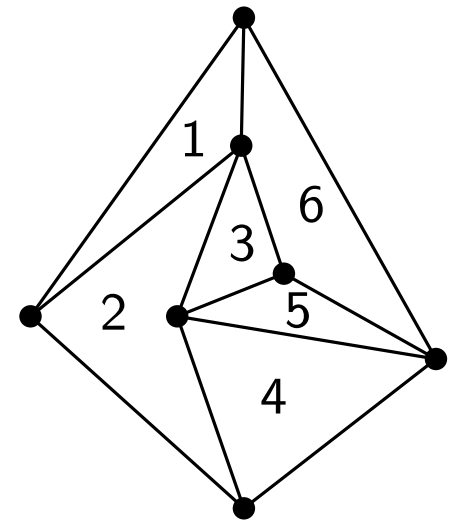
Das geht bestimmt auch anders!

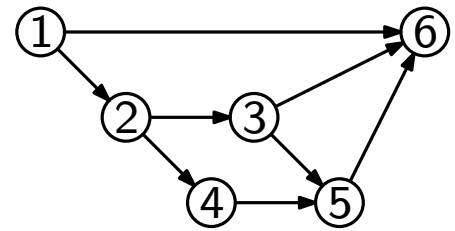
- mache jede Facette y-monoton

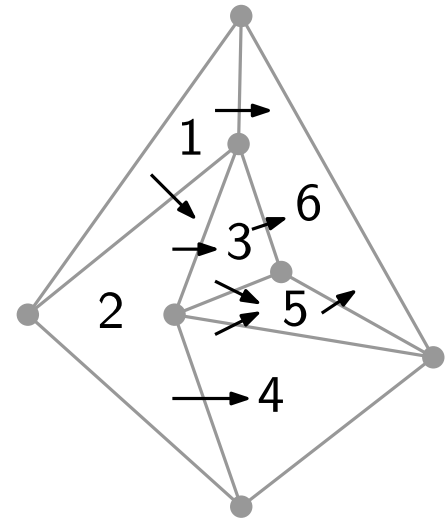


Das geht bestimmt auch anders!

- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)

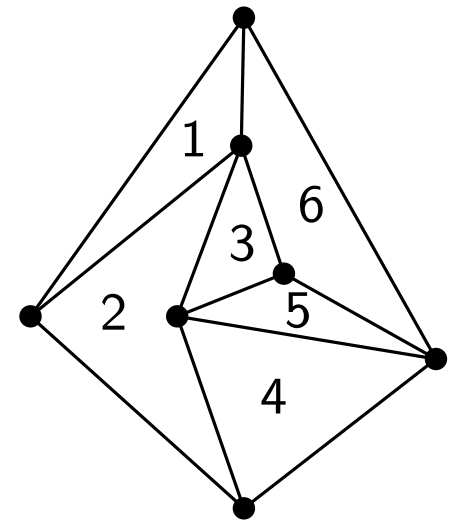


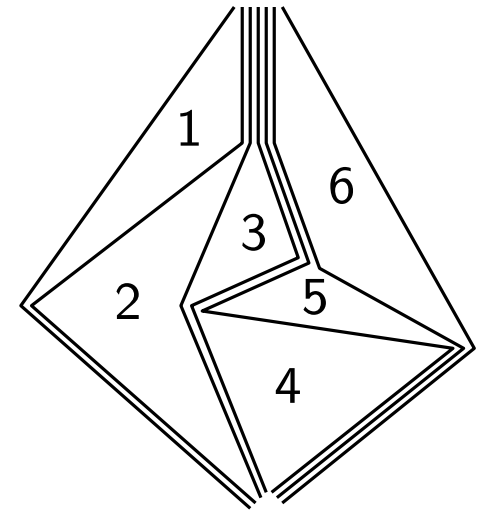




Das geht bestimmt auch anders!

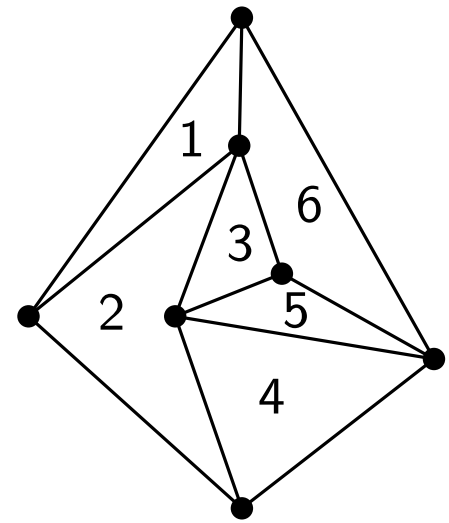
- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung

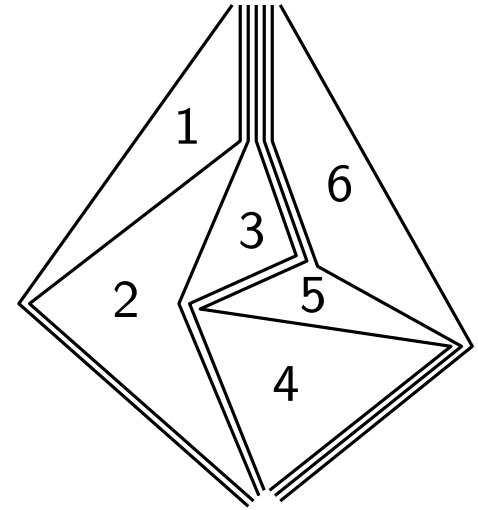




Das geht bestimmt auch anders!

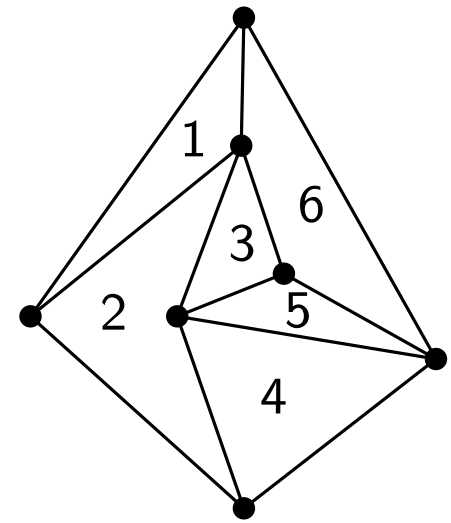
- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton

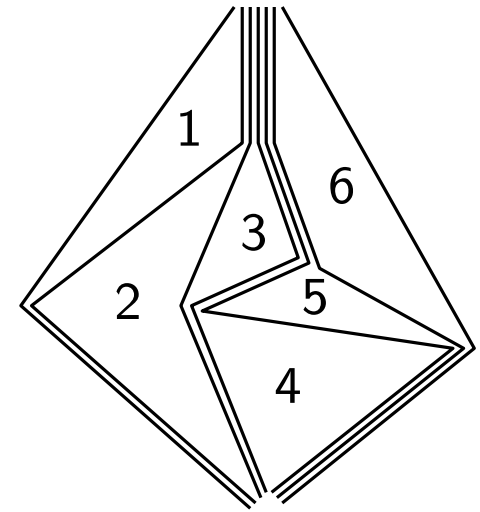




Das geht bestimmt auch anders!

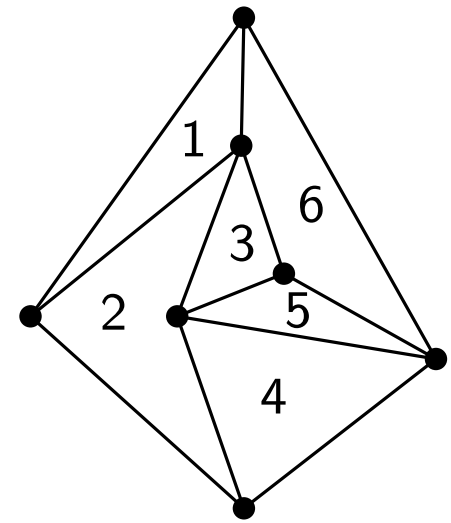
- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen

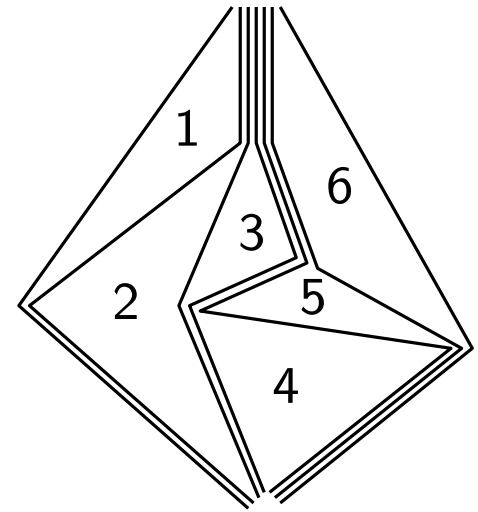




Das geht bestimmt auch anders!

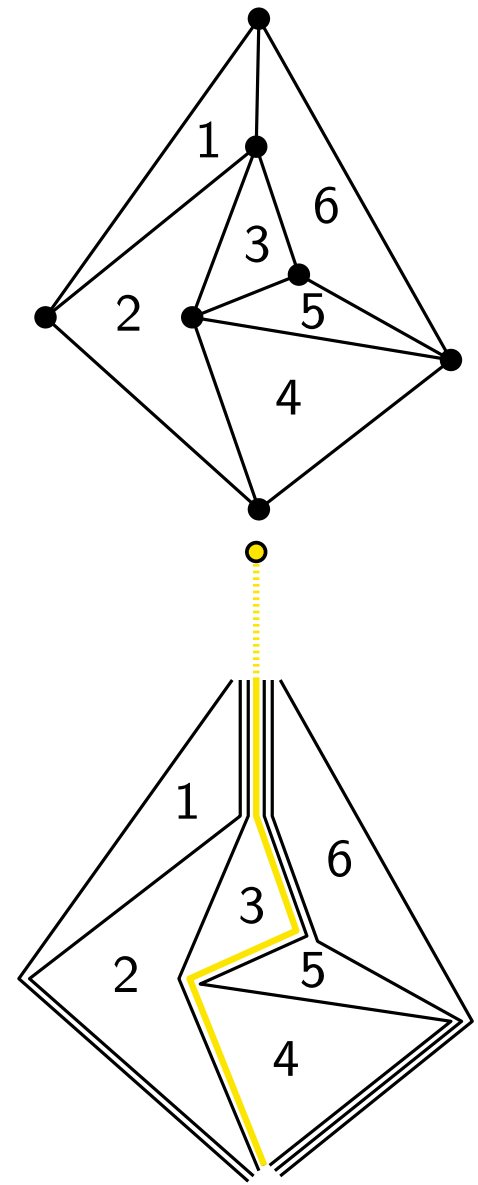
- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen
- binärer Suchbaum auf den Pfaden
- jeder Knoten speichert sortierten Teilpfad
  (Kanten aus Vorgängern werden nicht erneut gespeichert)





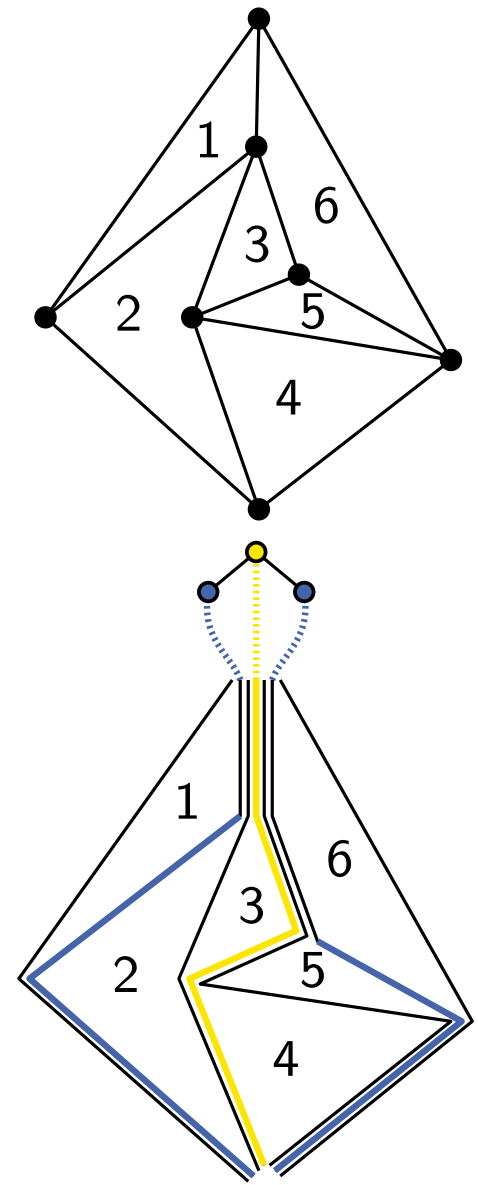
Das geht bestimmt auch anders!

- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen
- binärer Suchbaum auf den Pfaden
- jeder Knoten speichert sortierten Teilpfad
  (Kanten aus Vorgängern werden nicht erneut gespeichert)



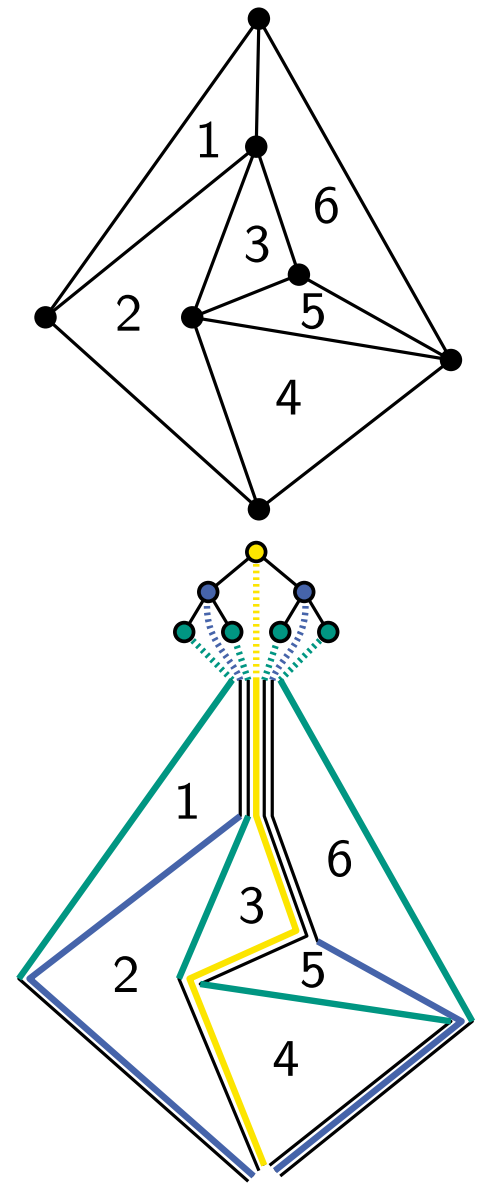
Das geht bestimmt auch anders!

- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen
- binärer Suchbaum auf den Pfaden
- jeder Knoten speichert sortierten Teilpfad
  (Kanten aus Vorgängern werden nicht erneut gespeichert)



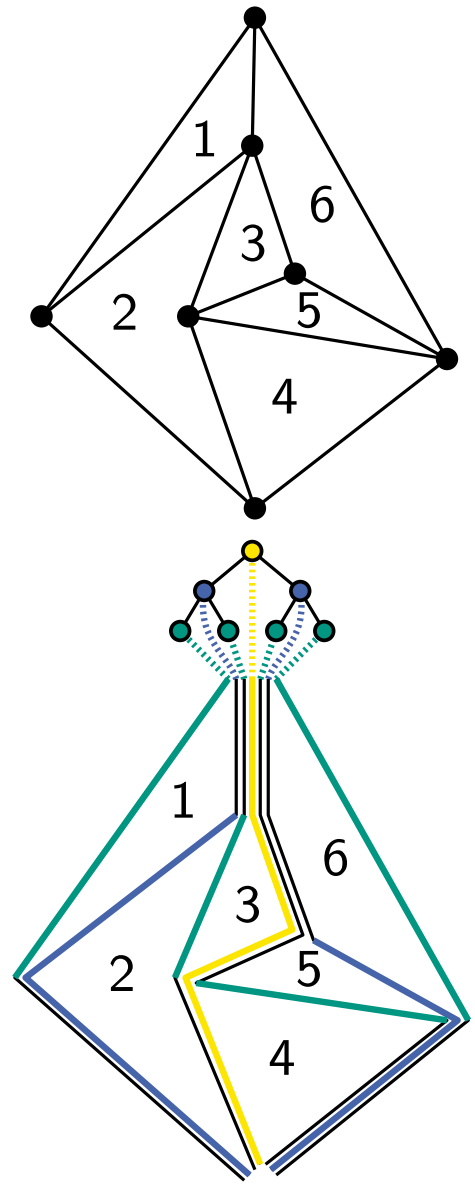
Das geht bestimmt auch anders!

- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen
- binärer Suchbaum auf den Pfaden
- jeder Knoten speichert sortierten Teilpfad
  (Kanten aus Vorgängern werden nicht erneut gespeichert)



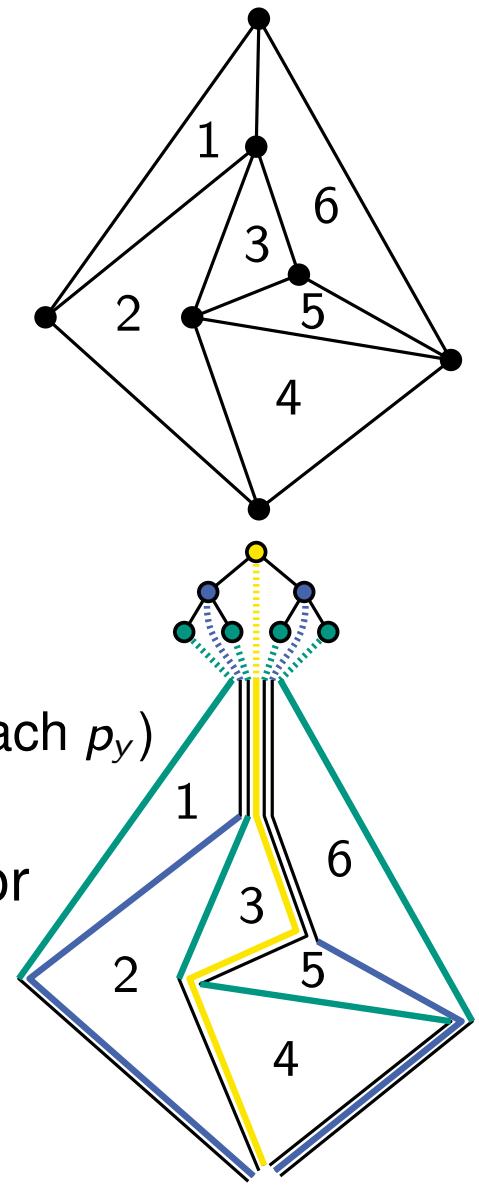
Das geht bestimmt auch anders!

- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen
- binärer Suchbaum auf den Pfaden
- jeder Knoten speichert sortierten Teilpfad
  (Kanten aus Vorgängern werden nicht erneut gespeichert)
- finde Punkt im Suchbaum:



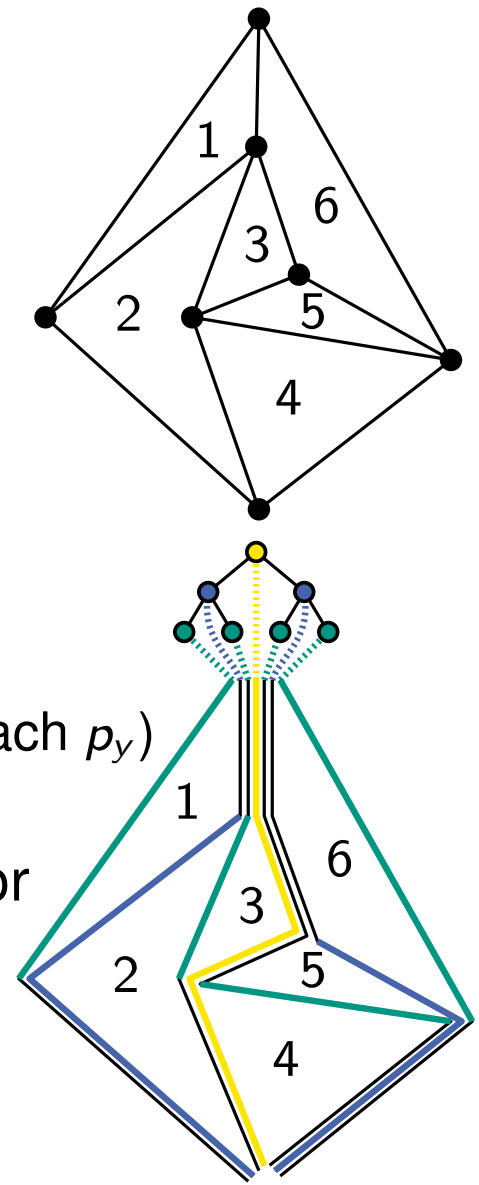
Das geht bestimmt auch anders!

- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen
- binärer Suchbaum auf den Pfaden
- jeder Knoten speichert sortierten Teilpfad
  (Kanten aus Vorgängern werden nicht erneut gespeichert)
- finde Punkt im Suchbaum:
  - pro Knoten: finde Kante e im Pfad neben p (Suche nach p_y)
  - laufe links/rechts wenn p links/rechts neben e
  - e existiert nicht $\rightarrow$ selbe Richtung wie ein Schritt zuvor



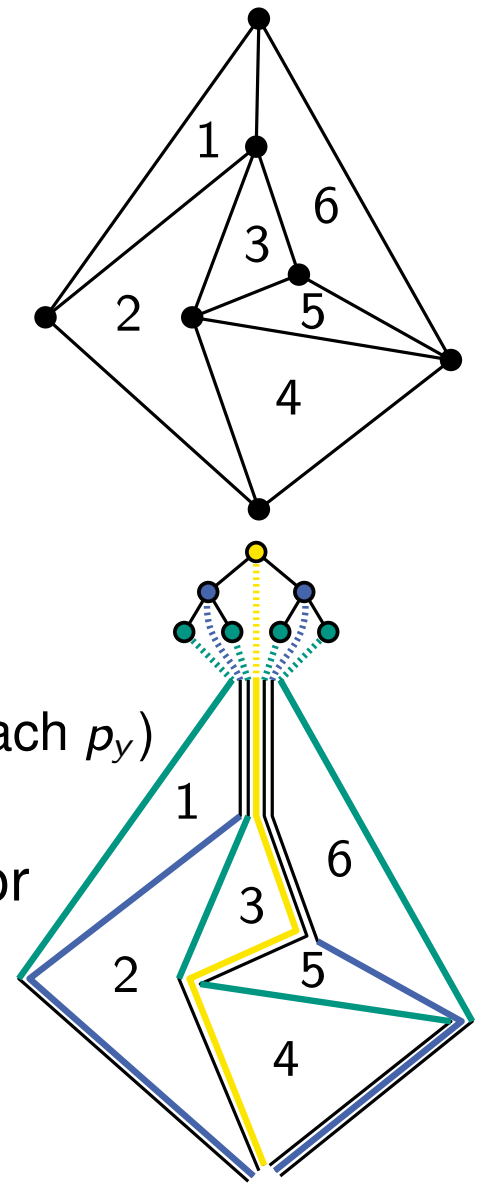
Das geht bestimmt auch anders!

- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen
- binärer Suchbaum auf den Pfaden
- jeder Knoten speichert sortierten Teilpfad
  (Kanten aus Vorgängern werden nicht erneut gespeichert)
- finde Punkt im Suchbaum:
  - pro Knoten: finde Kante e im Pfad neben p (Suche nach p_y)
  - laufe links/rechts wenn p links/rechts neben e
  - e existiert nicht $\rightarrow$ selbe Richtung wie ein Schritt zuvor
- Anfrage: $O(\log n)$ binäre Suchen $\rightarrow O(\log^2 n)$



Das geht bestimmt auch anders!

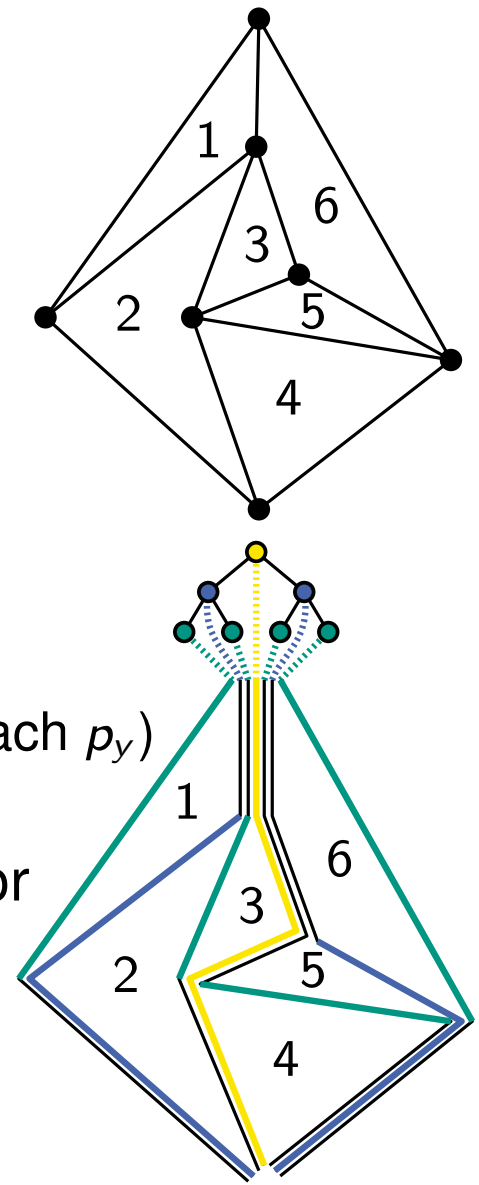
- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen
- binärer Suchbaum auf den Pfaden
- jeder Knoten speichert sortierten Teilpfad
  (Kanten aus Vorgängern werden nicht erneut gespeichert)
- finde Punkt im Suchbaum:
  - pro Knoten: finde Kante e im Pfad neben p (Suche nach p_y)
  - laufe links/rechts wenn p links/rechts neben e
  - e existiert nicht $\rightarrow$ selbe Richtung wie ein Schritt zuvor
- Anfrage: $O(\log n)$ binäre Suchen $\rightarrow O(\log^2 n)$
- fractional cascading $\rightarrow$ **Anfragen:** $O(\log n)$



Das geht bestimmt auch anders!

- mache jede Facette y-monoton
- nummeriere Facette von links nach rechts
  (das geht dank der y-Monotonie)
- zerlege Graph in Pfade bezüglich dieser Ordnung
- jeder dieser Pfade ist monoton
- Ziel: finde zwei konsekutive Pfade, mit p dazwischen
- binärer Suchbaum auf den Pfaden
- jeder Knoten speichert sortierten Teilpfad
  (Kanten aus Vorgängern werden nicht erneut gespeichert)
- finde Punkt im Suchbaum:
  - pro Knoten: finde Kante e im Pfad neben p (Suche nach p_y)
  - laufe links/rechts wenn p links/rechts neben e
  - e existiert nicht $\rightarrow$ selbe Richtung wie ein Schritt zuvor
- Anfrage: $O(\log n)$ binäre Suchen $\rightarrow O(\log^2 n)$
- fractional cascading $\rightarrow$ **Anfragen:** $O(\log n)$

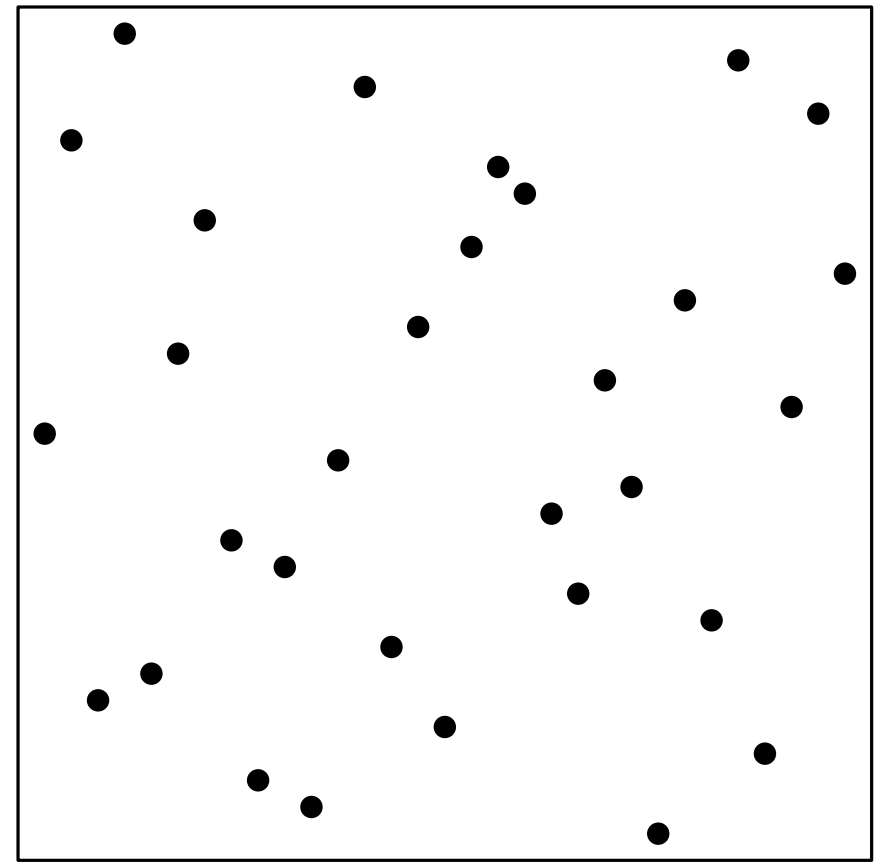
Vorberechnung: $O(n \log n)$ **Speicherplatz:** $O(n)$



kd-Bäume

Binäre Raumpartitionierung (in 2d)

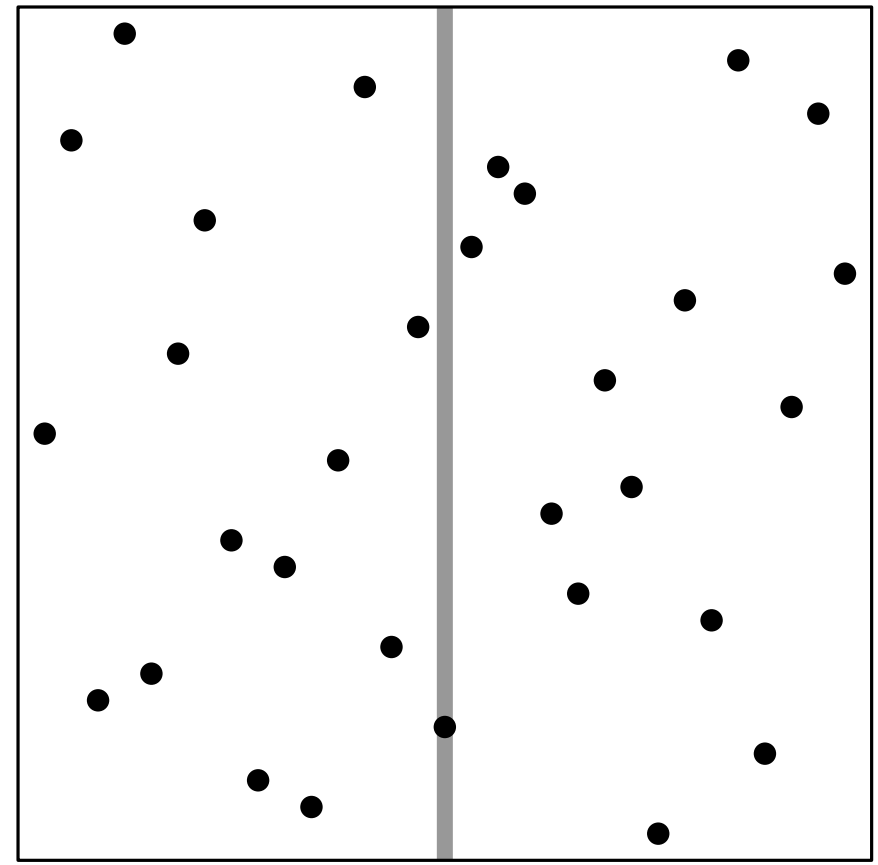
- Gegeben: Menge von Punkten



kd-Bäume

Binäre Raumpartitionierung (in 2d)

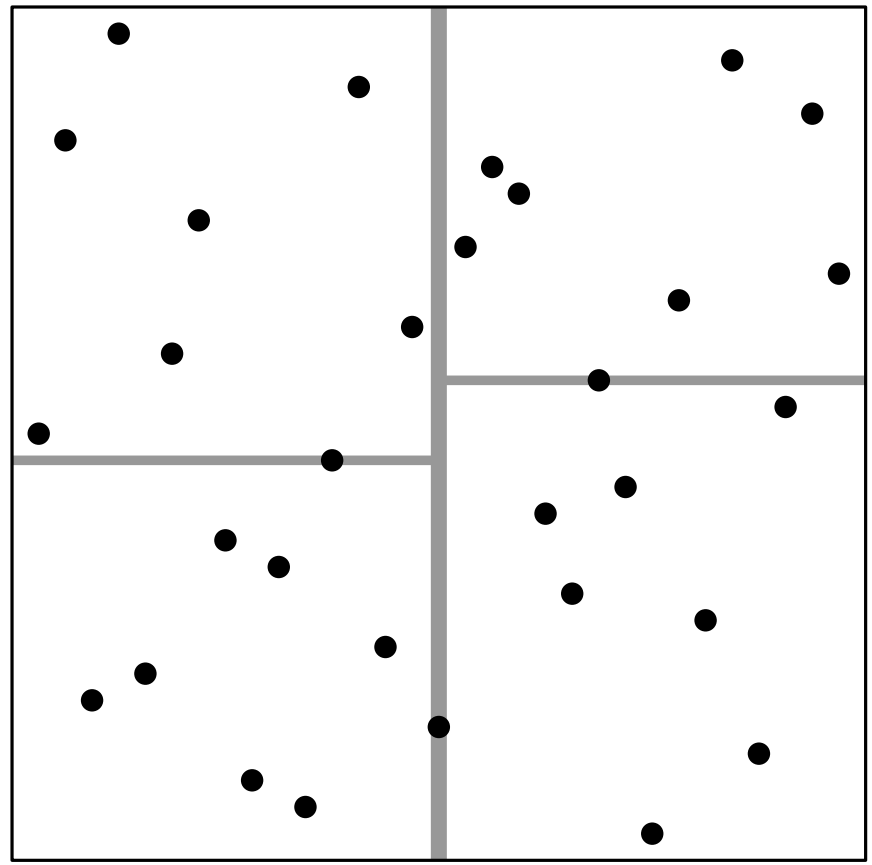
- Gegeben: Menge von Punkten
- halbiere bzgl. x-Koordinate



kd-Bäume

Binäre Raumpartitionierung (in 2d)

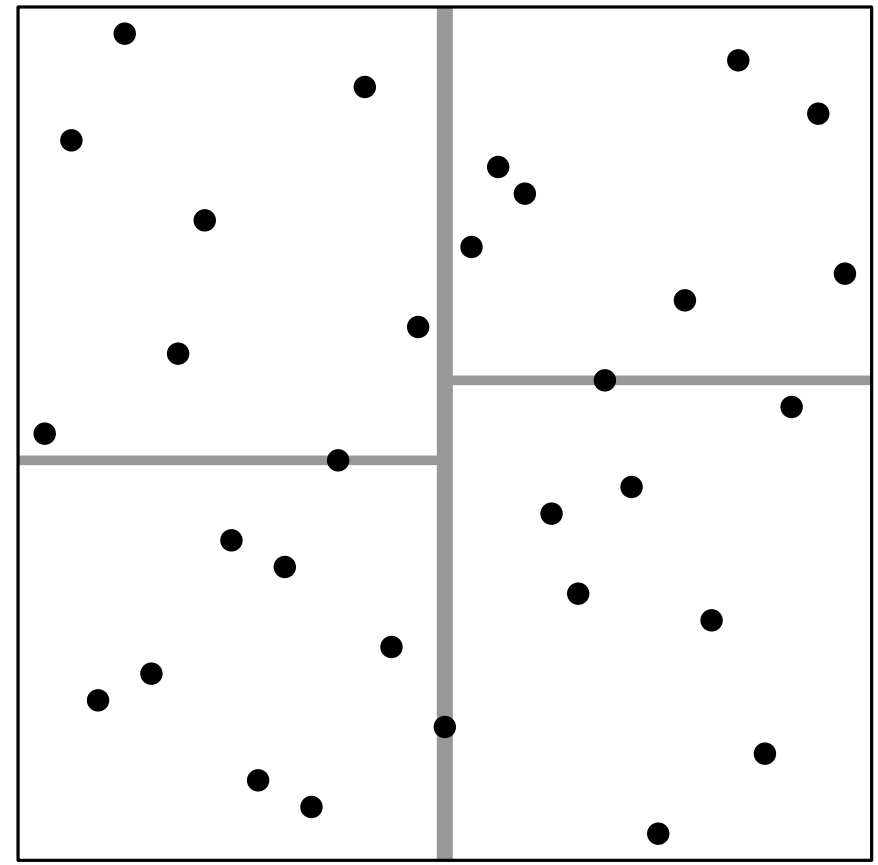
- Gegeben: Menge von Punkten
- halbiere bzgl. x-Koordinate
- halbiere jede Seite bzgl. y-Koordinate



kd-Bäume

Binäre Raumpartitionierung (in 2d)

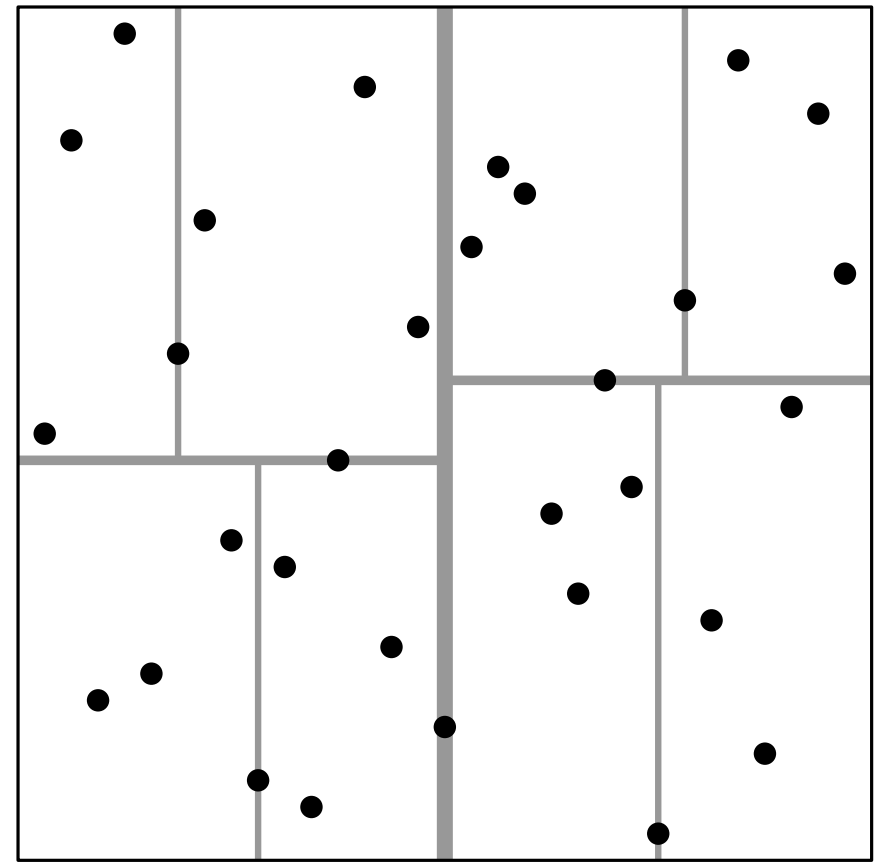
- Gegeben: Menge von Punkten
- halbiere bzgl. x-Koordinate
- halbiere jede Seite bzgl. y-Koordinate
- iteriere, bis jede Region nur noch $\Theta(1)$ Punkte enthält



kd-Bäume

Binäre Raumpartitionierung (in 2d)

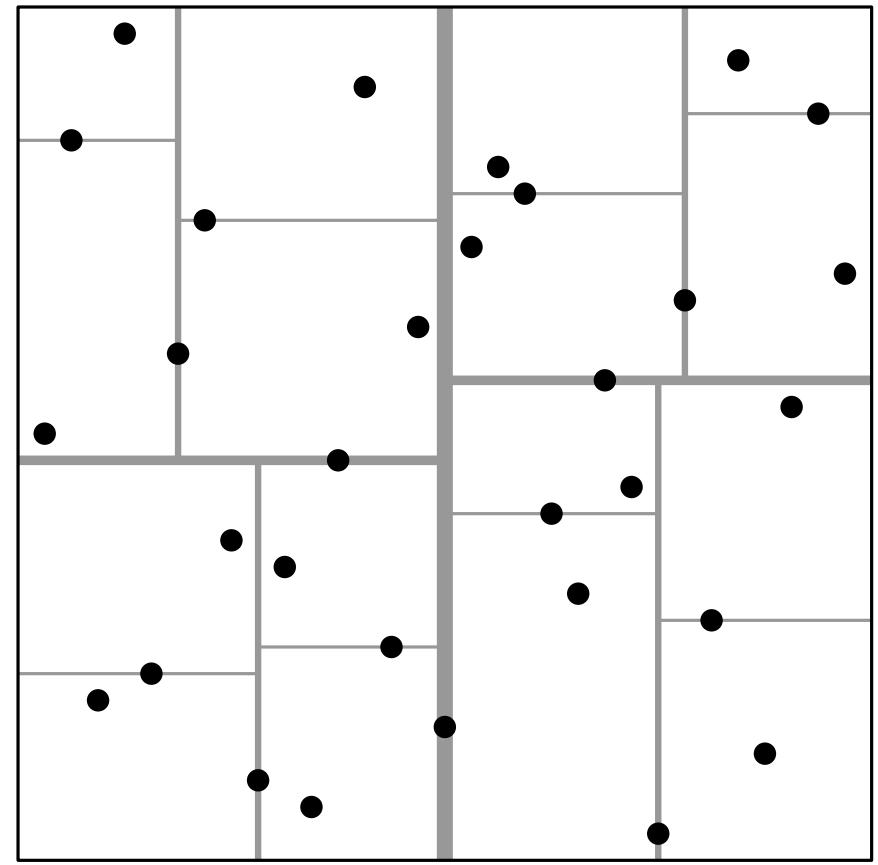
- Gegeben: Menge von Punkten
- halbiere bzgl. x-Koordinate
- halbiere jede Seite bzgl. y-Koordinate
- iteriere, bis jede Region nur noch $\Theta(1)$ Punkte enthält



kd-Bäume

Binäre Raumpartitionierung (in 2d)

- Gegeben: Menge von Punkten
- halbiere bzgl. x-Koordinate
- halbiere jede Seite bzgl. y-Koordinate
- iteriere, bis jede Region nur noch $\Theta(1)$ Punkte enthält

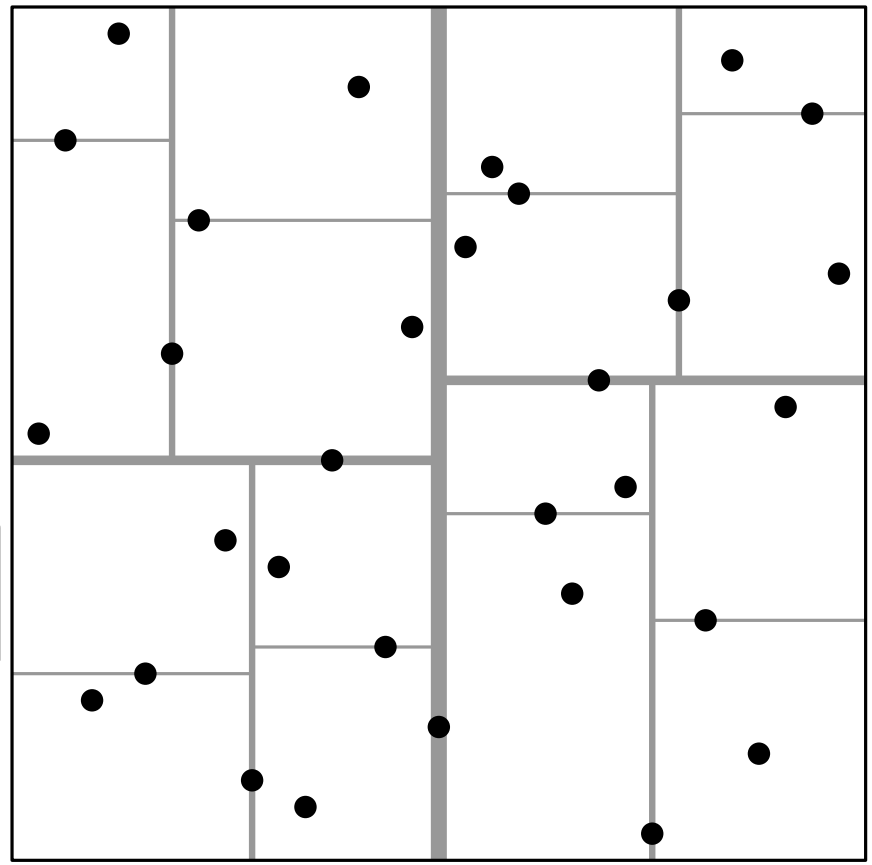


kd-Bäume

Binäre Raumpartitionierung (in 2d)

- Gegeben: Menge von Punkten
- halbiere bzgl. x-Koordinate
- halbiere jede Seite bzgl. y-Koordinate
- iteriere, bis jede Region nur noch $\Theta(1)$ Punkte enthält

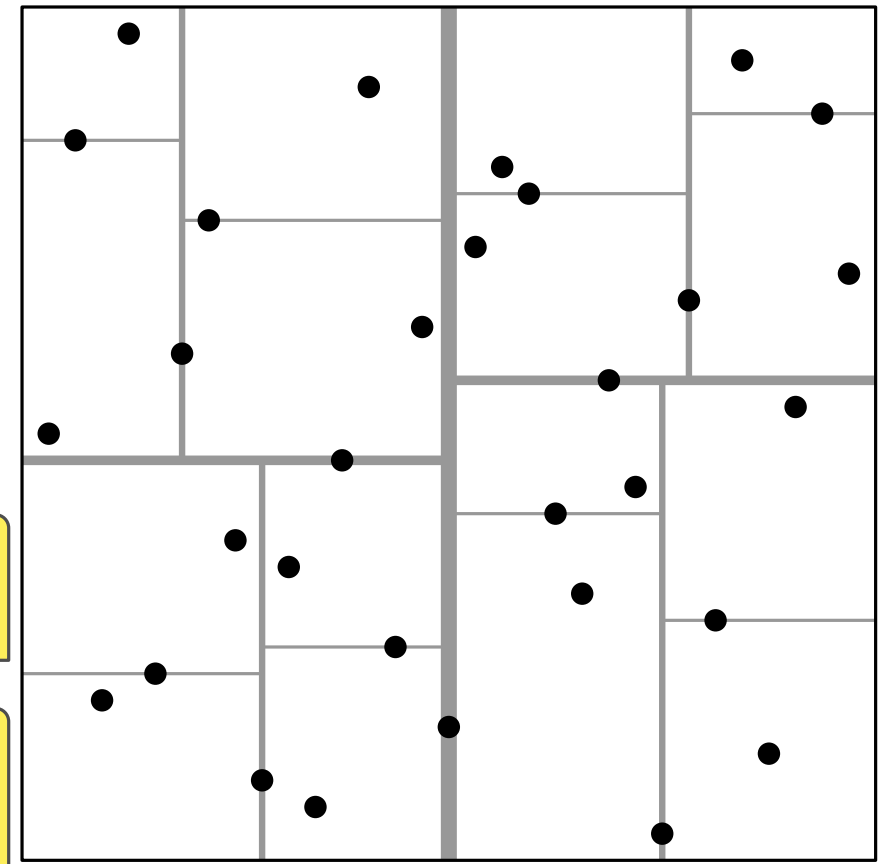
Wie schnell geht der Aufbau?
Wie viel Speicher brauchen wir?



kd-Bäume

Binäre Raumpartitionierung (in 2d)

- Gegeben: Menge von Punkten
- halbiere bzgl. x-Koordinate
- halbiere jede Seite bzgl. y-Koordinate
- iteriere, bis jede Region nur noch $\Theta(1)$ Punkte enthält



Wie schnell geht der Aufbau?
Wie viel Speicher brauchen wir?

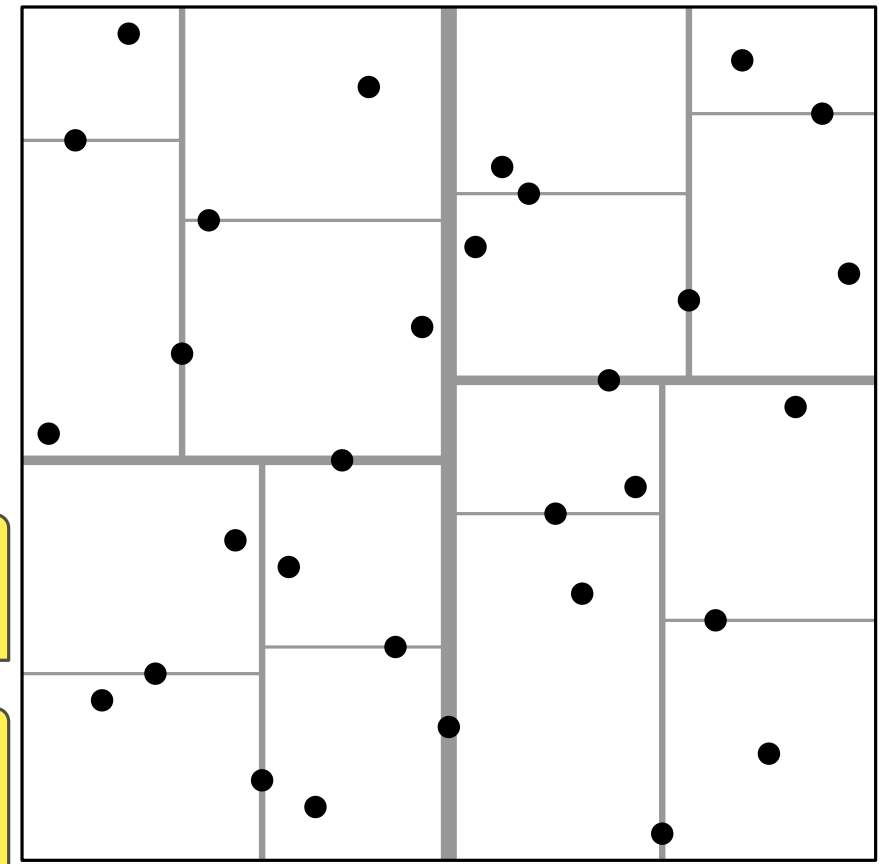
Wie können wir damit Bereichsanfragen beantworten?

(Beispielquery: finde Punkte in einem Rechteck)

kd-Bäume

Binäre Raumpartitionierung (in 2d)

- Gegeben: Menge von Punkten
- halbiere bzgl. x-Koordinate
- halbiere jede Seite bzgl. y-Koordinate
- iteriere, bis jede Region nur noch $\Theta(1)$ Punkte enthält



Wie schnell geht der Aufbau?
Wie viel Speicher brauchen wir?

Wie können wir damit Bereichsanfragen beantworten?
(Beispielquery: finde Punkte in einem Rechteck)

Wie teuer ist eine orthogonale Bereichsanfrage im Worst Case?

Plan für Weihnachten

Montag	Di	Mi	Do	Freitag
13 Übung	14	15	16	17 Vorlesung
Übungsblatt 4				
20 Aktiv	21	22	23	24
Übungsblatt 5				
27	28	29	30	31
3	4	5	6	7 Vorlesung
10 Übung	11	12	13	14 Vorlesung
Übungsblatt 5				

Plan für Weihnachten

Montag	Di	Mi	Do	Freitag
13 Übung	14	15	16	17 Vorlesung
Übungsblatt 4				
20 Aktiv	21	22	23	24
Übungsblatt 5				
27	28	29	30	31
3	4	5	6	7 Vorlesung
10 Übung	11	12	13	14 Vorlesung
Übungsblatt 5				

Computational Origami

Plan für Weihnachten

Montag	Di	Mi	Do	Freitag
13 Übung	14	15	16	17 Vorlesung
Übungsblatt 4				
20 Aktiv	21	22	23	24
	Übungsblatt 5			
27	28	29	30	31
3	4	5	6	7 Vorlesung
10 Übung	11	12	13	14 Vorlesung
Übungsblatt 5				

Computational Origami

Weihnachtsvorlesung (TGI)

- Donnerstag 23.12., 12 Uhr
- Fritz-Haller-HS