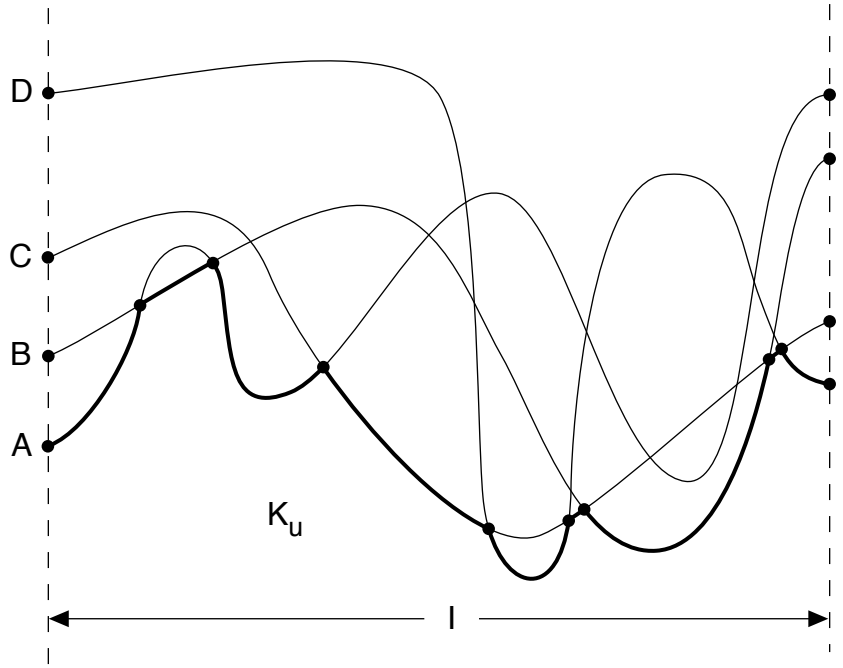
Untere Konturen

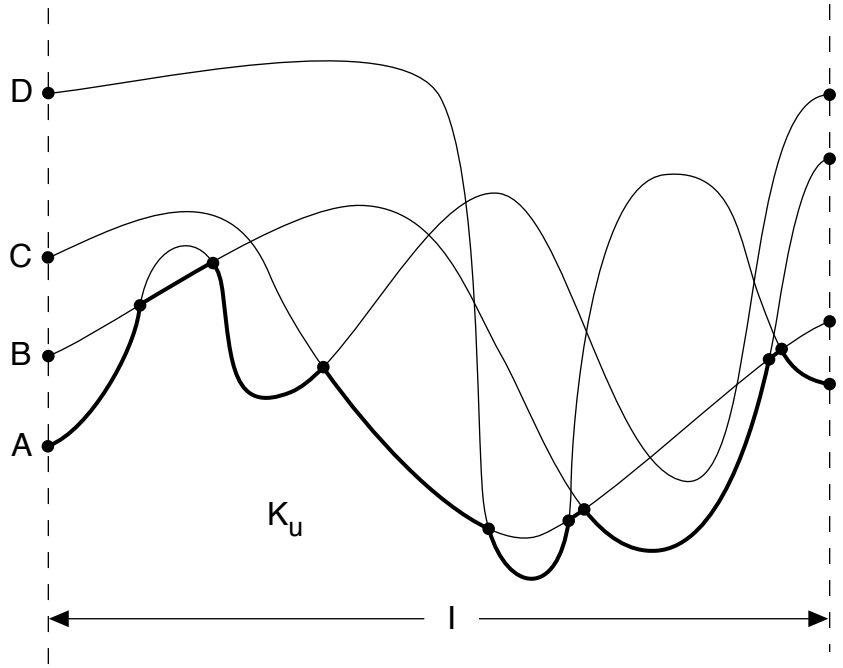
Elmar Langetepe
University of Bonn

Berechnung Kontur! Komplexität!

Berechnung Kontur! Komplexität!



Berechnung Kontur! Komplexität!



Divide and Conquer mit Sweep im Merge-Schritt!

$O(\lambda_s(n) \log n)$! Wie groß ist $\lambda_s(n)$?

$\lambda_s(n)$, kombinatorische Definition

$\lambda_s(n)$, kombinatorische Definition

Alphabet $\Sigma = \{A, B, C, \ldots\}$ über n Buchstaben.

$\lambda_s(n)$, kombinatorische Definition

Alphabet $\Sigma = \{A, B, C, \ldots\}$ über n Buchstaben.

Davenport–Schinzel–Sequenz der Ordnung s

$\lambda_s(n)$, kombinatorische Definition

Alphabet $\Sigma = \{A, B, C, \ldots\}$ über n Buchstaben.

Davenport–Schinzel–Sequenz der Ordnung s

- Wort w über Σ

$\lambda_s(n)$, kombinatorische Definition

Alphabet $\Sigma = \{A, B, C, \ldots\}$ über n Buchstaben.

Davenport–Schinzel–Sequenz der Ordnung s

- Wort w über Σ
- in w keine benachbarten Buchstaben gleich

$\lambda_s(n)$, kombinatorische Definition

Alphabet $\Sigma = \{A, B, C, \ldots\}$ über n Buchstaben.

Davenport–Schinzel–Sequenz der Ordnung s

- Wort w über Σ
- in w keine benachbarten Buchstaben gleich
- keine zwei verschiedenen Buchstaben wechseln mehr als s mal

$\lambda_s(n)$, kombinatorische Definition

Alphabet $\Sigma = \{A, B, C, \ldots\}$ über n Buchstaben.

Davenport–Schinzel–Sequenz der Ordnung s

- Wort w über Σ
- in w keine benachbarten Buchstaben gleich
- keine zwei verschiedenen Buchstaben wechseln mehr als s mal

Bsp.: ABRAKADABRA;

$\lambda_s(n)$, kombinatorische Definition

Alphabet $\Sigma = \{A, B, C, \ldots\}$ über n Buchstaben.

Davenport–Schinzel–Sequenz der Ordnung s

- Wort w über Σ
- in w keine benachbarten Buchstaben gleich
- keine zwei verschiedenen Buchstaben wechseln mehr als s mal

Bsp.: ABRAKADABRA; max. 4 Wechsel (A und B)

$\lambda_s(n)$, kombinatorische Definition

Alphabet $\Sigma = \{A, B, C, \ldots\}$ über n Buchstaben.

Davenport–Schinzel–Sequenz der Ordnung s

- Wort w über Σ
- in w keine benachbarten Buchstaben gleich
- keine zwei verschiedenen Buchstaben wechseln mehr als s mal

Bsp.: ABRAKADABRA; max. 4 Wechsel (A und B)

Zu zeigen: $\lambda_s(n)$ ist maximale Länge eines solchen Wortes

Davenport-Schinzel-Sequenzen: Komplexität

Davenport-Schinzel-Sequenzen: Komplexität

Fast linear!!

Davenport-Schinzel-Sequenzen: Komplexität

Fast linear!! (Ohne Beweis!)

$$
\begin{aligned}
\lambda_1(n) &= n \\
\lambda_2(n) &= 2n-1 \\
\lambda_3(n) &\in \Theta(n\,\alpha(n)) \\
\lambda_4(n) &\in \Theta(n\cdot 2^{\alpha(n)}) \\
\lambda_s(n) &\in O(n\log^*(n)) \in O(n^2)
\end{aligned}
$$

Davenport-Schinzel-Sequenzen: Komplexität

Fast linear!! (Ohne Beweis!)

$$
\begin{aligned}
\lambda_1(n) &= n \\
\lambda_2(n) &= 2n-1 \\
\lambda_3(n) &\in \Theta(n\,\alpha(n)) \\
\lambda_4(n) &\in \Theta(n\cdot 2^{\alpha(n)}) \\
\lambda_s(n) &\in O(n\log^*(n)) \in O(n^2)
\end{aligned}
$$

$\alpha(n)$ Inverse Ackermann Fkt.

Davenport-Schinzel-Sequenzen: Komplexität

Fast linear!! (Ohne Beweis!)

$$
\begin{aligned}
\lambda_1(n) &= n \\
\lambda_2(n) &= 2n-1 \\
\lambda_3(n) &\in \Theta(n\,\alpha(n)) \\
\lambda_4(n) &\in \Theta(n \cdot 2^{\alpha(n)}) \\
\lambda_s(n) &\in O(n \log^*(n)) \in O(n^2)
\end{aligned}
$$

$\alpha(n)$ Inverse Ackermann Fkt.

$\log^*(n)$

Beweis: Untere Kontur/DSS

Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

Beweis: Untere Kontur/DSS

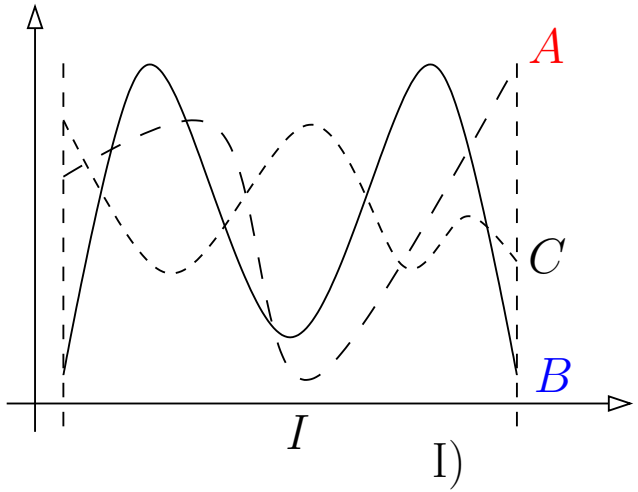
Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

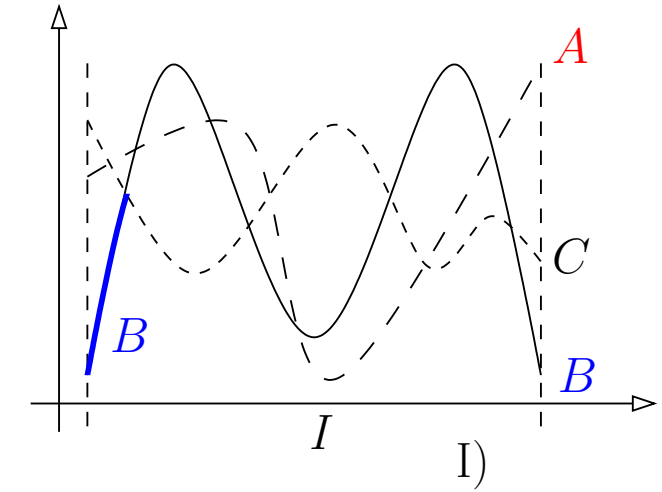
I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

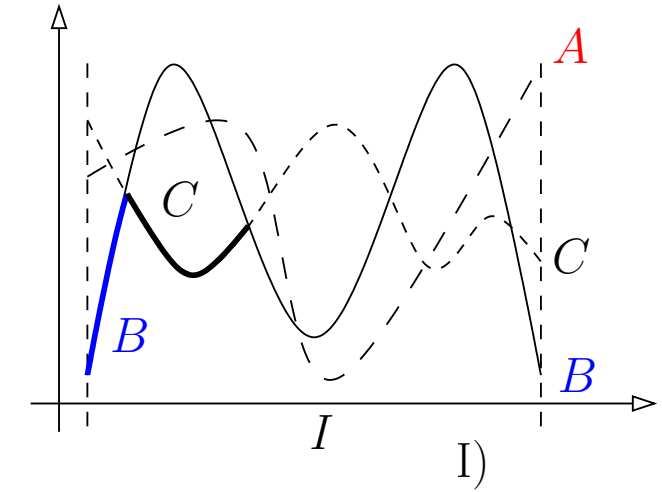


B

Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

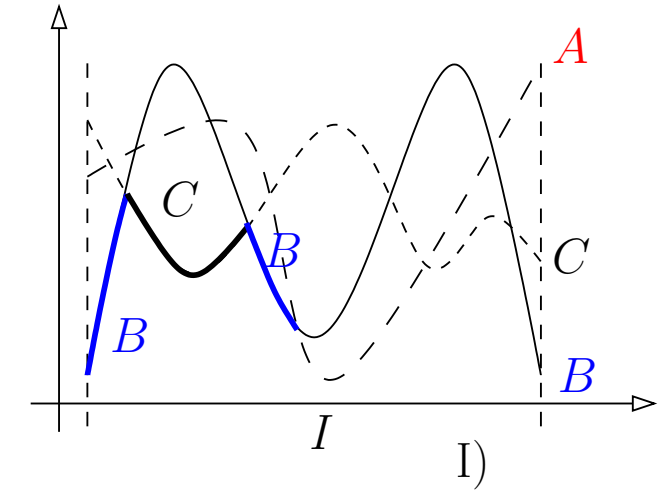


B C

Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

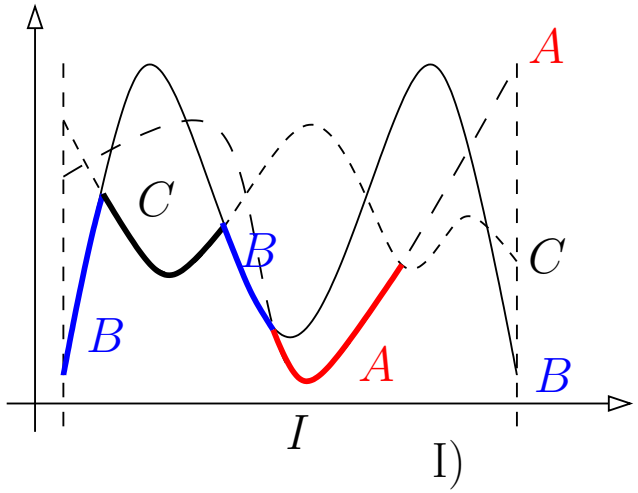


$B\ C\ B$

Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

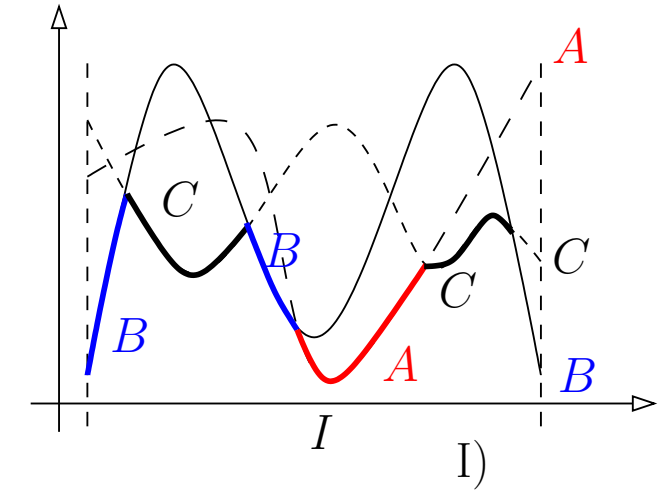


$B\ C\ B\ A$

Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

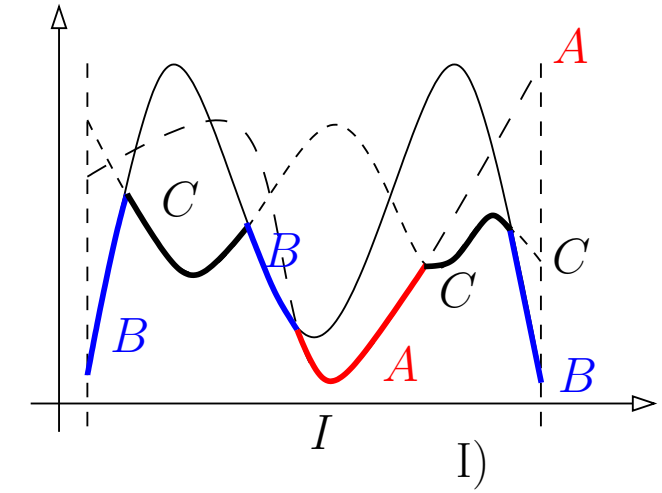


$B\ C\ B\ A\ C$

Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

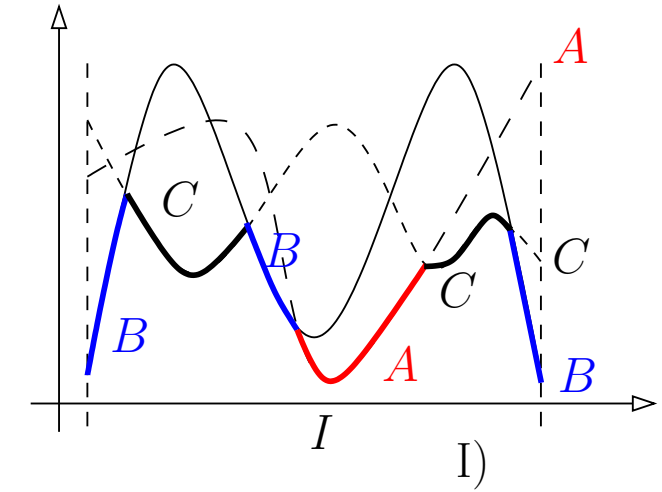


$B\ C\ B\ A\ C\ B$

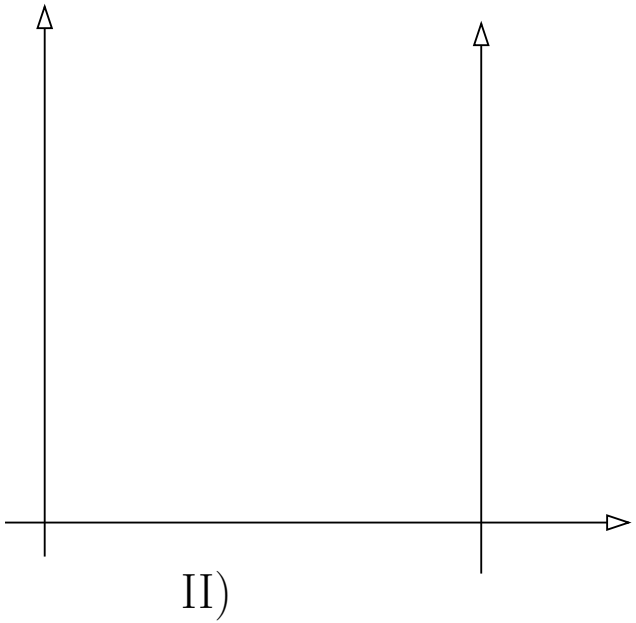
Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



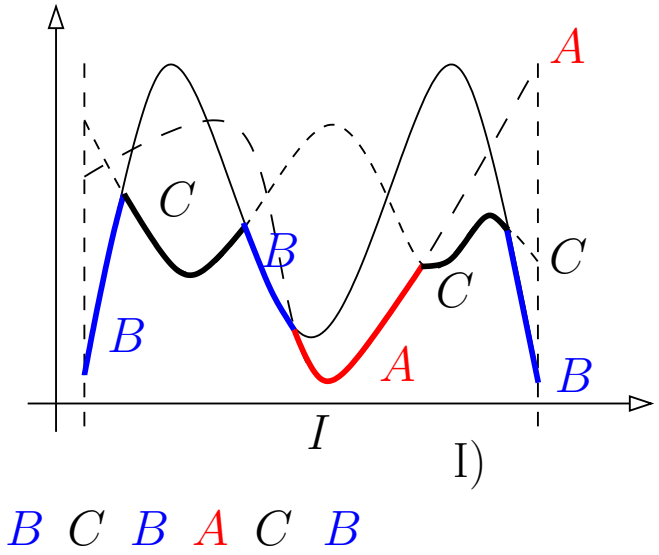
$B\ C\ B\ A\ C\ B$

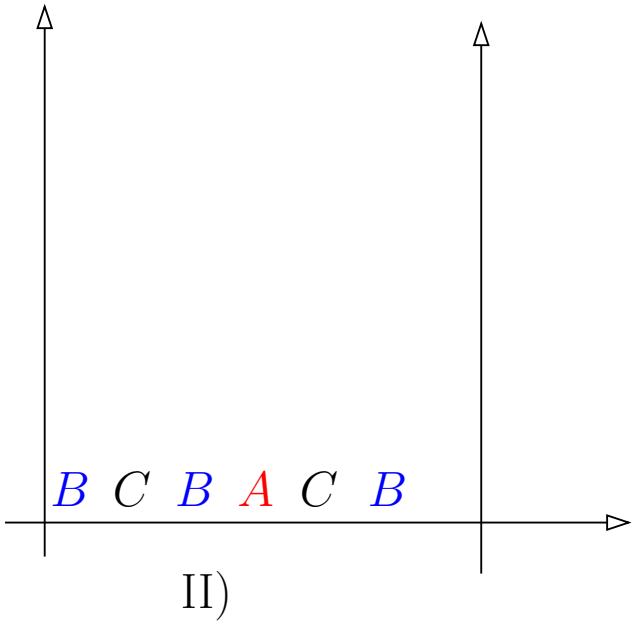


Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

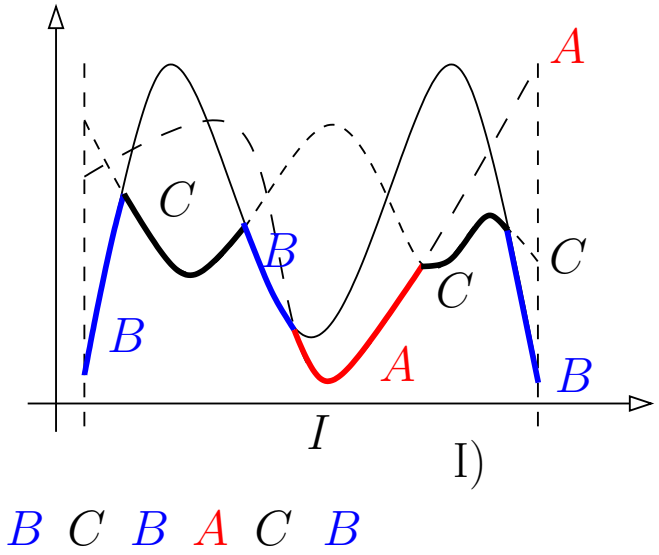


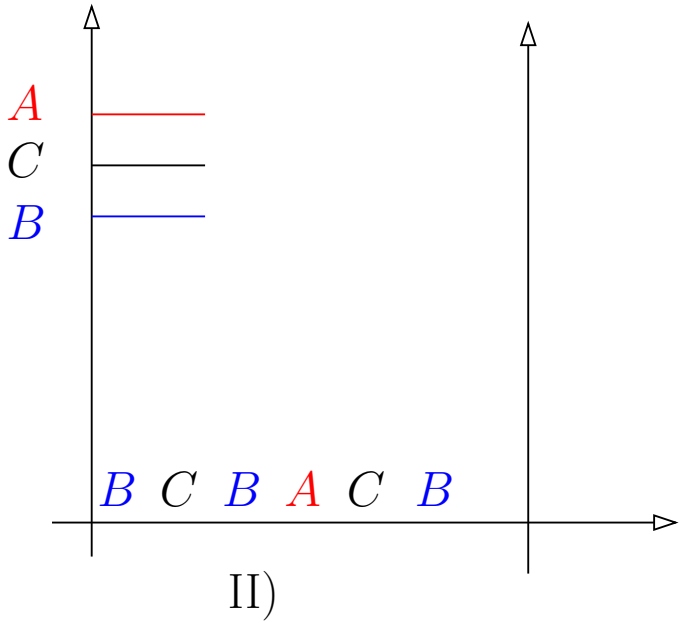


Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

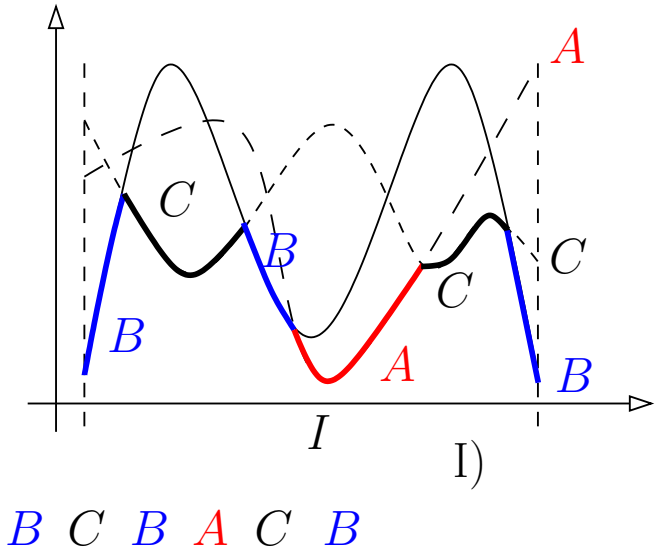


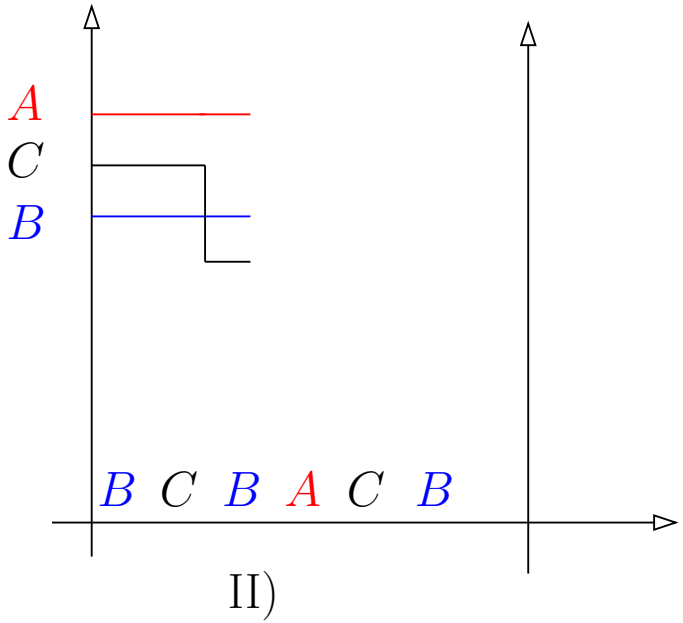


Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

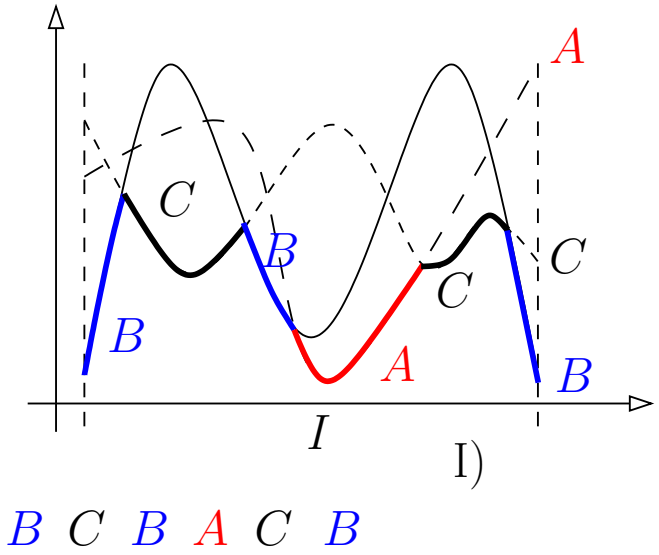


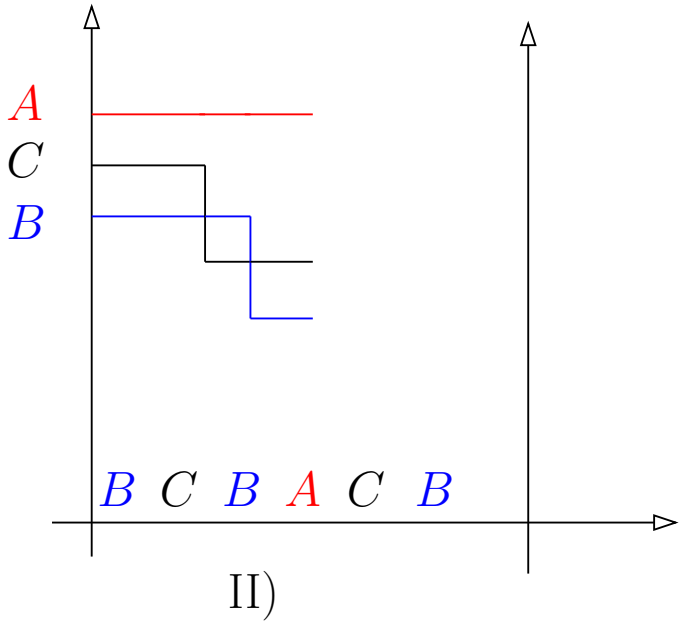


Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

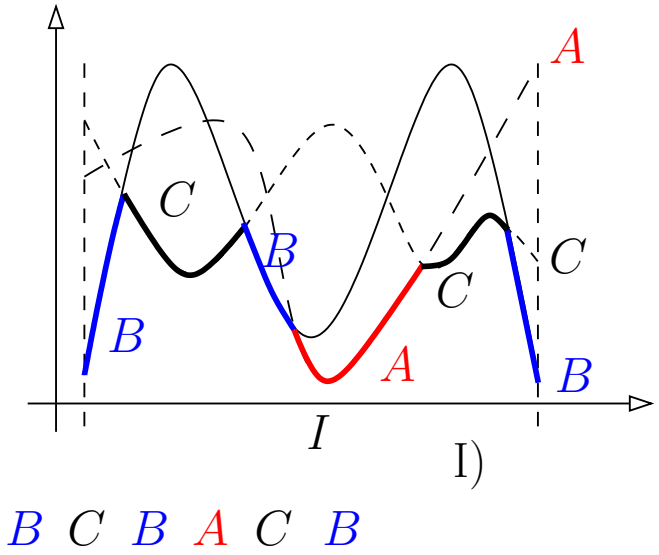


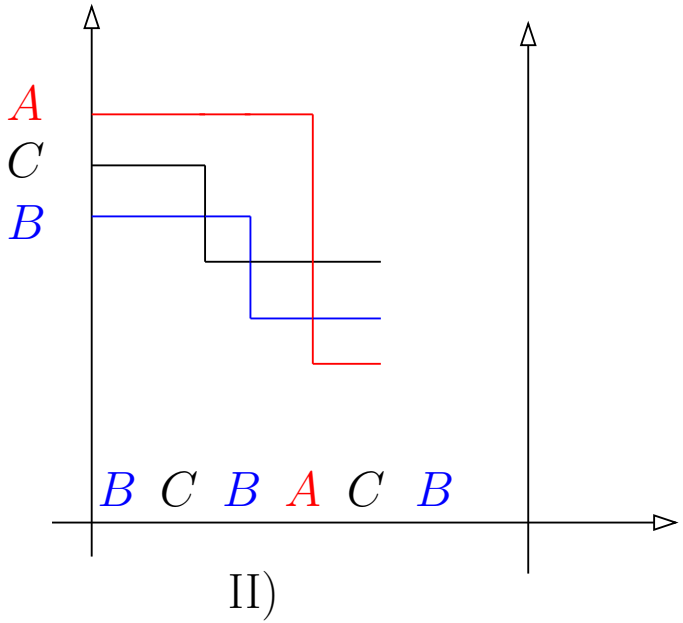


Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

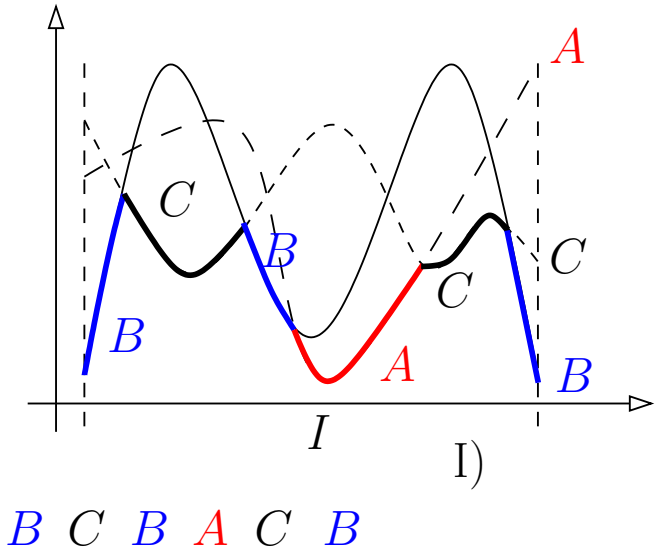


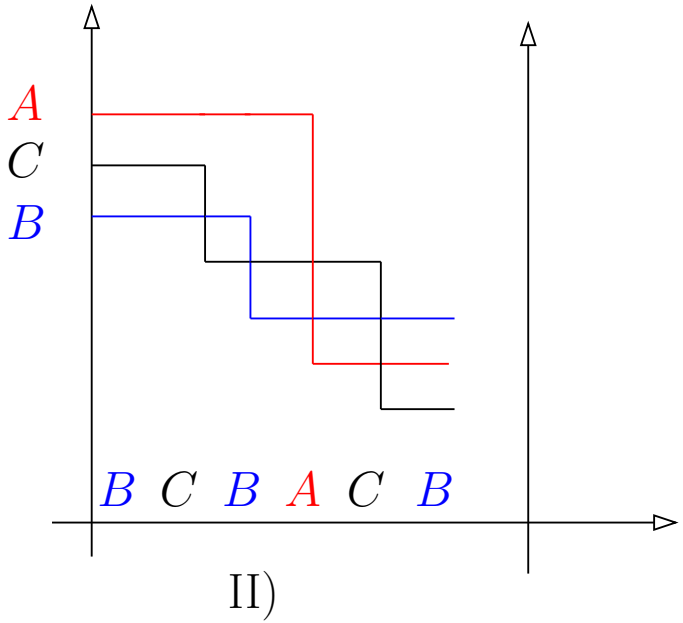


Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

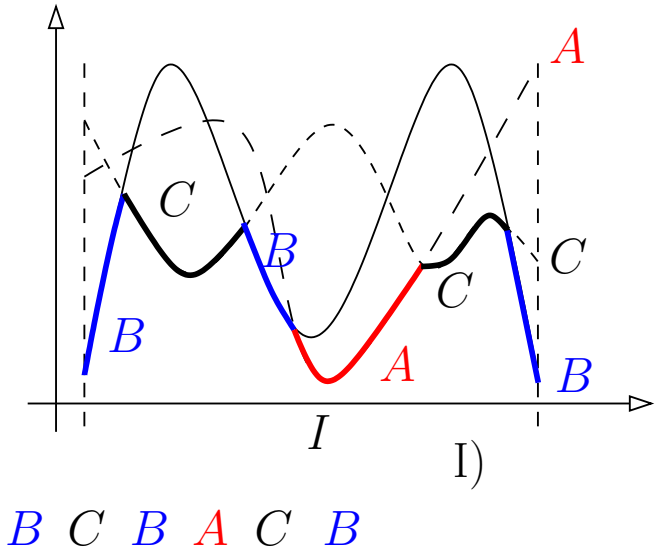


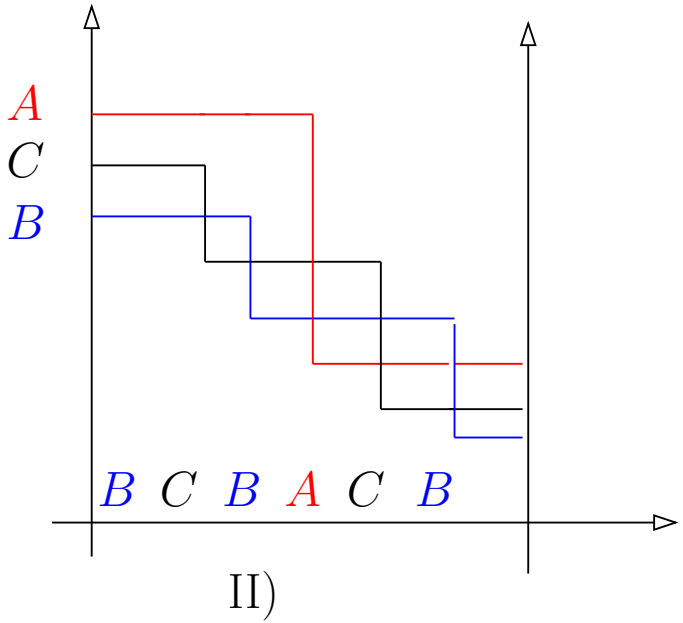


Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen

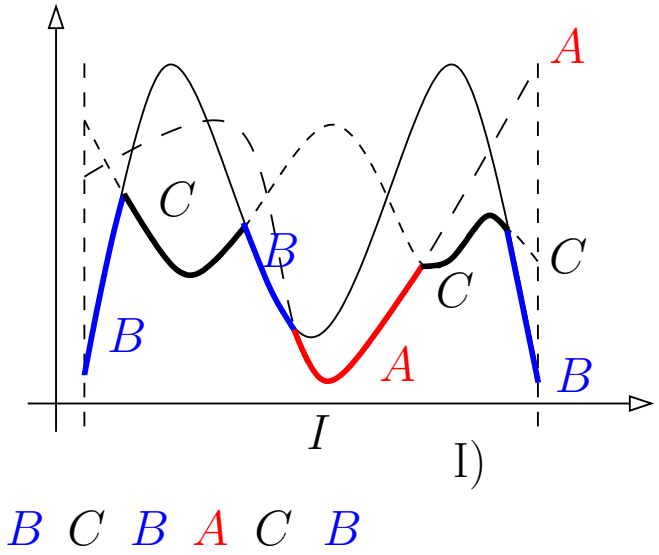


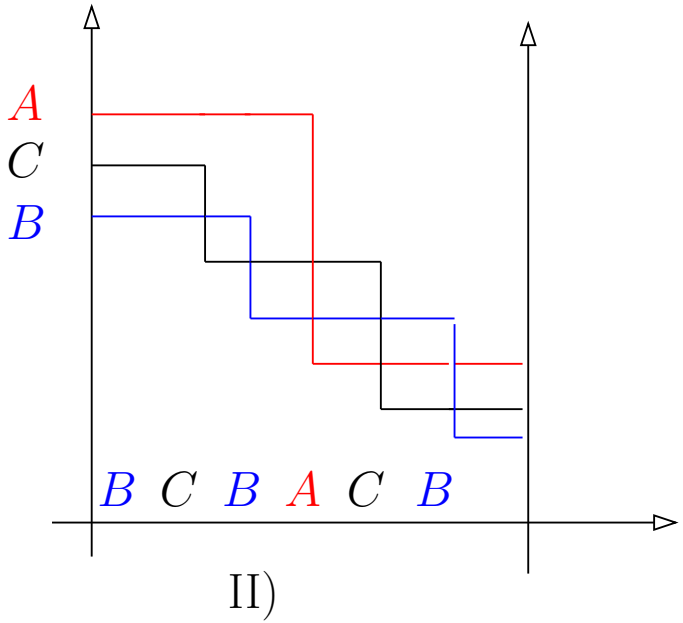


Beweis: Untere Kontur/DSS

Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen





Ergebnisse

Ergebnisse

Korollar 2.15 Die untere Kontur von n Liniensegmenten über einem gemeinsamen Intervall kann in $O(n \log n)$ mit Platz $O(n)$ berechnet werden.

Ergebnisse

Korollar 2.15 Die untere Kontur von n Liniensegmenten über einem gemeinsamen Intervall kann in $O(n \log n)$ mit Platz $O(n)$ berechnet werden.

Lemma 2.12: Für alle $s, n \geq 1$ gilt $2\lambda_s(n) \leq \lambda_s(2n)$

Ergebnisse

Korollar 2.15 Die untere Kontur von n Liniensegmenten über einem gemeinsamen Intervall kann in $O(n \log n)$ mit Platz $O(n)$ berechnet werden.

Lemma 2.12: Für alle $s, n \geq 1$ gilt $2\lambda_s(n) \leq \lambda_s(2n)$

$\lambda_1(n) = n$

Funktionen über Teilintervalle

Funktionen über Teilintervalle

Aussage: Komplexität der Kontur von Funktionen die über gesamtem Intervall definiert sind 1) oder nur über ein Teilintervall 2).

1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

Funktionen über Teilintervalle

Aussage: Komplexität der Kontur von Funktionen die über gesamtem Intervall definiert sind 1) oder nur über ein Teilintervall 2).

1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

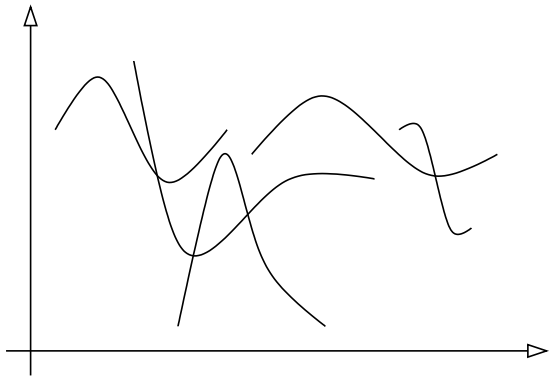
2. Verlängern auf gesamtes Intervall, dann 1. verwenden,

Funktionen über Teilintervalle

Aussage: Komplexität der Kontur von Funktionen die über gesamtem Intervall definiert sind 1) oder nur über ein Teilintervall 2).

1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

2. Verlängern auf gesamtes Intervall, dann 1. verwenden,



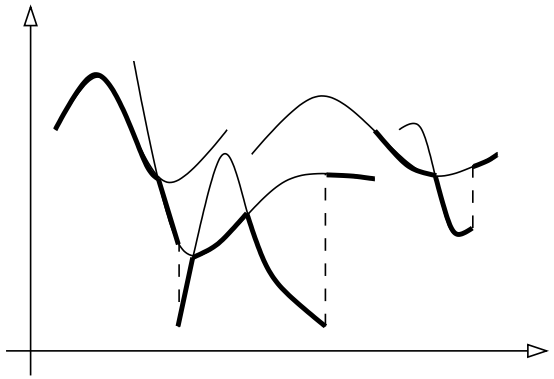
(2)

Funktionen über Teilintervalle

Aussage: Komplexität der Kontur von Funktionen die über gesamtem Intervall definiert sind 1) oder nur über ein Teilintervall 2).

1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

2. Verlängern auf gesamtes Intervall, dann 1. verwenden,



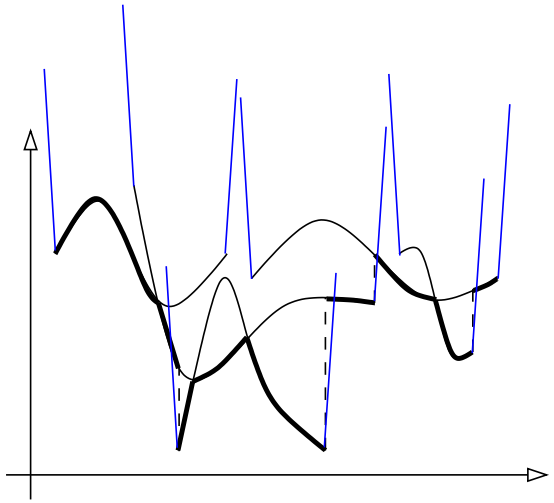
(2)

Funktionen über Teilintervalle

Aussage: Komplexität der Kontur von Funktionen die über gesamtem Intervall definiert sind 1) oder nur über ein Teilintervall 2).

1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

2. Verlängern auf gesamtes Intervall, dann 1. verwenden,



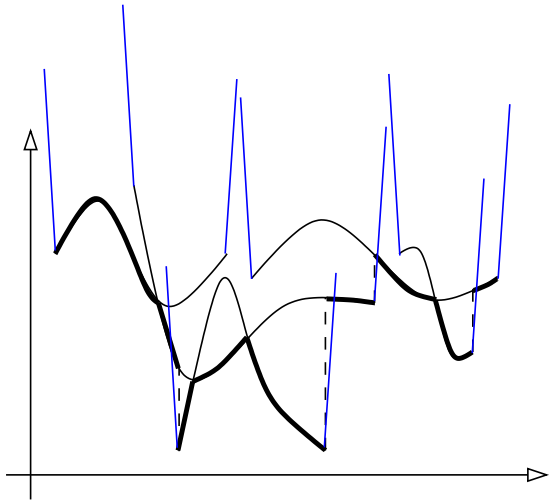
(2)

Funktionen über Teilintervalle

Aussage: Komplexität der Kontur von Funktionen die über gesamtem Intervall definiert sind 1) oder nur über ein Teilintervall 2).

1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

2. Verlängern auf gesamtes Intervall, dann 1. verwenden,



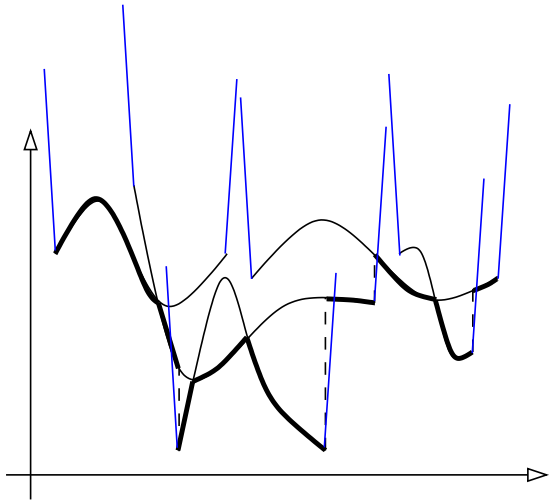
(2)

Funktionen über Teilintervalle

Aussage: Komplexität der Kontur von Funktionen die über gesamtem Intervall definiert sind 1) oder nur über ein Teilintervall 2).

1. $\lambda_s(n)$
2. $\lambda_{s+2}(n)$

2. Verlängern auf gesamtes Intervall, dann 1. verwenden, max. zwei zusätzliche Schnitte



(2)

Ergebnisse

Ergebnisse

Korollar 2.16 Die untere Kontur von n Liniensegmenten beliebiger Länge enthält $\lambda_3(n)$ viele Segmente und kann in Zeit $O(\lambda_3(n) \log n)$ berechnet werden.

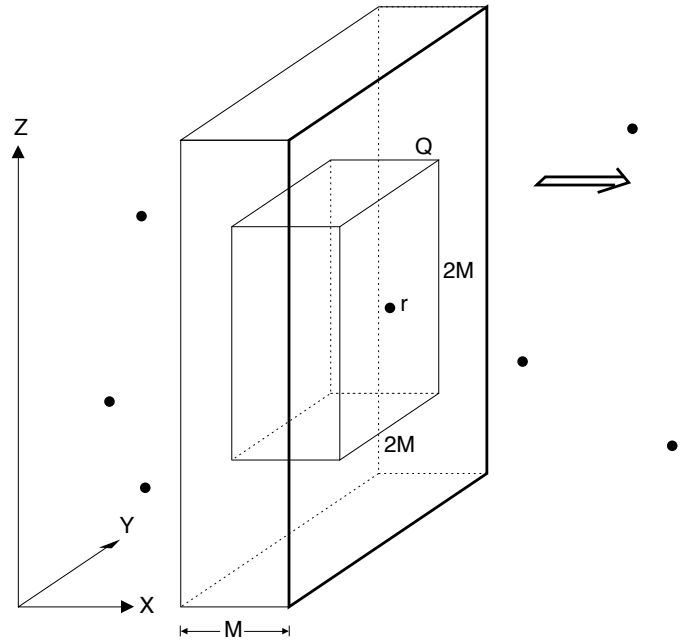
Weitere Sweepvarianten

Weitere Sweepvarianten

- Sweep mit Halbgerader, Sweep mit Circle
- Sweep-Ebene, Sweep im Raum

Weitere Sweepvarianten

- Sweep mit Halbgerader, Sweep mit Circle
- Sweep-Ebene, Sweep im Raum



Geometrische Datenstruktur

Geometrische Datenstruktur

- Datenobjekte Punkte im $\mathbb{R}^k$, andere Objekte übertragen

Geometrische Datenstruktur

- Datenobjekte Punkte im $\mathbb{R}^k$, andere Objekte übertragen
- Bislang: Eindimensionaler balancierter Suchbaum

Geometrische Datenstruktur

- Datenobjekte Punkte im $\mathbb{R}^k$, andere Objekte übertragen
- Bislang: Eindimensionaler balancierter Suchbaum
- Bereichsanfrage: Inklusionsanfrage, Schnittanfrage, Anfrageobjekt q

Geometrische Datenstruktur

- Datenobjekte Punkte im $\mathbb{R}^k$, andere Objekte übertragen
- Bislang: Eindimensionaler balancierter Suchbaum
- Bereichsanfrage: Inklusionsanfrage, Schnittanfrage, Anfrageobjekt q
- Interne, externe Datenstrukturen

Geometrische Datenstruktur

- Datenobjekte Punkte im $\mathbb{R}^k$, andere Objekte übertragen
- Bislang: Eindimensionaler balancierter Suchbaum
- Bereichsanfrage: Inklusionsanfrage, Schnittanfrage, Anfrageobjekt q
- Interne, externe Datenstrukturen
- Statische, dynamische Datenstrukturen

Geometrische Datenstruktur

- Datenobjekte Punkte im $\mathbb{R}^k$, andere Objekte übertragen
- Bislang: Eindimensionaler balancierter Suchbaum
- Bereichsanfrage: Inklusionsanfrage, Schnittanfrage, Anfrageobjekt q
- Interne, externe Datenstrukturen
- Statische, dynamische Datenstrukturen
- Rechteckige Bereichsanfrage im $\mathbb{R}^k$

Geometrische Datenstruktur

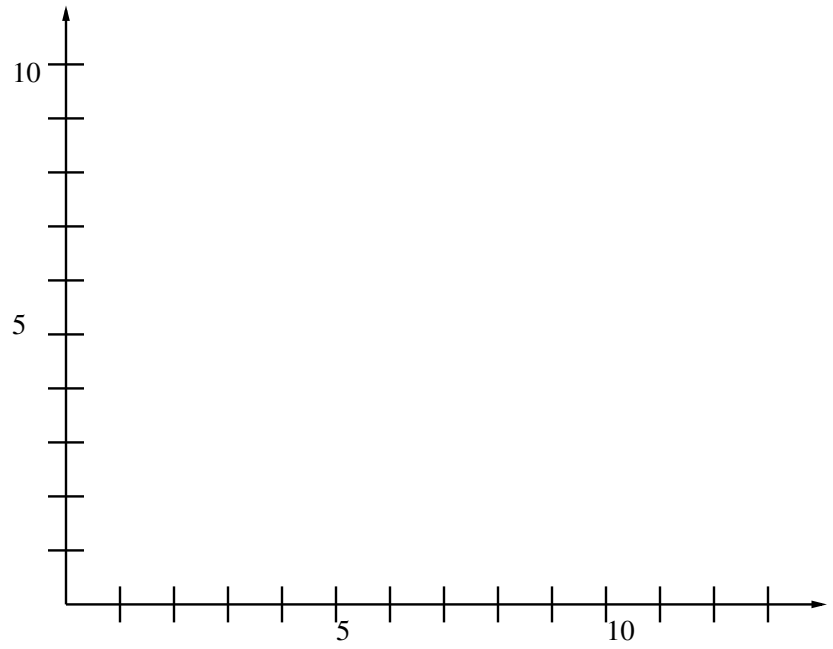
- Datenobjekte Punkte im $\mathbb{R}^k$, andere Objekte übertragen
- Bislang: Eindimensionaler balancierter Suchbaum
- Bereichsanfrage: Inklusionsanfrage, Schnittanfrage, Anfrageobjekt q
- Interne, externe Datenstrukturen
- Statische, dynamische Datenstrukturen
- Rechteckige Bereichsanfrage im $\mathbb{R}^k$
- k-dimensionaler Suchbaum

Geometrische Datenstruktur

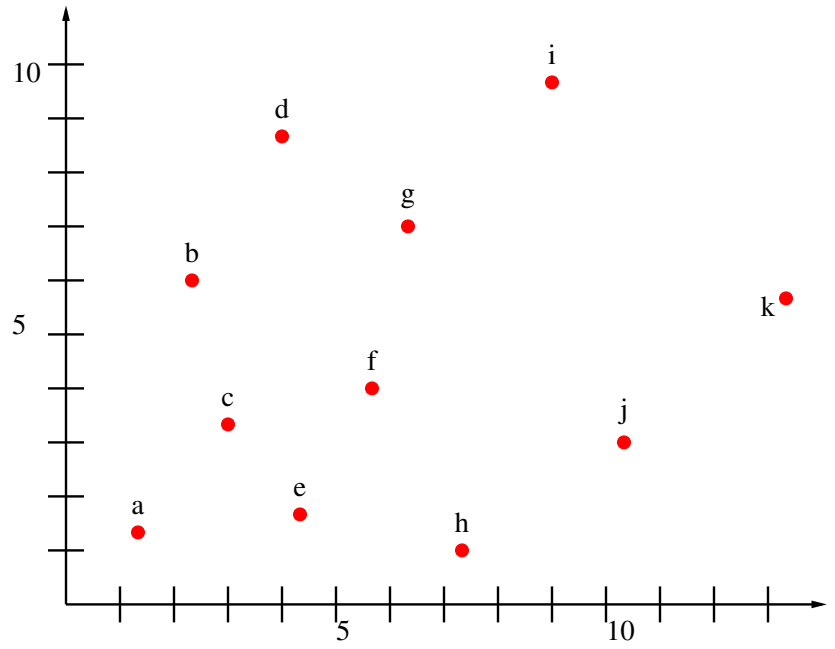
- Datenobjekte Punkte im $\mathbb{R}^k$, andere Objekte übertragen
- Bislang: Eindimensionaler balancierter Suchbaum
- Bereichsanfrage: Inklusionsanfrage, Schnittanfrage, Anfrageobjekt q
- Interne, externe Datenstrukturen
- Statische, dynamische Datenstrukturen
- Rechteckige Bereichsanfrage im $\mathbb{R}^k$
- k-dimensionaler Suchbaum
- Achsenparalleles Rechteck q, Punkte im $\mathbb{R}^2$

kd-Baum Aufbau

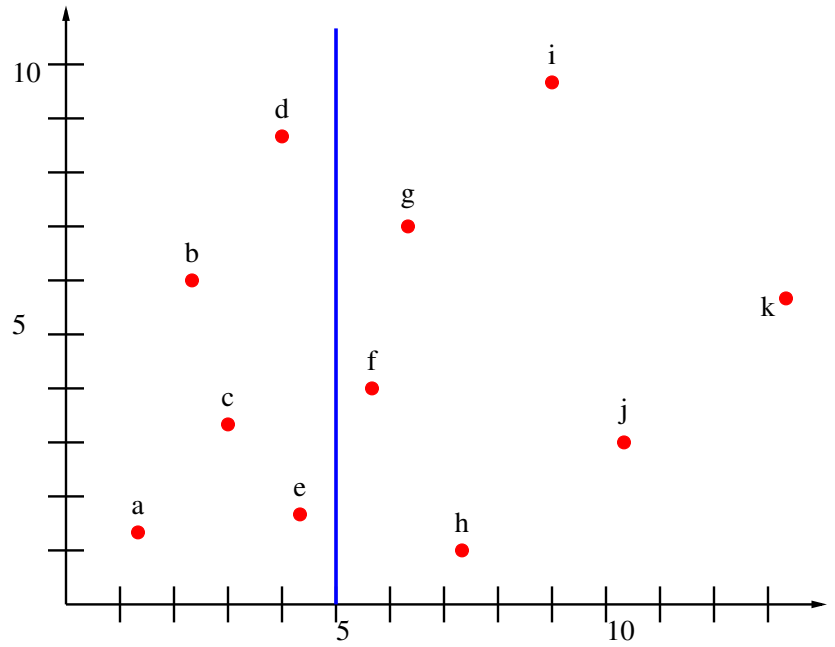
kd-Baum Aufbau

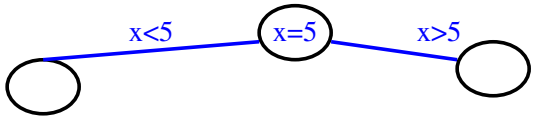


kd-Baum Aufbau

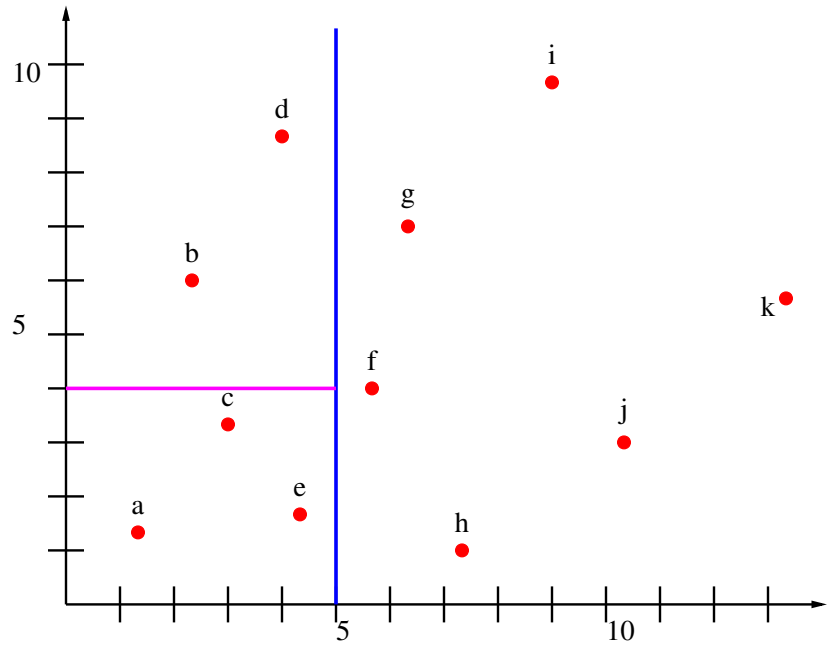


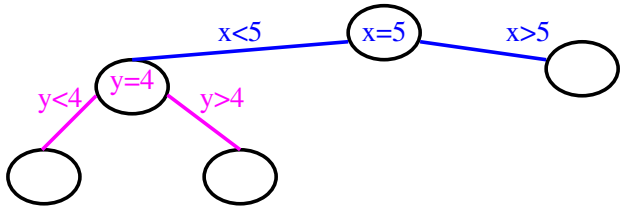
kd-Baum Aufbau



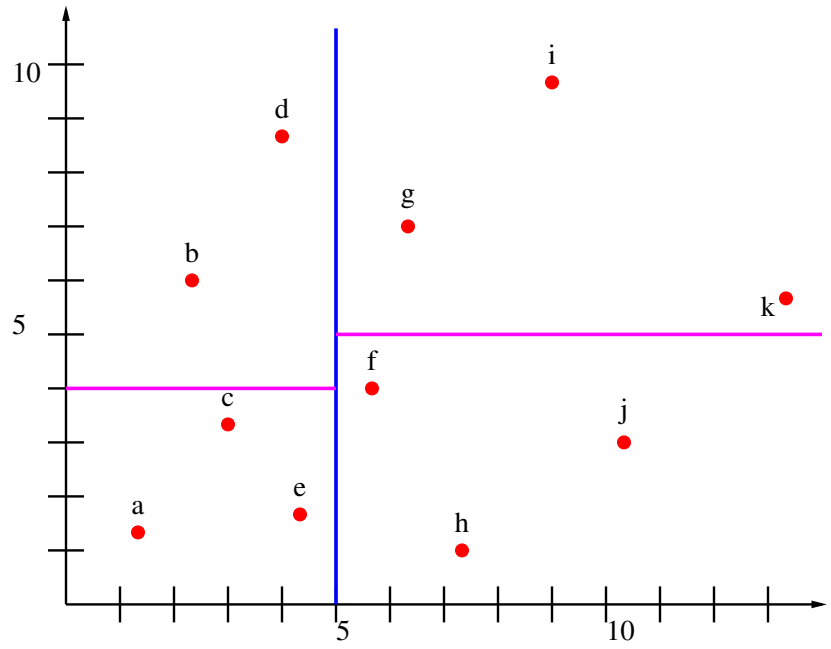


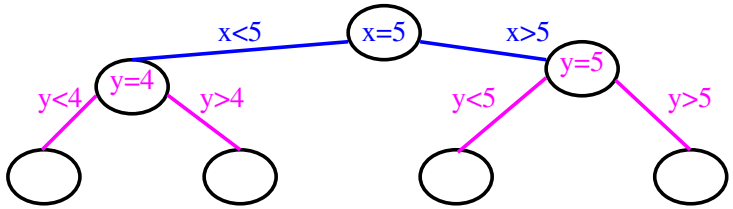
kd-Baum Aufbau



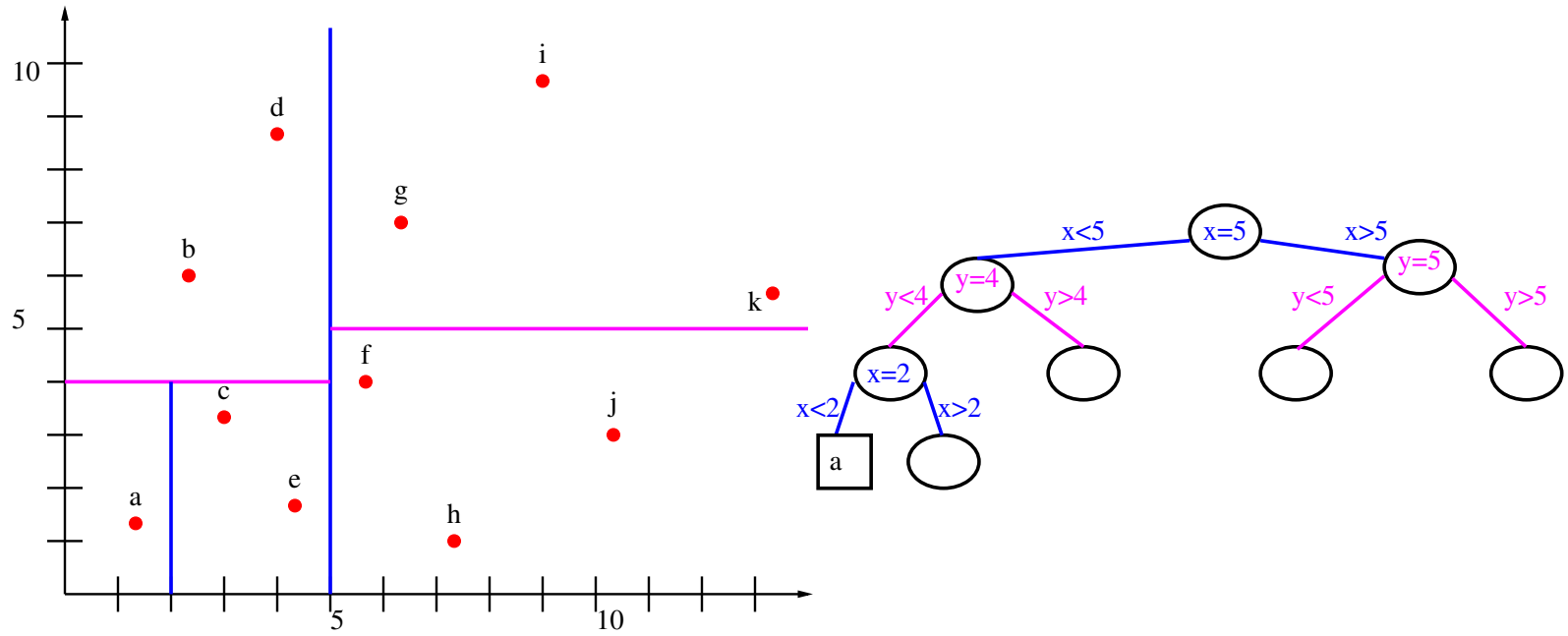


kd-Baum Aufbau

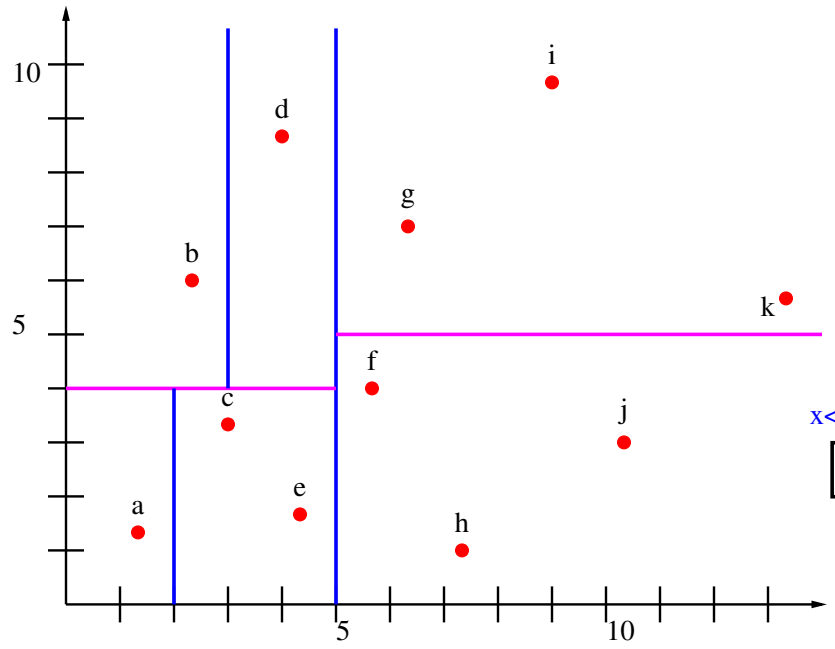


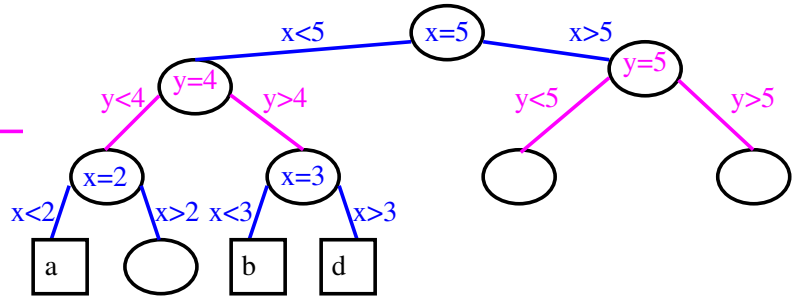


kd-Baum Aufbau

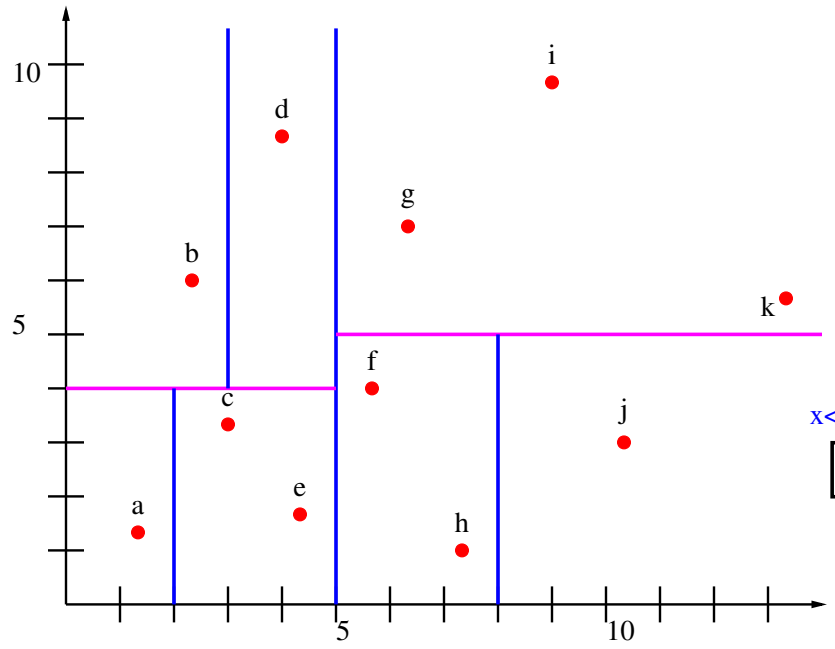


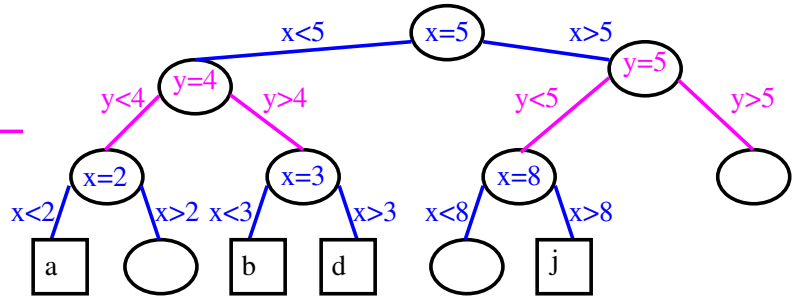
kd-Baum Aufbau



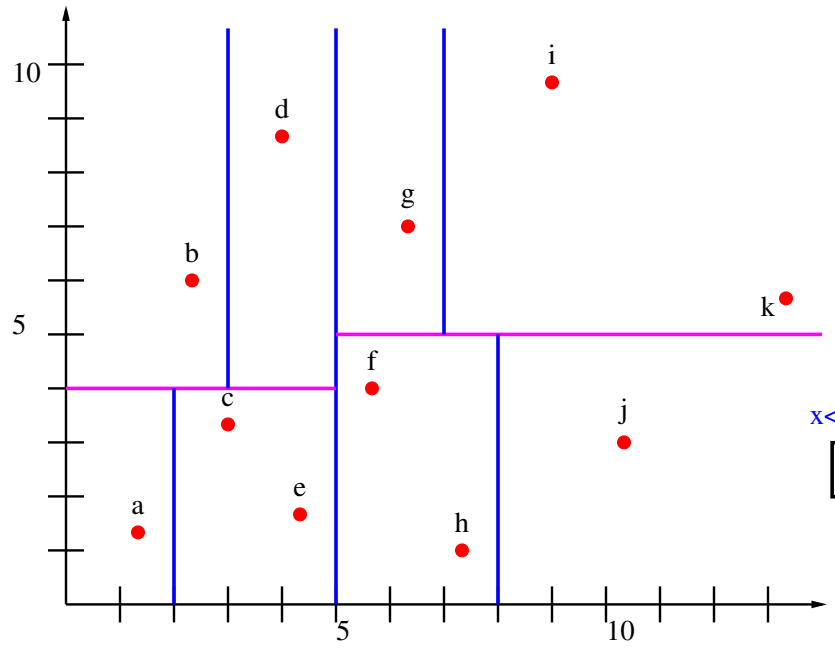


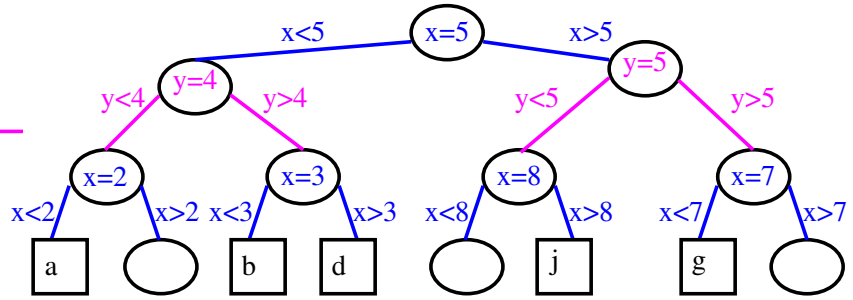
kd-Baum Aufbau



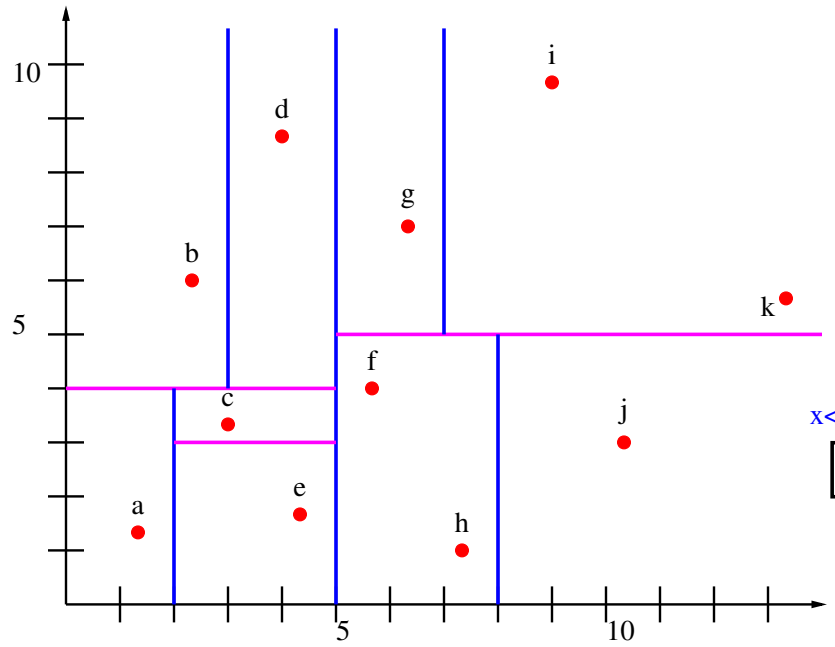


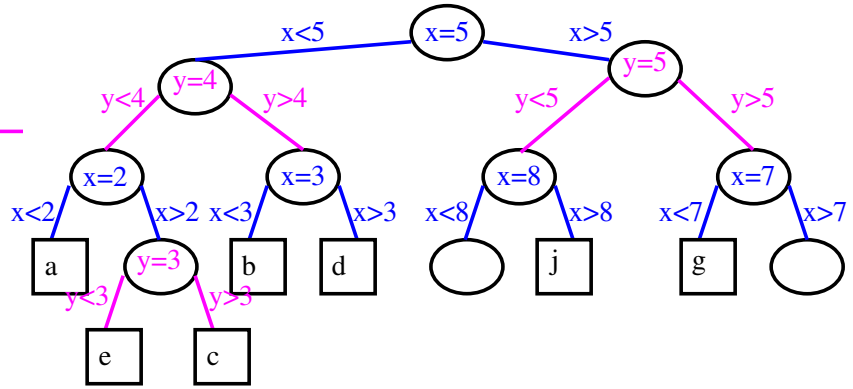
kd-Baum Aufbau



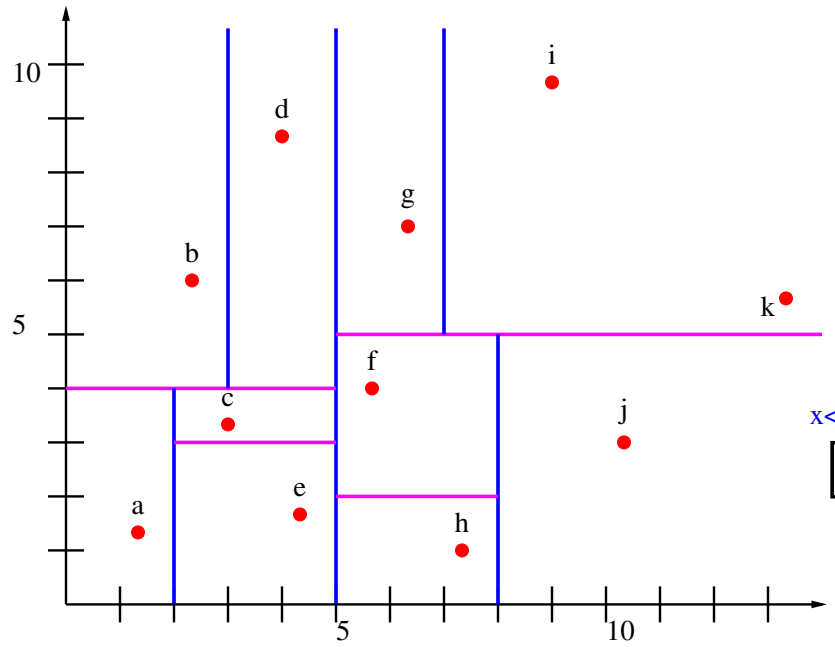


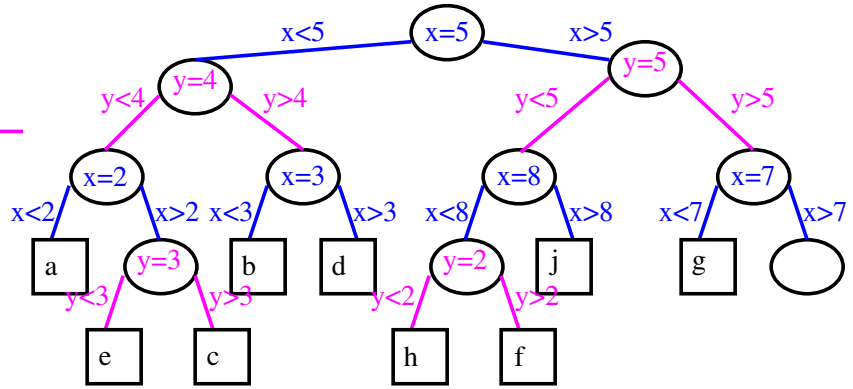
kd-Baum Aufbau



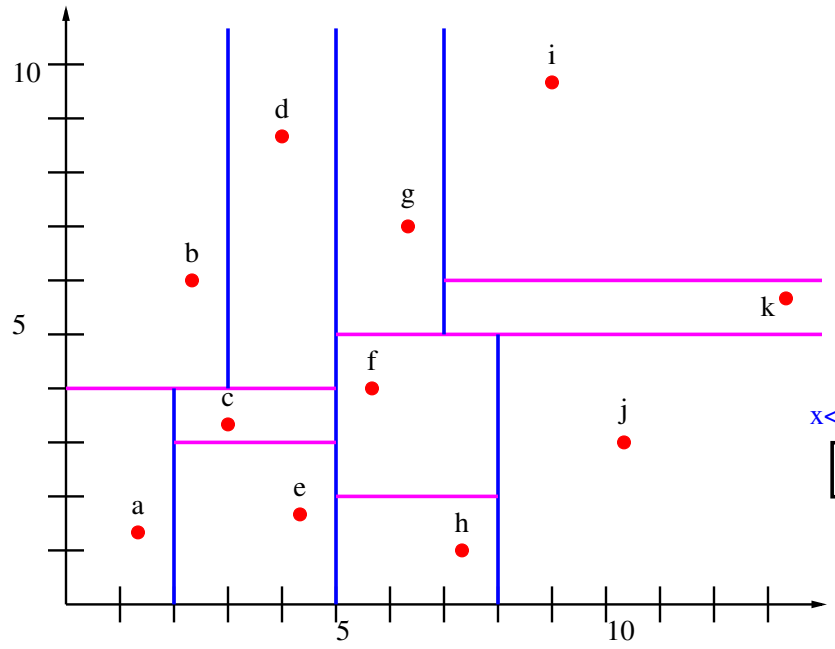


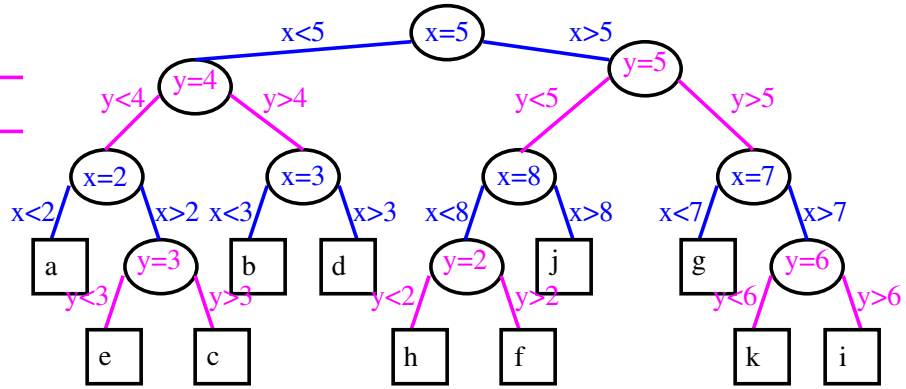
kd-Baum Aufbau





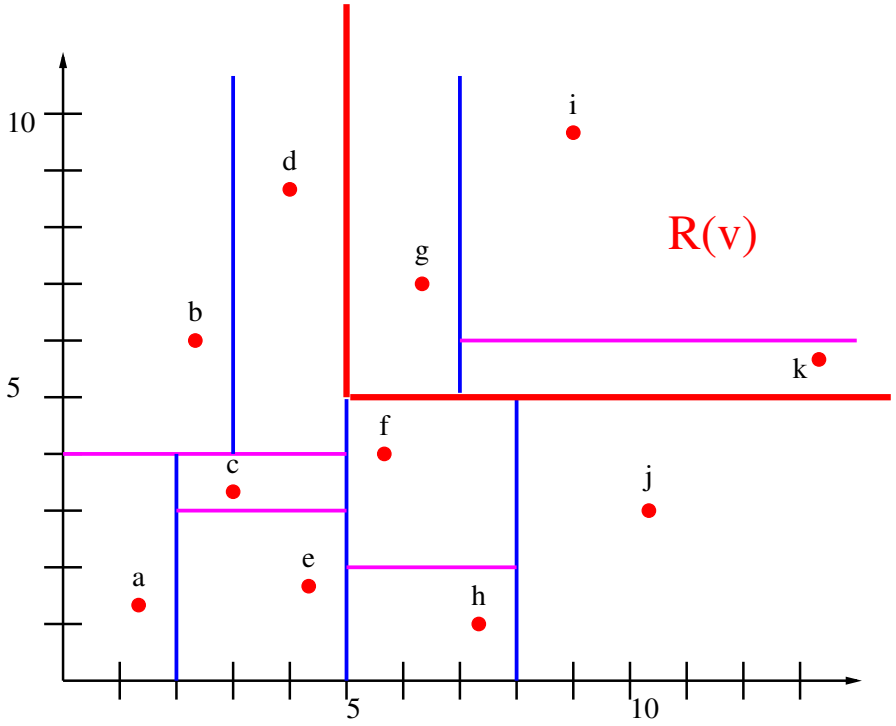
kd-Baum Aufbau

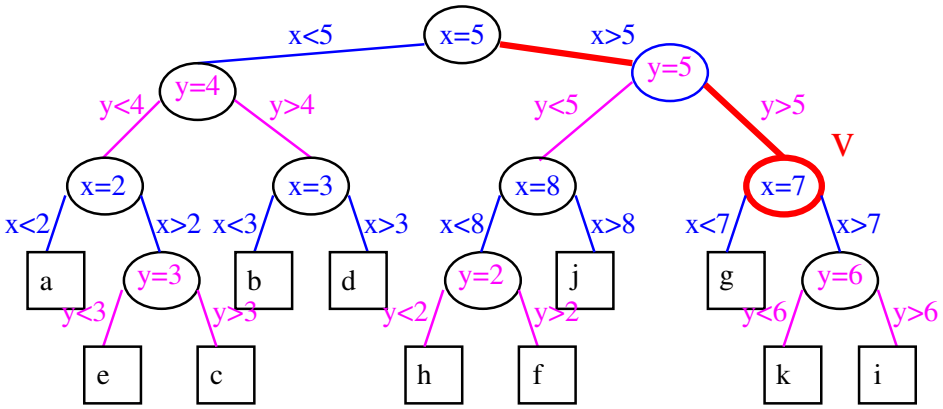




Knoten entspricht Rechteck $R(v)$

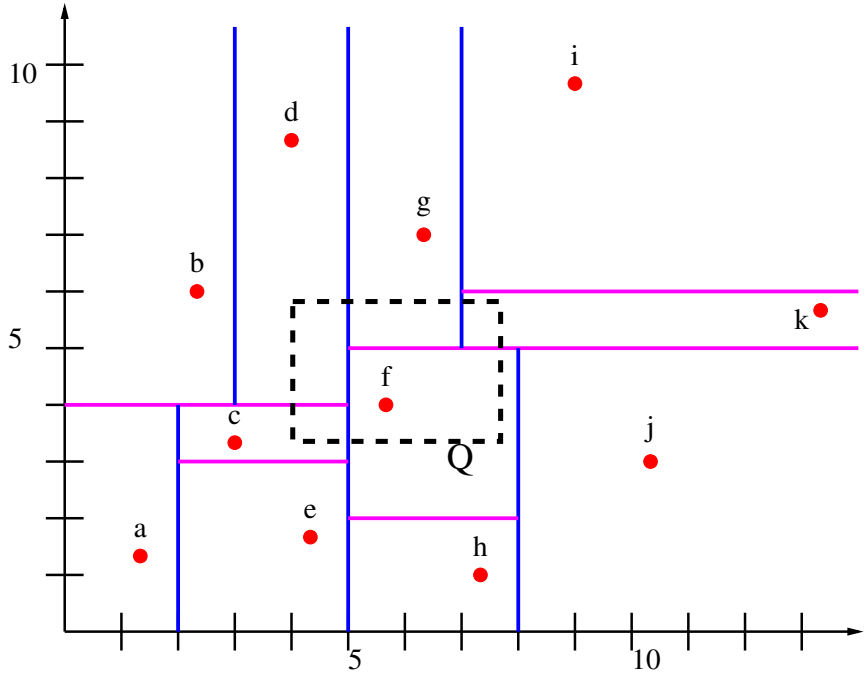
Knoten entspricht Rechteck $R(v)$

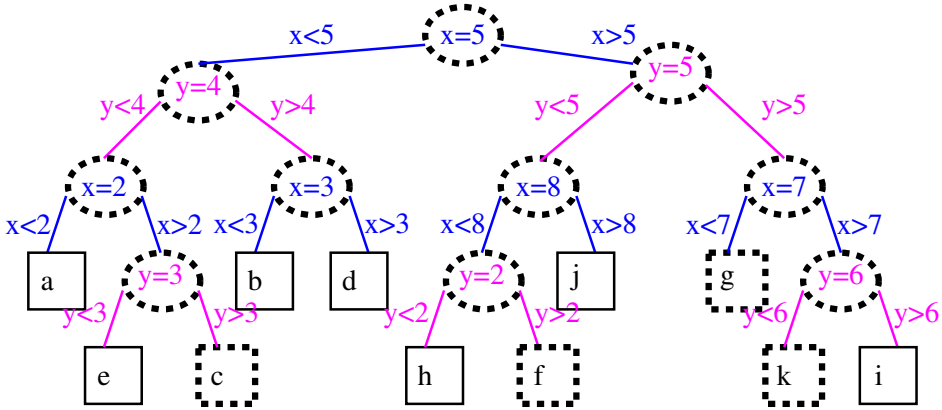




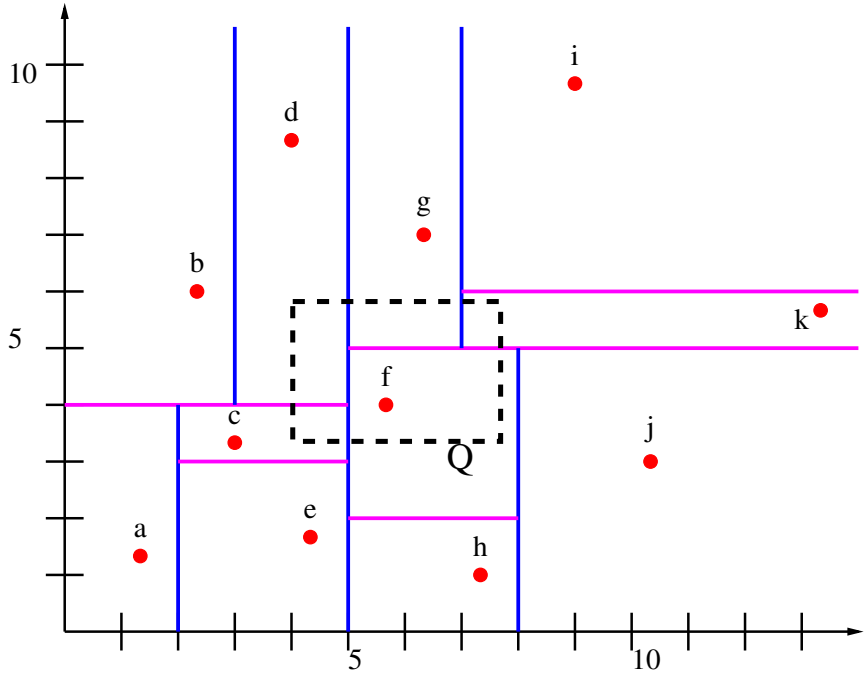
Query mit Rechteck q

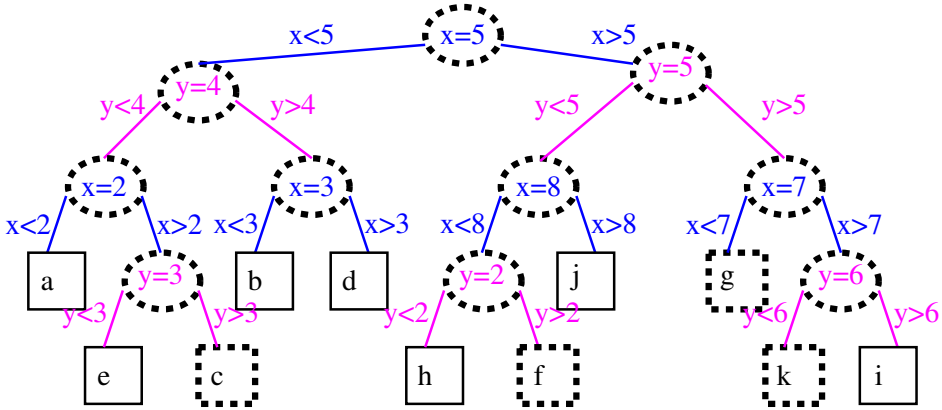
Query mit Rechteck q





Query mit Rechteck q





- Bestimme alle Knoten v mit $R(v) \cap q \neq \emptyset$
- Falls v Blatt, teste $v \in q$

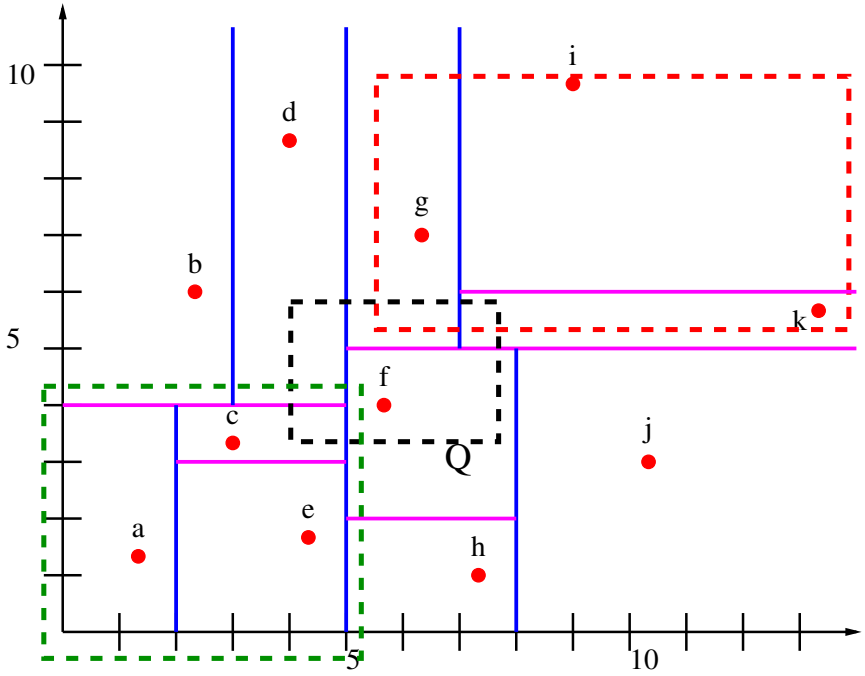
Query Laufzeit

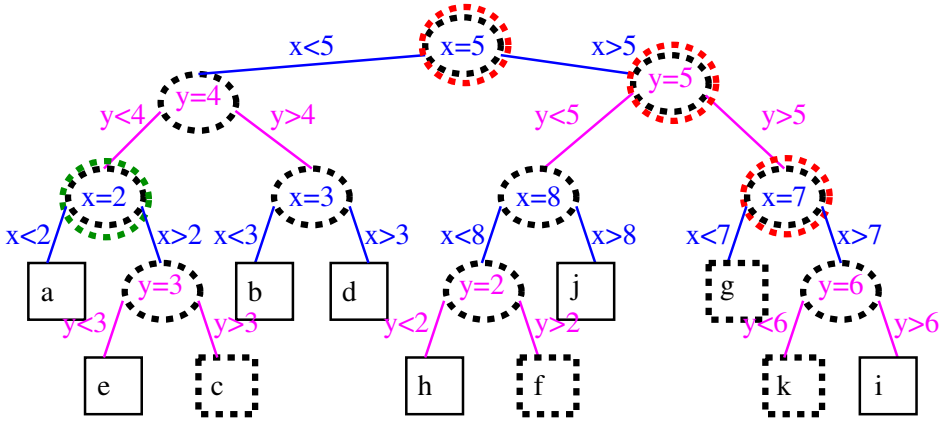
Query Laufzeit

Lemma 3.6: Sei T ein 2d-Baum der Höhe h mit n Punkten. Eine Bereichsanfrage mit achsenparallelen Rechteck Q läßt sich in Zeit $O\left(2^{\frac{h}{2}} + a\right)$ beantworten, wobei a die Größe der Antwort ist.

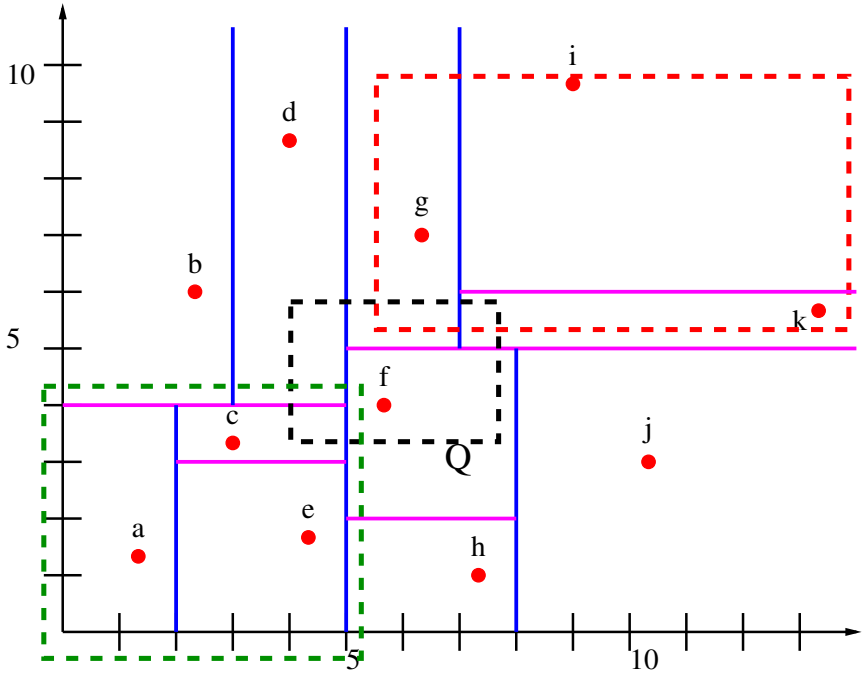
Beweis: Laufzeit $O\left(2^{\frac{h}{2}} + a\right)$

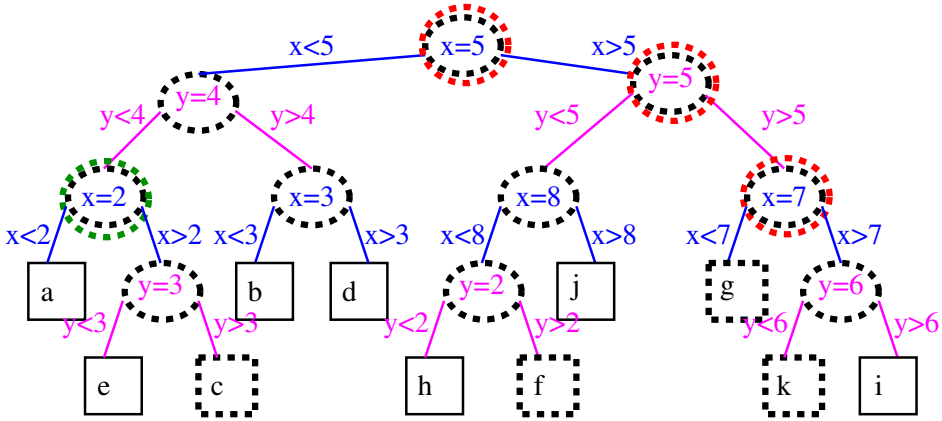
Beweis: Laufzeit $O\left(2^{\frac{h}{2}} + a\right)$





Beweis: Laufzeit $O\left(2^{\frac{h}{2}} + a\right)$





1. Knoten v mit $R(v) \subseteq q$
2. Knoten v mit $R(v) \not\subseteq q$ (aber $R(v) \cap q \neq \emptyset$), unterteilt sich in zwei Subtypen

Ergebnis

Ergebnis

Theorem 3.7: Ein ausgeglichener 2d-Baum für n Punkte in der Ebene läßt sich in Zeit $O(n \log n)$ konstruieren. Er benötigt $O(n)$ Speicherplatz . Eine Bereichsanfrage mit achsenparallelen Rechteck q kann in Zeit $O\left(\sqrt{n} + a\right)$ beantwortet werden, wobei a die Größe der Antwort ist.

Ergebnis

Theorem 3.7: Ein ausgeglichener 2d-Baum für n Punkte in der Ebene läßt sich in Zeit $O(n \log n)$ konstruieren. Er benötigt $O(n)$ Speicherplatz . Eine Bereichsanfrage mit achsenparallelen Rechteck q kann in Zeit $O\left(\sqrt{n} + a\right)$ beantwortet werden, wobei a die Größe der Antwort ist.

Sortieren nach X und Y und rekursiv in gleichgroße Teilmengen aufteilen, lineares Aufteilen