Untere Schranken/Untere Konturen

Elmar Langetepe
University of Bonn

Epsilon-Closeness

Korollar 1.6 Das Problem ϵ-Closeness hat im linearen Modell eine Zeitkomplexität von $\Theta(n \log n)$.

- $x_1, x_2, \ldots, x_n$, $\epsilon > 0$, ex. (i, j) mit $|x_i - x_j| < \epsilon$
- $O(n \log n)$ durch Sortieren!
- $\Omega(n \log n)$ verwende Elementtest!
- $W = \{(x_1, x_2, \ldots, x_n) || x_i - x_j| \geq \epsilon$ für alle $i \neq j\}$
- ϵ-Closeness ist Elementtest für W
- $W_\pi = \{(x_1, x_2, \ldots, x_n) \in W | x_{\pi(1)} < x_{\pi(2)} < \cdots < x_{\pi(n)}\}$ sind Zs.-hangskomponenten von W
- $n!$ Permutationen π

Epsilon-Closeness

Epsilon-Closeness

Analog Element-Uniqueness!
Problem: $x_1, x_2, \ldots, x_n \in \mathbb{R}$, ex. (i, j) mit $|x_i - x_j| = 0$?

Korollar 1.7 Das Problem Element-Uniqueness hat im linearen Modell eine Zeitkomplexität von $\Theta(n \log n)$.

Weitere Folgerungen!

Weitere Folgerungen!

Korollar 1.8 Sortieren hat auch im linearen Modell eine Zeitkomplexität von $\Theta(n \log n)$.

ϵ-Closeness/Element Uniqeness $\leq_P$ Sortieren, $P \in O(n)$!

Weitere Folgerungen!

Korollar 1.8 Sortieren hat auch im linearen Modell eine Zeitkomplexität von $\Theta(n \log n)$.

ϵ-Closeness/Element Uniqeness $\leq_P$ Sortieren, $P \in O(n)$!

Korollar 1.9 Für jeden Punkt einer n elementigen Menge $S \subseteq \mathbb{R}^d$ seinen nächsten Nachbarn zu finden, hat Zeitkomplexität $\Omega(n \log n)$.

ϵ-Closeness $\leq_P$ All-Nearest-Neighbors, $P \in O(n)$!

Weitere Folgerungen!

Korollar 1.8 Sortieren hat auch im linearen Modell eine Zeitkomplexität von $\Theta(n \log n)$.

ϵ-Closeness/Element Uniqeness $\leq_P$ Sortieren, $P \in O(n)$!

Korollar 1.9 Für jeden Punkt einer n elementigen Menge $S \subseteq \mathbb{R}^d$ seinen nächsten Nachbarn zu finden, hat Zeitkomplexität $\Omega(n \log n)$.

ϵ-Closeness $\leq_P$ All-Nearest-Neighbors, $P \in O(n)$!

Korollar 1.10 Das dichteste Punktepaar (closest-pair) einer n elementigen Menge $S \subseteq \mathbb{R}^d$ zu finden, hat Zeitkomplexität $\Omega(n \log n)$.

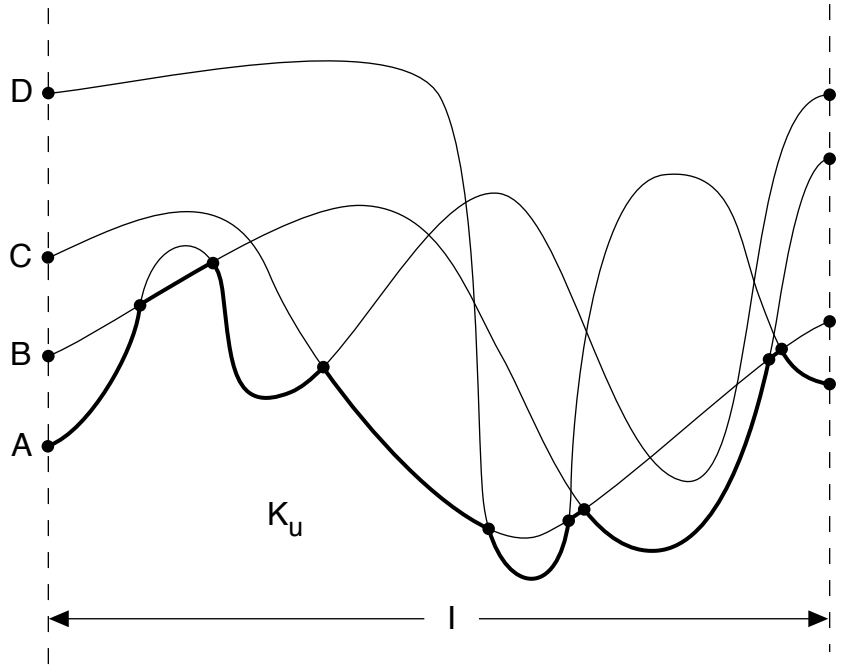
Untere Schranke Schnitt Liniensegmente

Lemma 2.6 Das Existenzproblem für den Schnitt von n Liniensegmenten hat Zeitkomplexität $\Omega(n \log n)$. Das Aufzählungsproblem für k Schnittpunkte hat Zeitkomplexität $\Omega(n \log n + k)$.

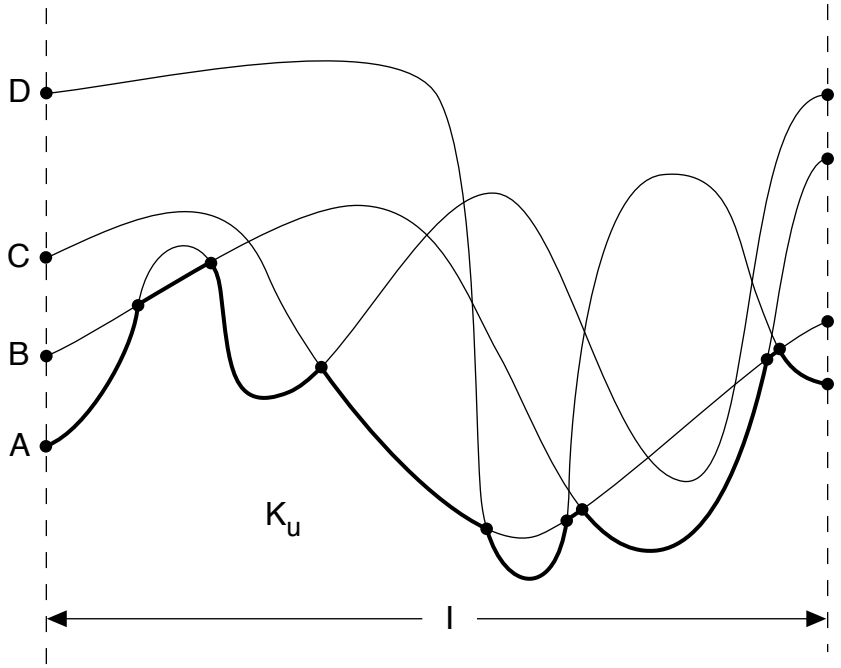
ϵ-Closeness $\leq_P$ Existenz Schnitt von Liniensegmenten, $P \in O(n)$!

Häufiges Hilfsmittel: Konturen

Häufiges Hilfsmittel: Konturen

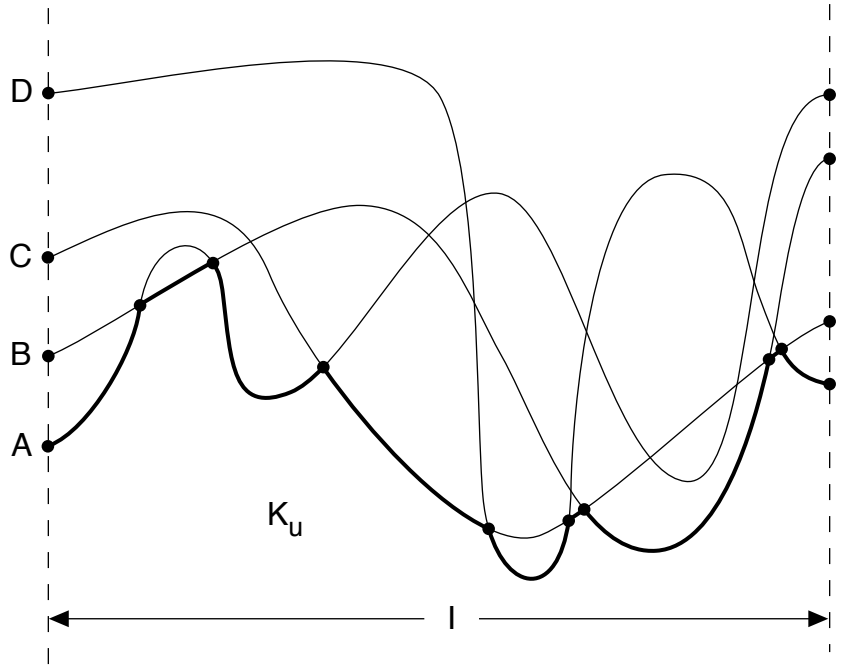


Häufiges Hilfsmittel: Konturen



- Sweep, X-monotone Wege, Schnittpunkte berechnen

Häufiges Hilfsmittel: Konturen



- Sweep, X-monotone Wege, Schnittpunkte berechnen
- Funktionsauswertung: Nächster Schnittpunkt $O(1)$

Erweiterung Theorem

Erweiterung Theorem

Theorem 2.11 Mit dem Sweep-Verfahren aus Abschnitt 2.3.2 lassen sich die k Schnittpunkte von n verschiedenen X-monotonen Wegen in Zeit $O((n+k)\log n)$ berechnen.

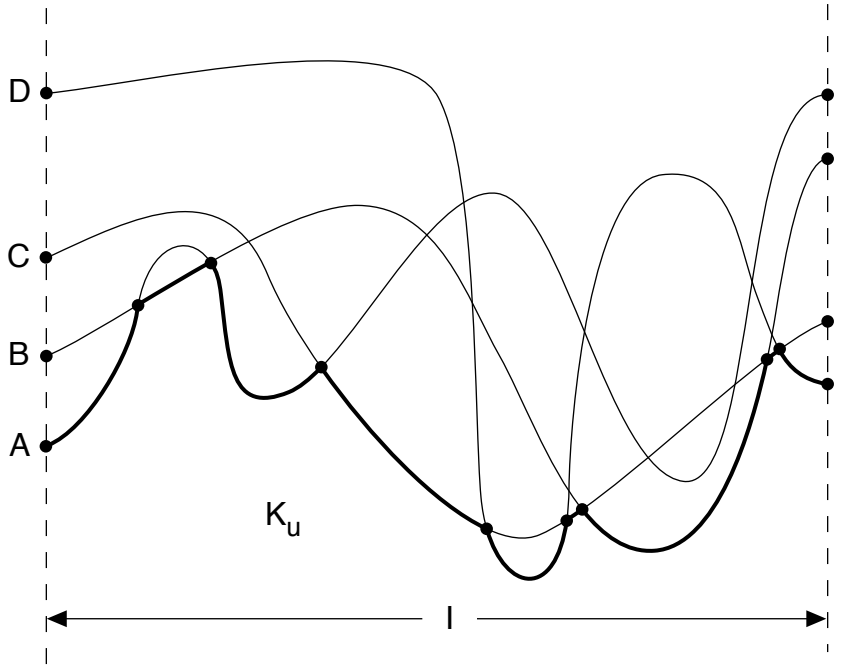
Erweiterung Theorem

Theorem 2.11 Mit dem Sweep-Verfahren aus Abschnitt 2.3.2 lassen sich die k Schnittpunkte von n verschiedenen X-monotonen Wegen in Zeit $O((n+k)\log n)$ berechnen.

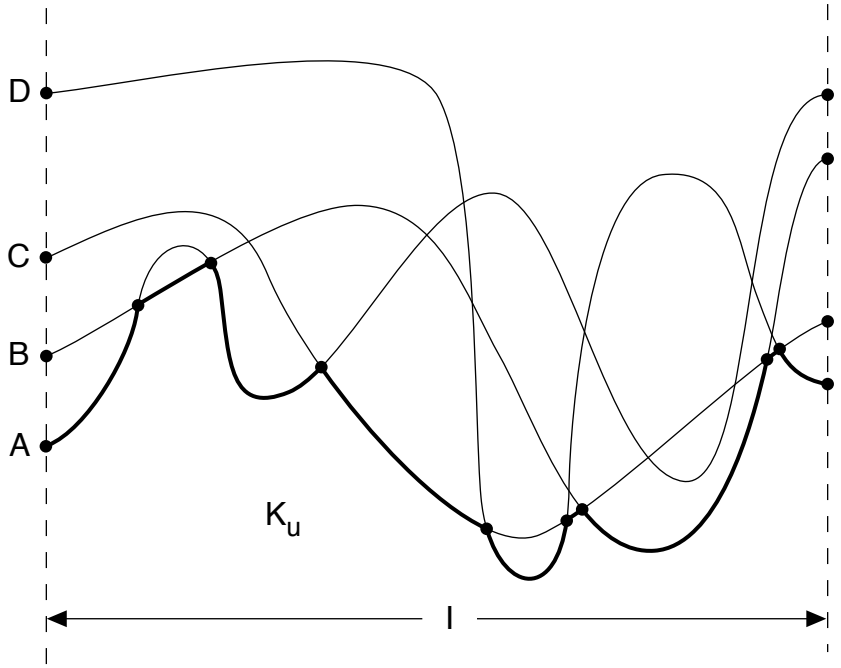
Untere Kontur genauso! Untere *Segmente* speichern!
Effizient?

Konturen

Konturen

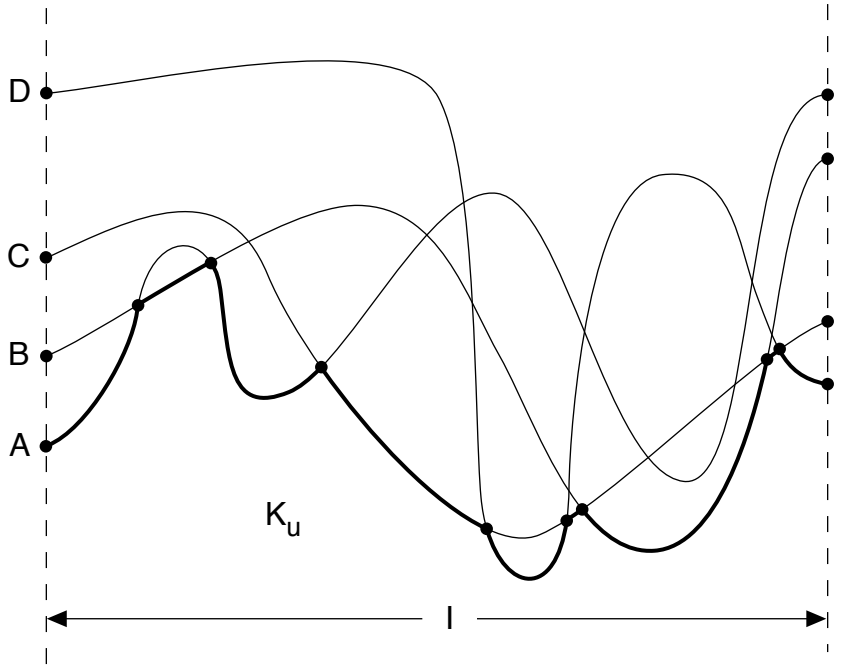


Konturen



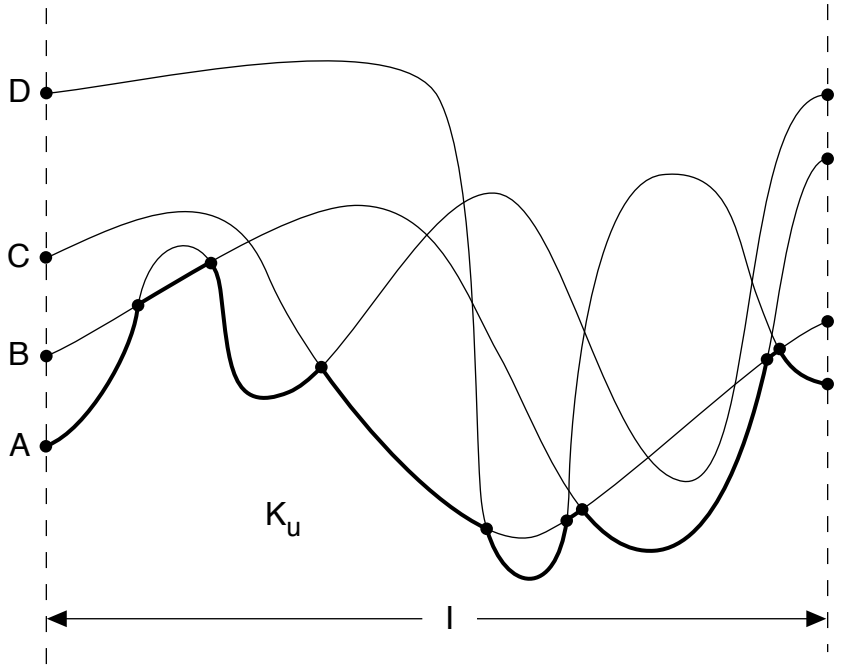
Polynome vom Grad 3!

Konturen



Polynome vom Grad 3! Deutlich geringere Komplexität!

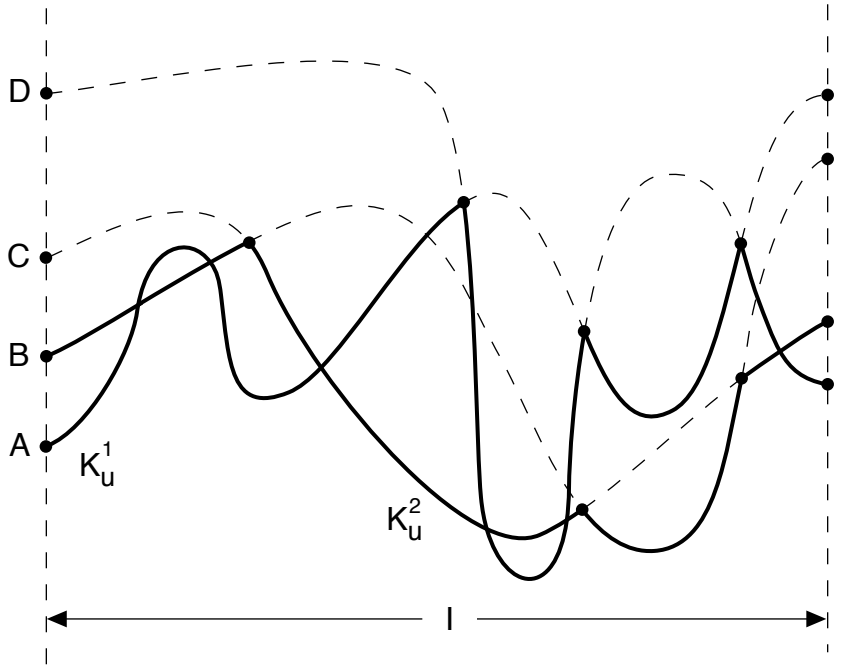
Konturen



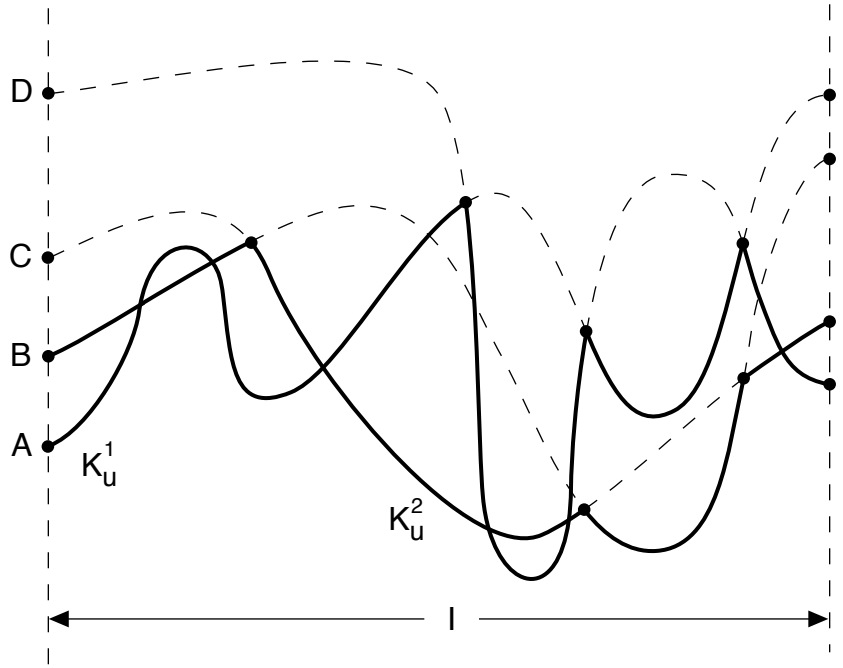
Polynome vom Grad 3! Deutlich geringere Komplexität!
Output-sensitiv!

Divide-and-Conquer, Sweep im Merge

Divide-and-Conquer, Sweep im Merge



Divide-and-Conquer, Sweep im Merge



n X-monotone Wege, zwei schneiden sich s mal!
Definition $\lambda_s(n)$: Maximale Komplexität der Kontur bei obigen Bedingungen.

Berechnung Kontur!

Berechnung Kontur!

Theorem 2.13 Die untere Kontur von n verschiedenen X monotonen Wegen über einem gemeinsamen Intervall, von denen sich je zwei höchstens s-mal schneiden, kann in Zeit $O(\lambda_s(n) \log n)$ berechnet werden.

Definition $\lambda_s(n)$: Maximale Komplexität der Kontur bei obigen Bedingungen.

Berechnung Kontur!

Theorem 2.13 Die untere Kontur von n verschiedenen X monotonen Wegen über einem gemeinsamen Intervall, von denen sich je zwei höchstens s-mal schneiden, kann in Zeit $O(\lambda_s(n) \log n)$ berechnet werden.

Definition $\lambda_s(n)$: Maximale Komplexität der Kontur bei obigen Bedingungen. Später Abschätzen!

Berechnung Kontur!

Theorem 2.13 Die untere Kontur von n verschiedenen X monotonen Wegen über einem gemeinsamen Intervall, von denen sich je zwei höchstens s-mal schneiden, kann in Zeit $O(\lambda_s(n) \log n)$ berechnet werden.

Definition $\lambda_s(n)$: Maximale Komplexität der Kontur bei obigen Bedingungen. Später Abschätzen!

Beweis! Divide and Conquer mit Sweep im Merge in $\lambda_s(n)$!

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

$\lambda_s(n)$, 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

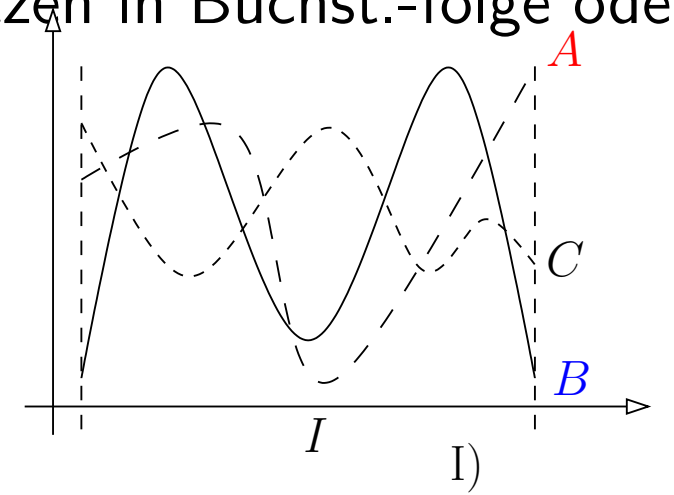
Theorem 2.14 Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$.

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 $\lambda_s(n)$.

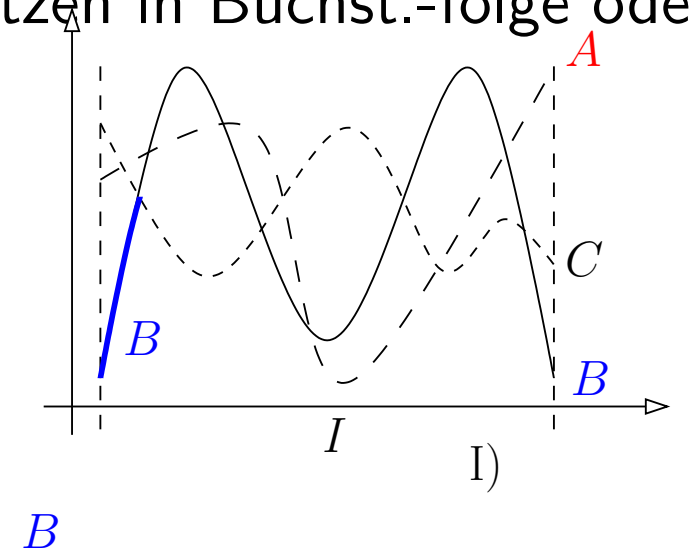
I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



Beweis: Untere Kontur/DSS

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

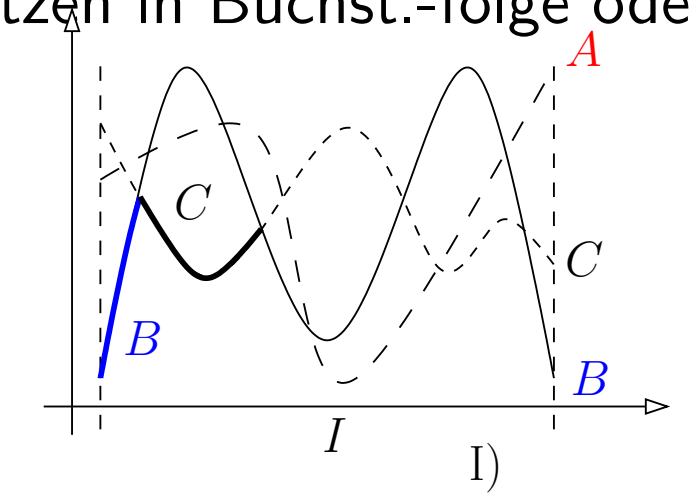
I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



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

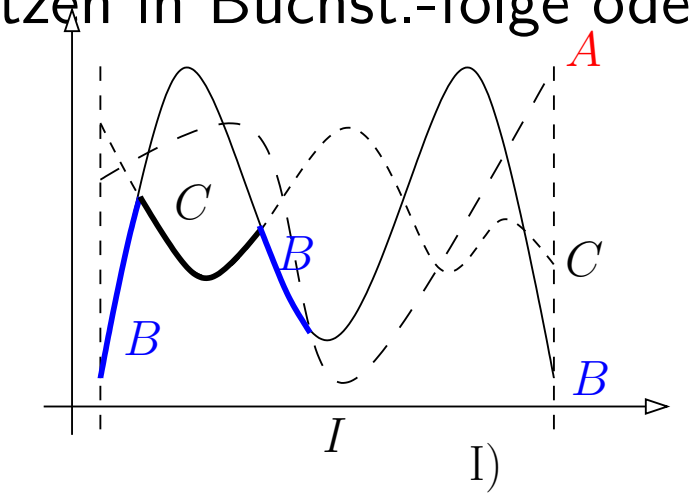


B C

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

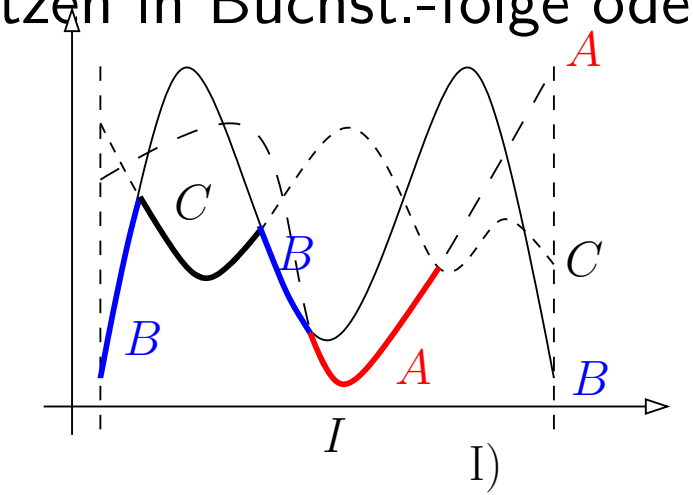


$B\ C\ B$

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

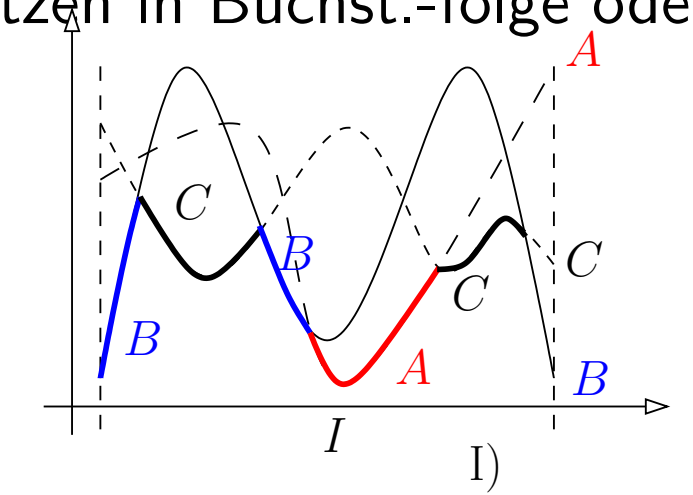


$B\ C\ B\ A$

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

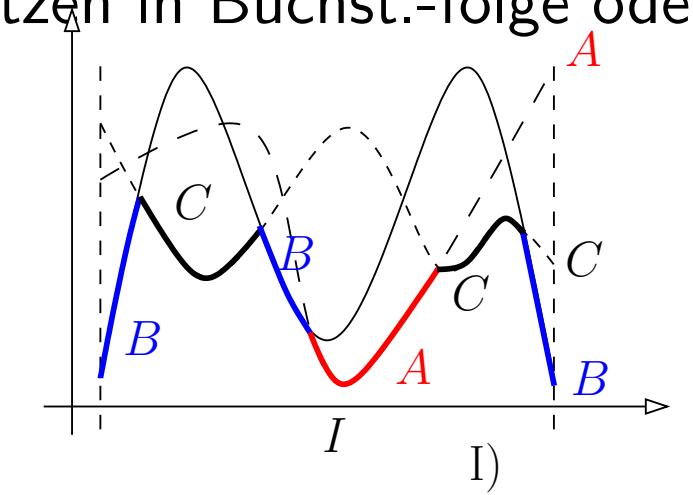


$B\ C\ B\ A\ C$

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

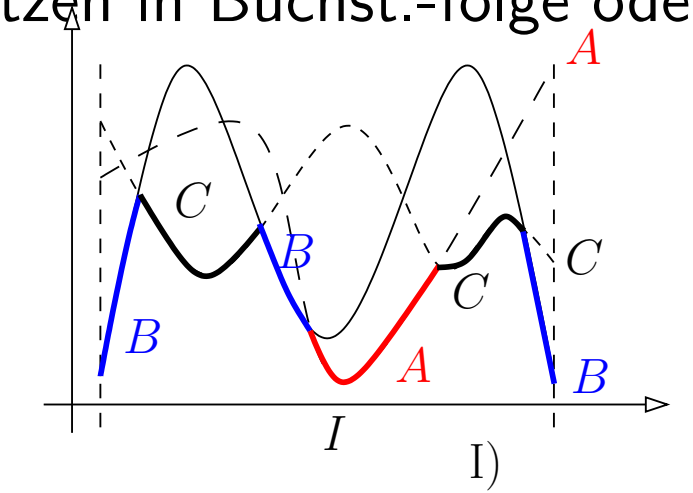


$B\ C\ B\ A\ C\ B$

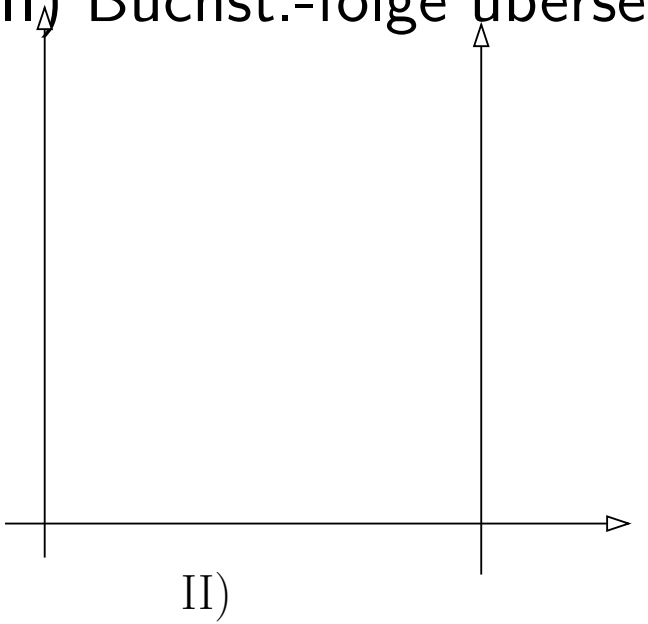
Beweis: Untere Kontur/DSS

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



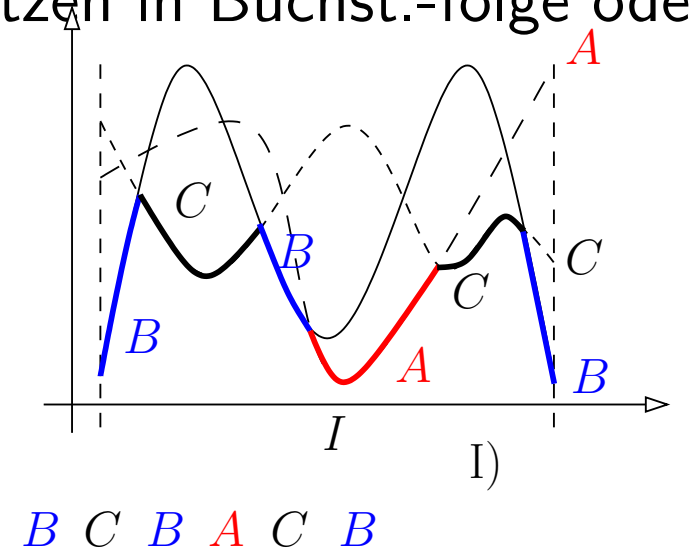
$B\ C\ B\ A\ C\ B$

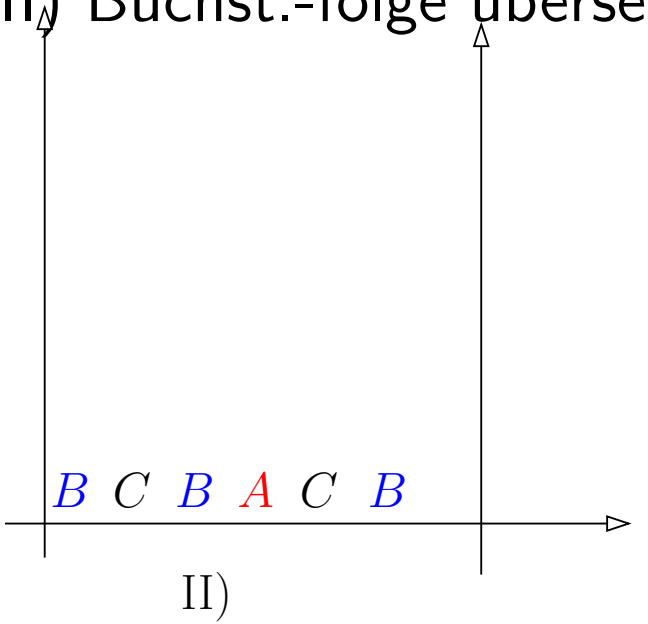


Beweis: Untere Kontur/DSS

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

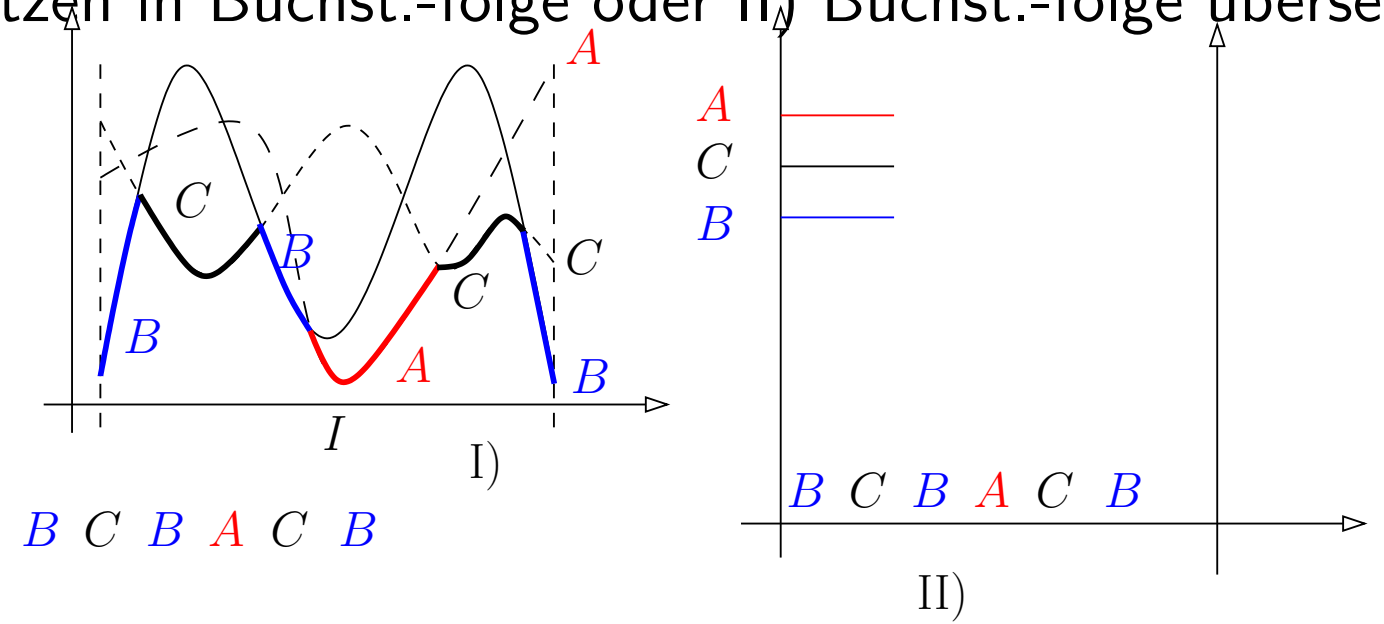




Beweis: Untere Kontur/DSS

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

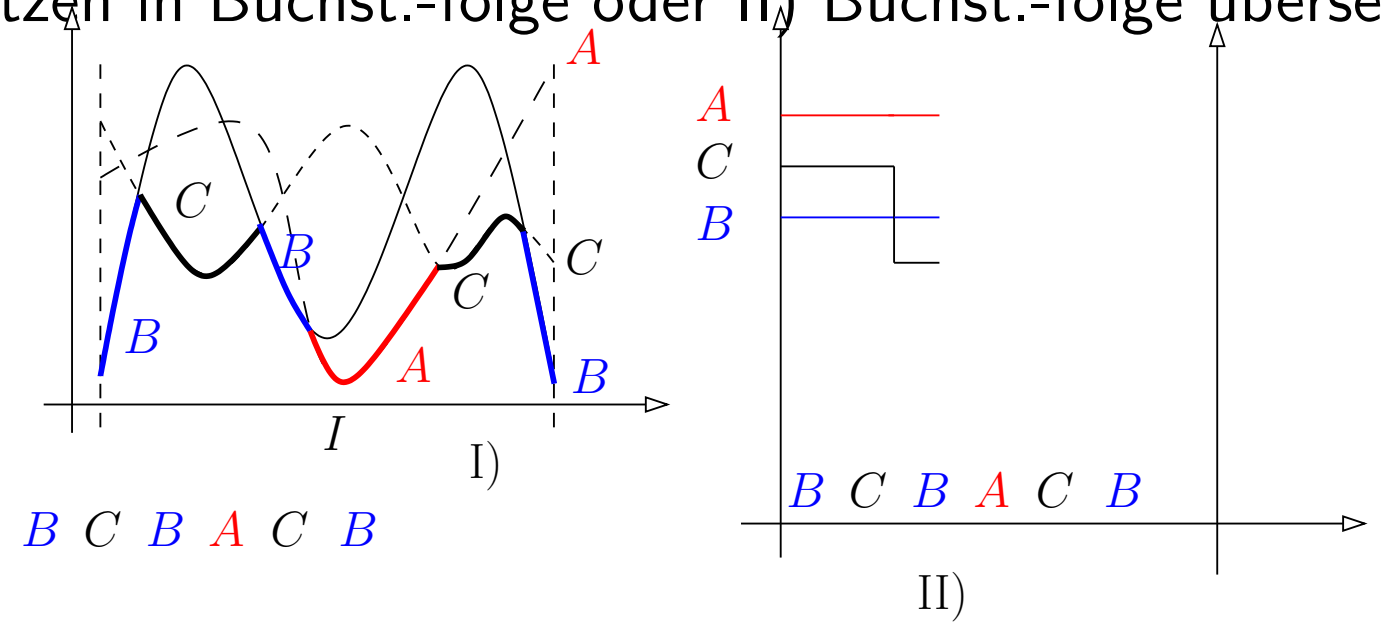
I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



Beweis: Untere Kontur/DSS

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

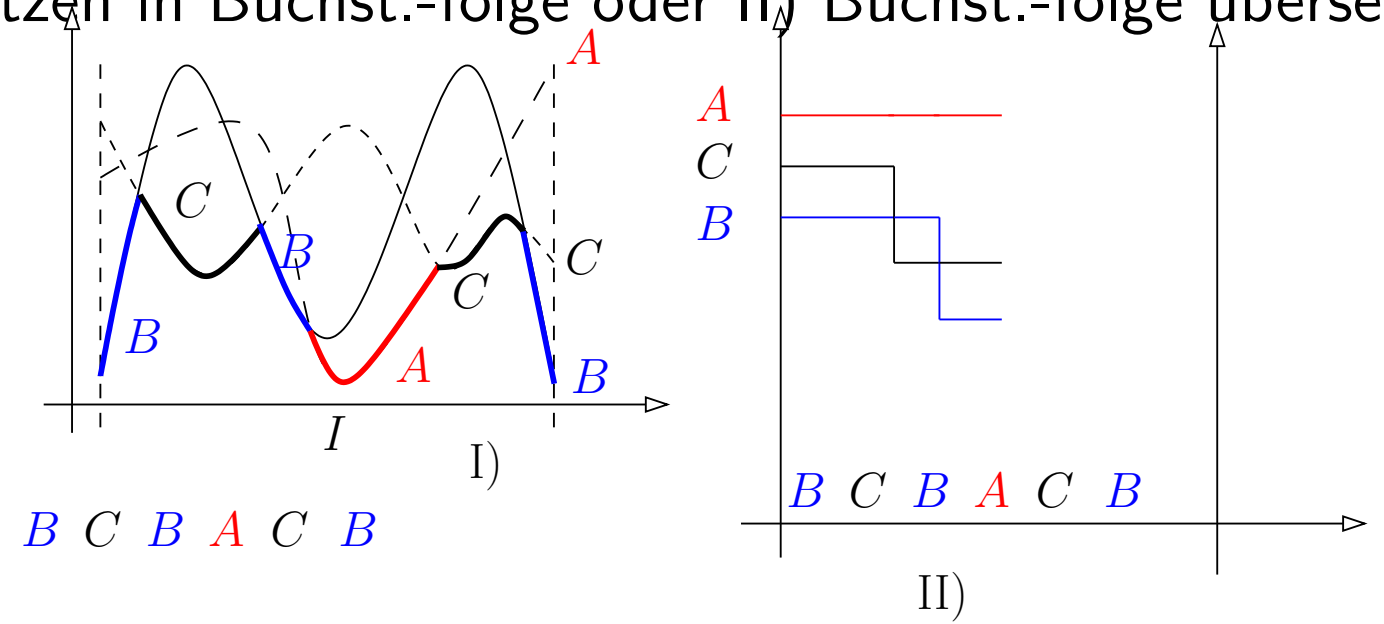
I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



Beweis: Untere Kontur/DSS

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

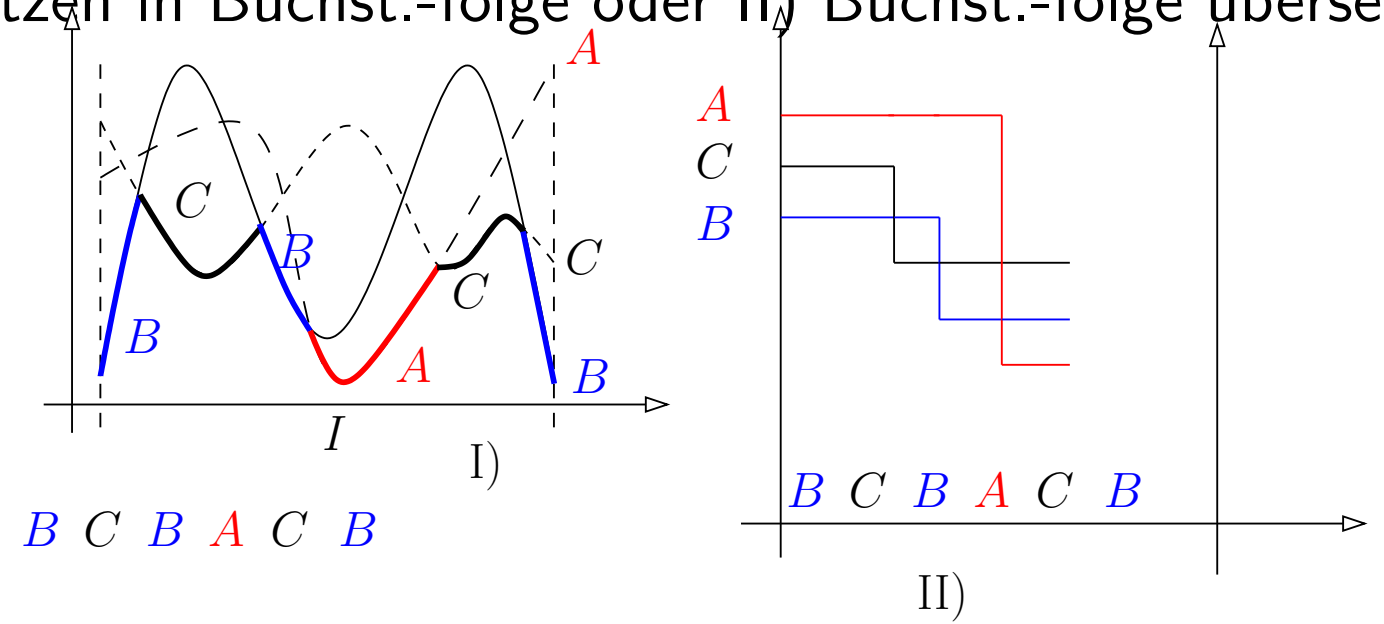
I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



Beweis: Untere Kontur/DSS

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

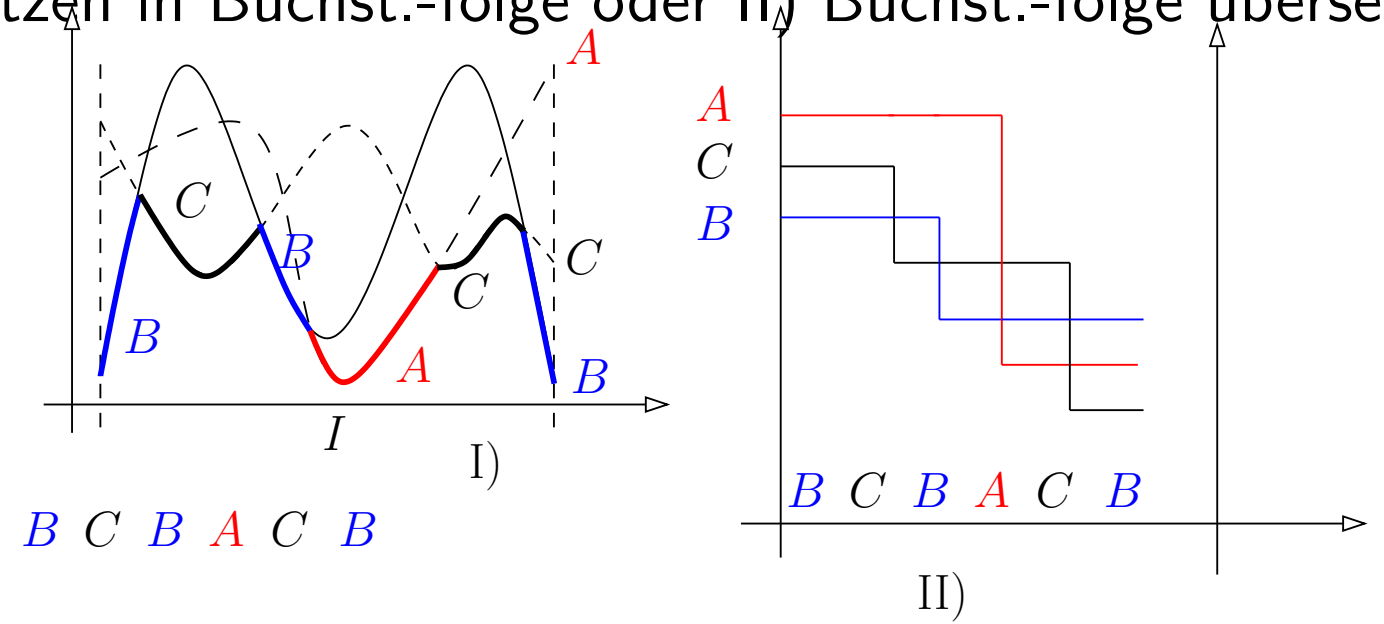
I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



Beweis: Untere Kontur/DSS

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

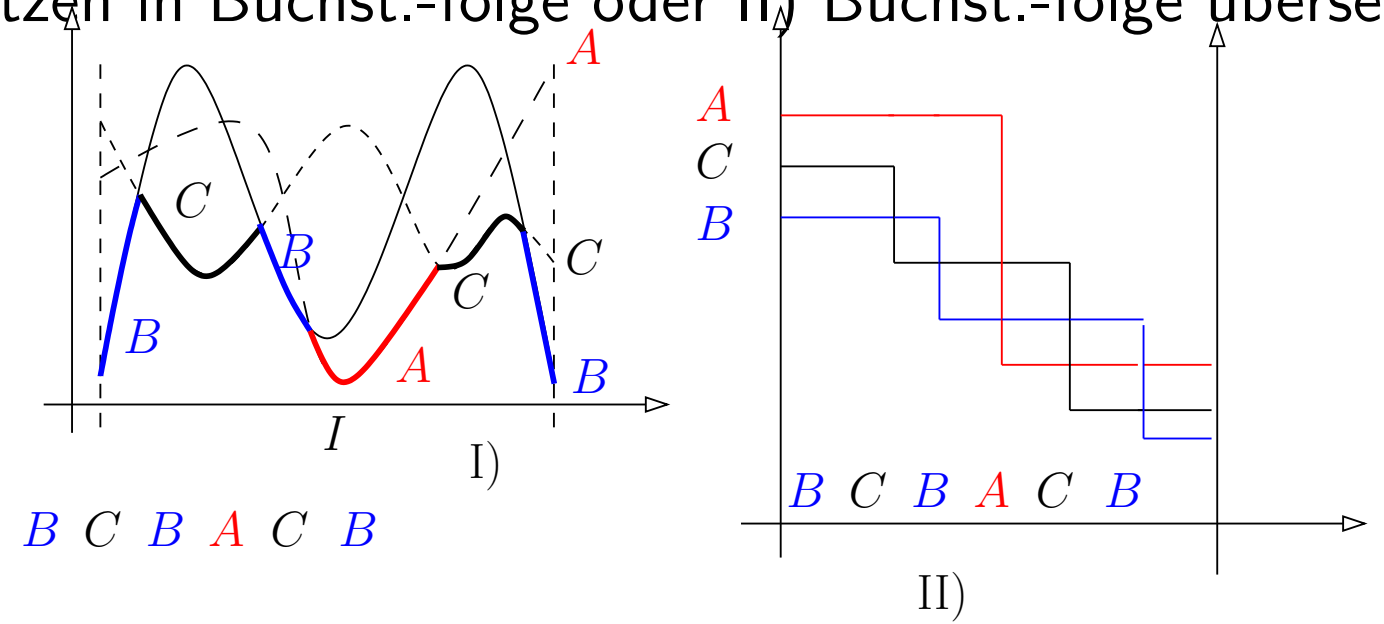
I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



Beweis: Untere Kontur/DSS

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

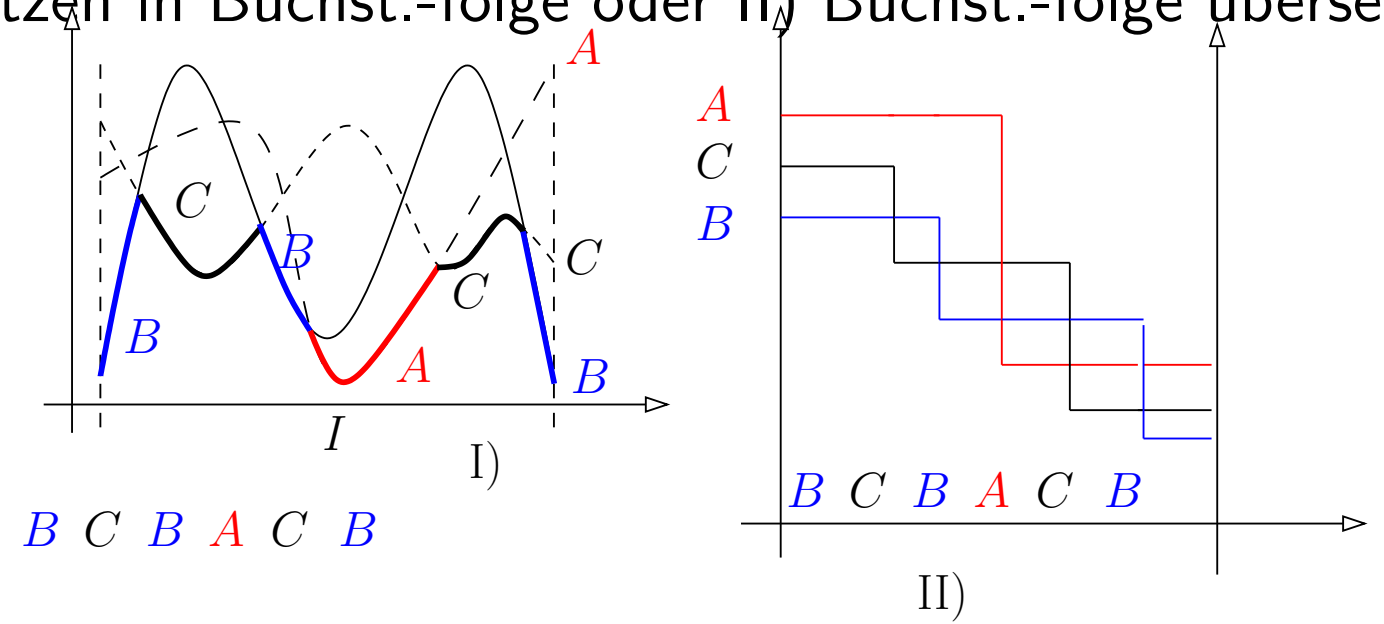
I) Übersetzen in Buchst.-folge oder II) Buchst.-folge übersetzen



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



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

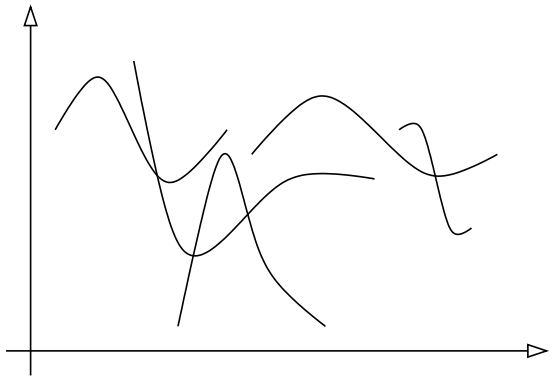
2. Verlängern auf gesamtes Intervall, dann 1. verwenden,

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,



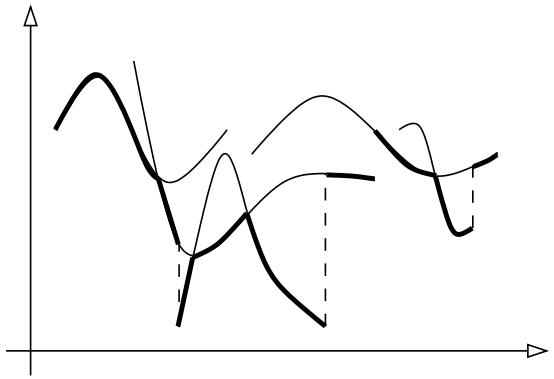
(2)

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,



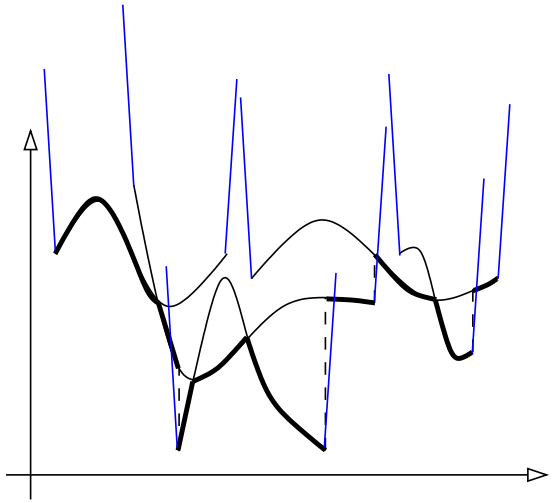
(2)

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,



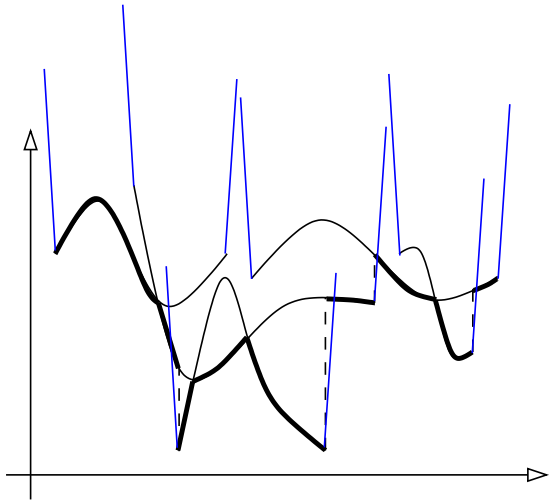
(2)

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,



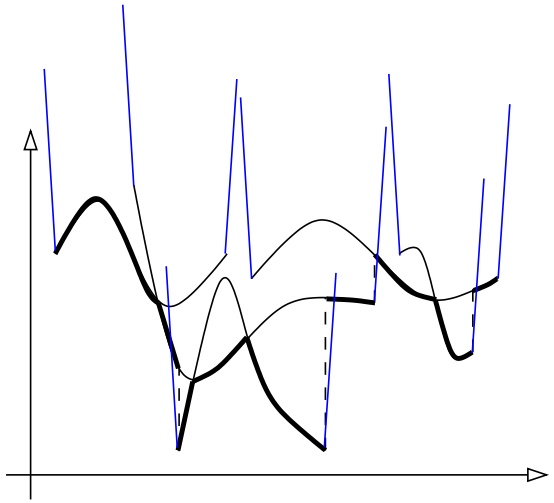
(2)

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 2.15 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.

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.