Karlsruher Institut für Technologie

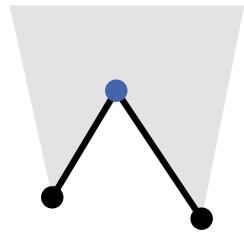
Algorithmische Geometrie

Aktivsession 2

Neue Merge- oder Splitknoten

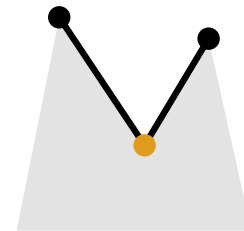
Split-Knoten

- Kanten liegen unten
- Polygon liegt oben

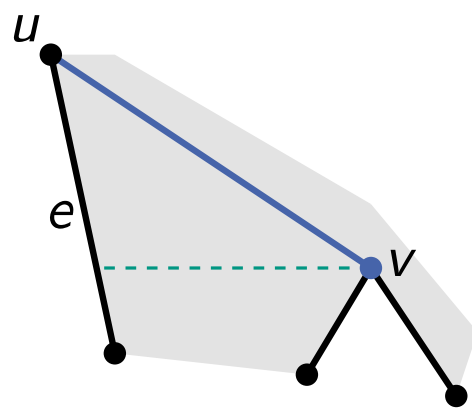


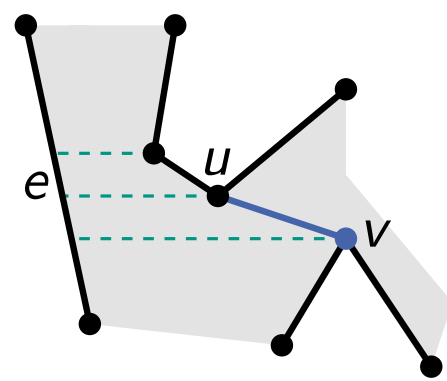
Merge-Knoten

- Kanten liegen oben
- Polygon liegt unten



Neue Diagonalen

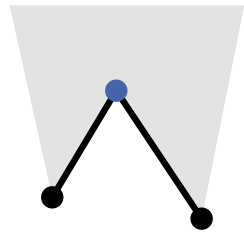




Neue Merge- oder Splitknoten

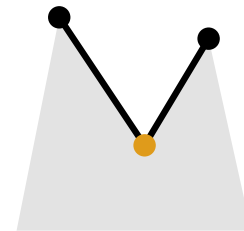
Split-Knoten

- Kanten liegen unten
- Polygon liegt oben

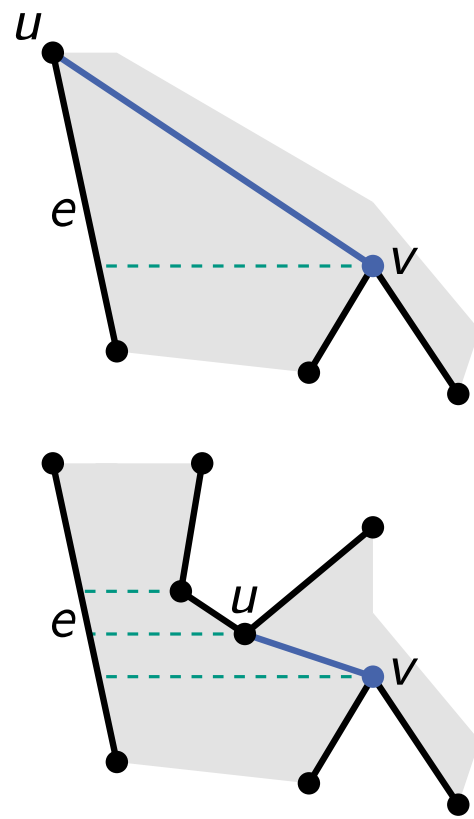


Merge-Knoten

- Kanten liegen oben
- Polygon liegt unten



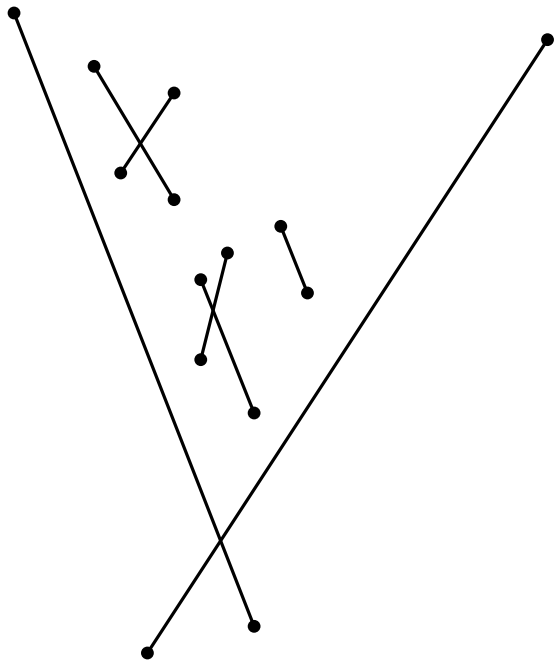
Neue Diagonalen



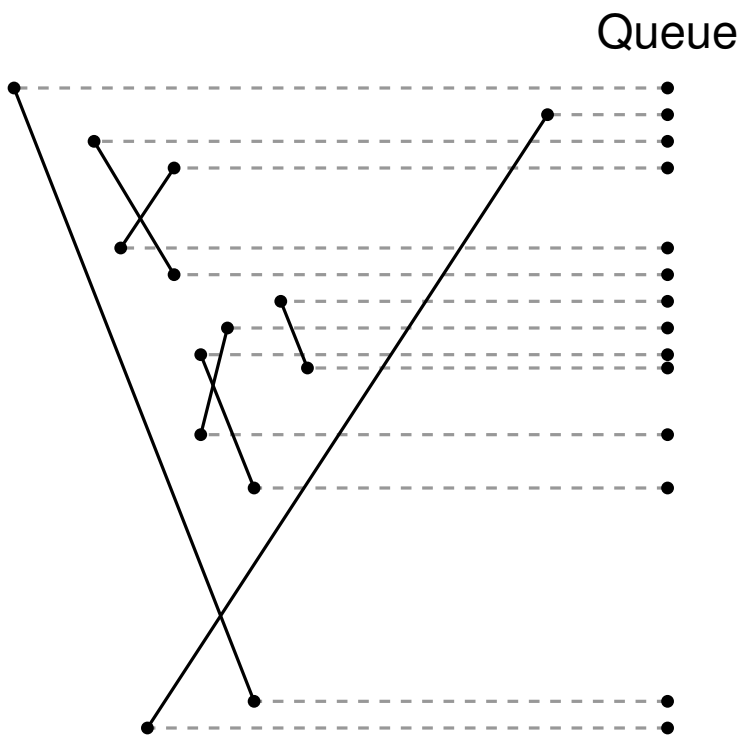
Warum entstehen beim Einfügen der Diagonalen keine neuen Merge- oder Splitkonten?

Kannst du ein möglichst schönes (einfaches) Argument finden?

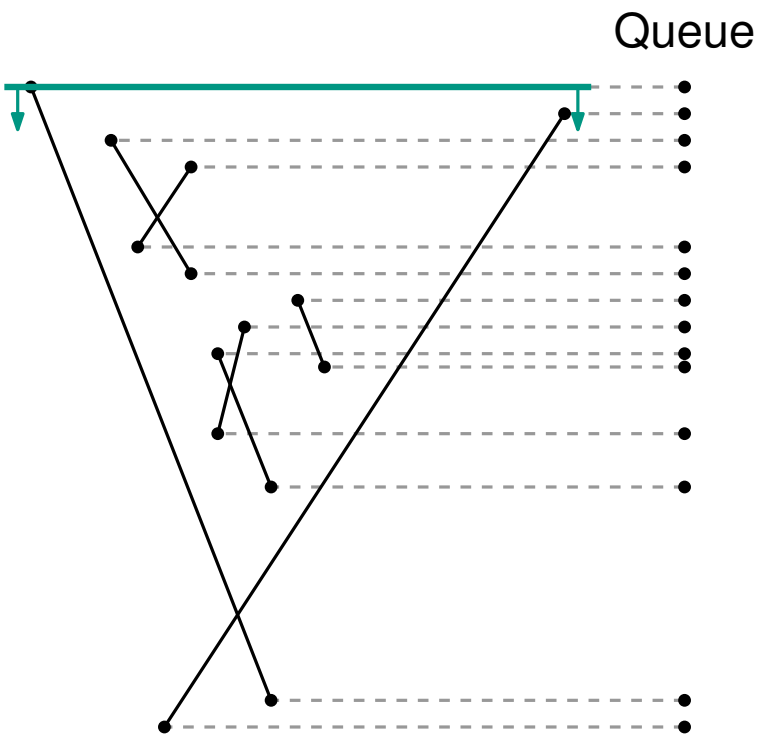
Speicherverbrauch beim Linienschnitt



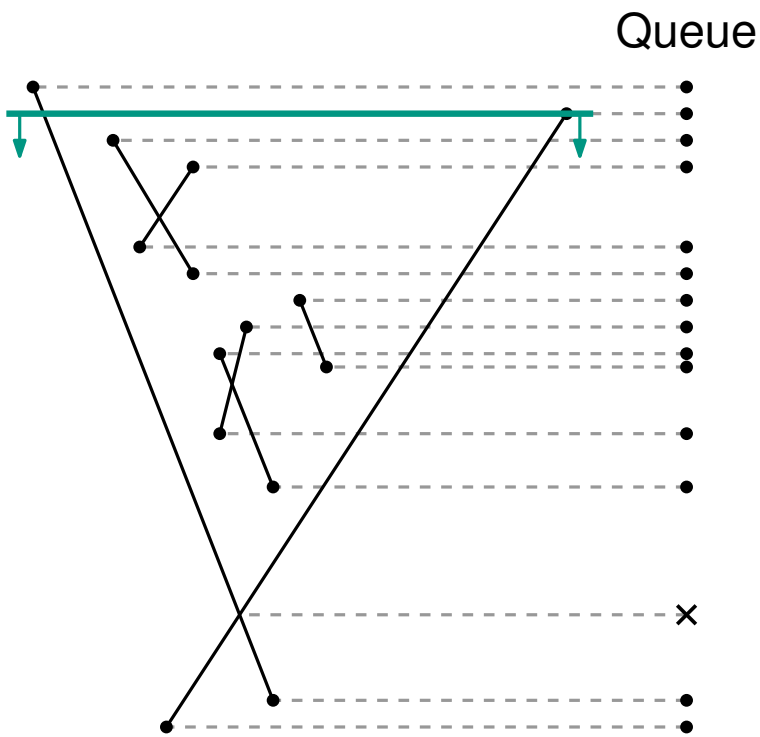
Speicherverbrauch beim Linienschnitt



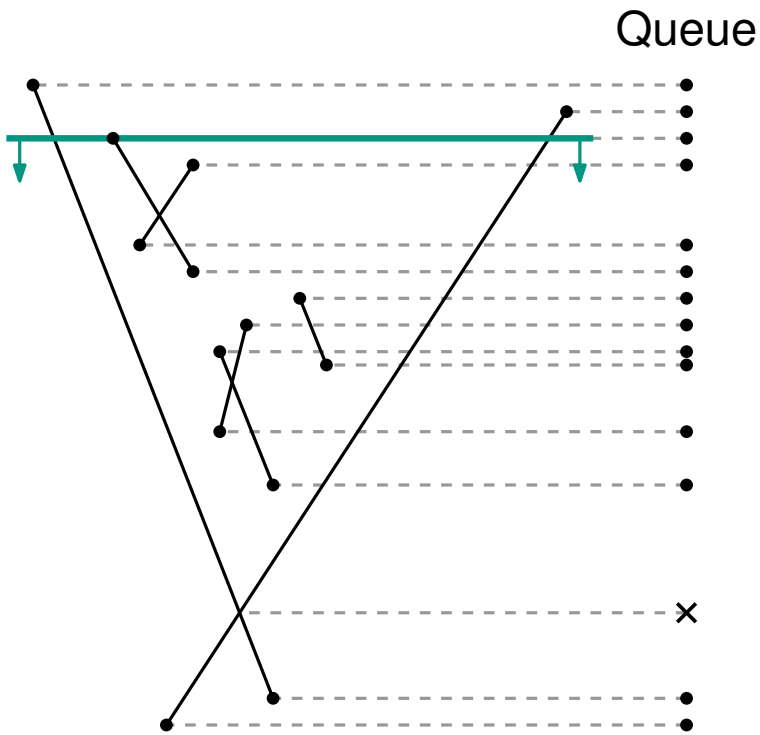
Speicherverbrauch beim Linienschnitt



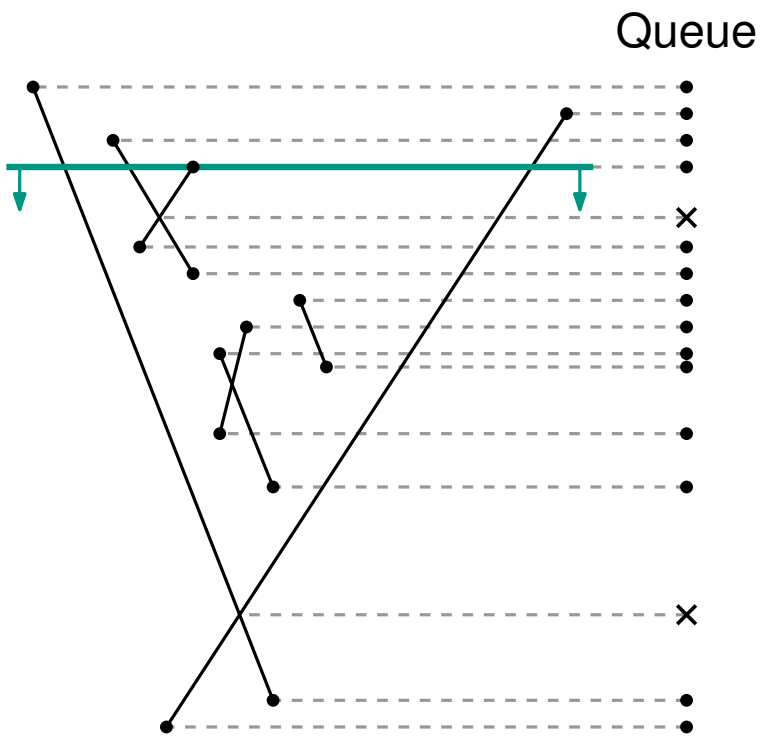
Speicherverbrauch beim Linienschnitt



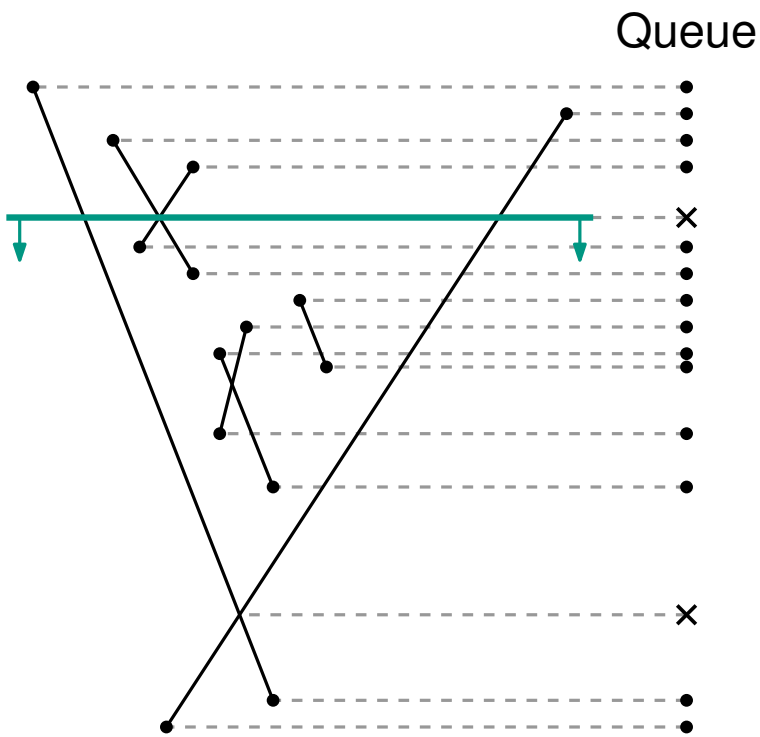
Speicherverbrauch beim Linienschnitt



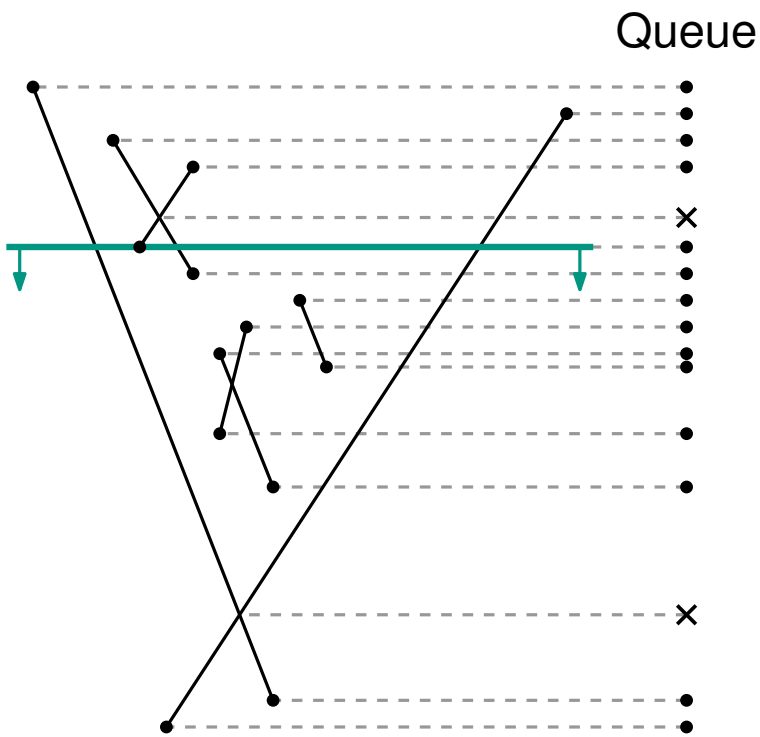
Speicherverbrauch beim Linienschnitt



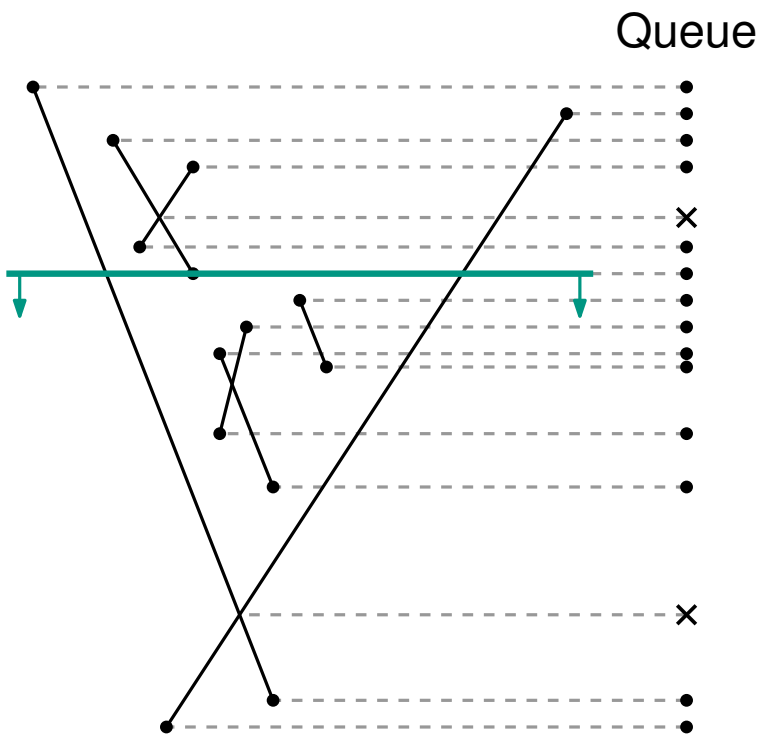
Speicherverbrauch beim Linienschnitt



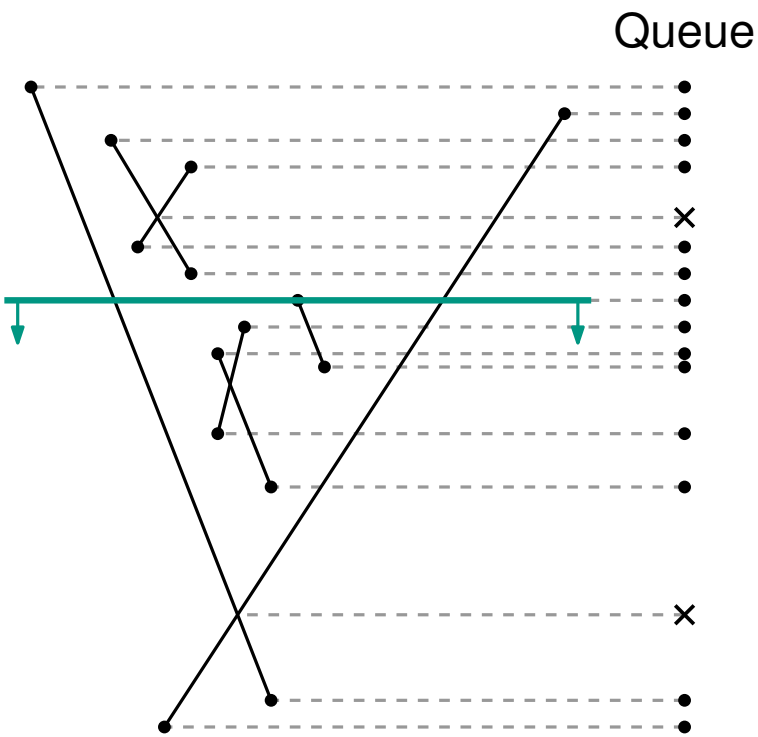
Speicherverbrauch beim Linienschnitt



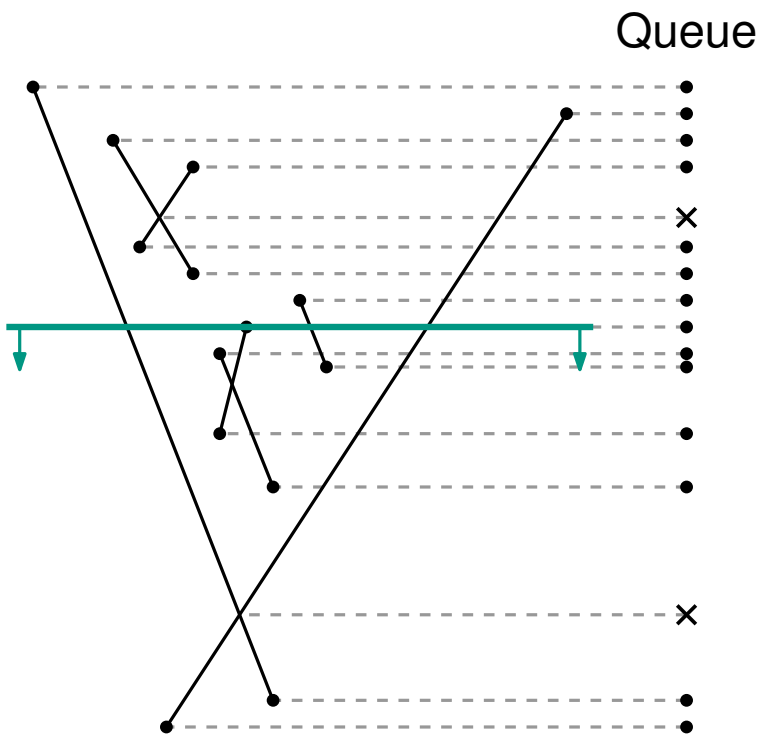
Speicherverbrauch beim Linienschnitt



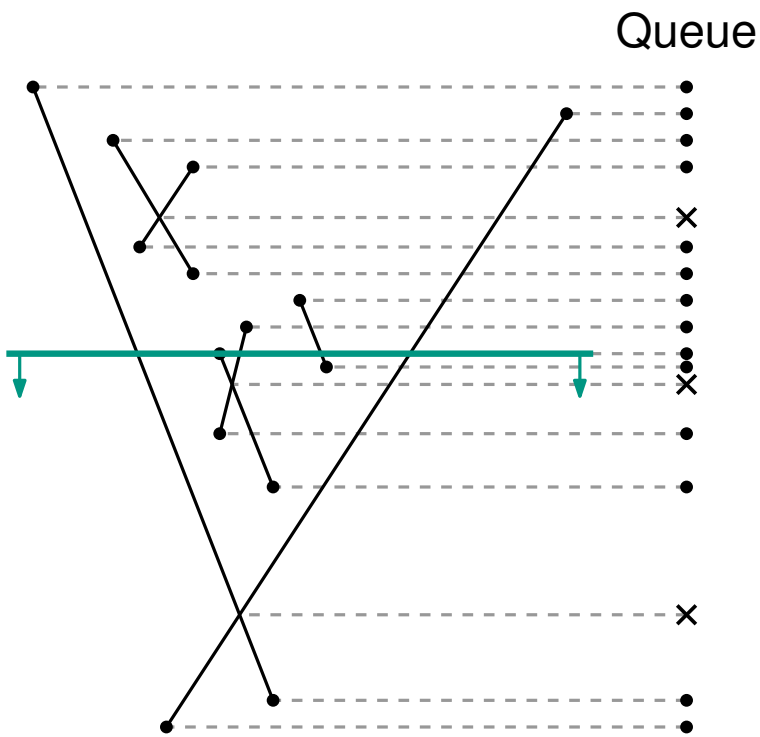
Speicherverbrauch beim Linienschnitt



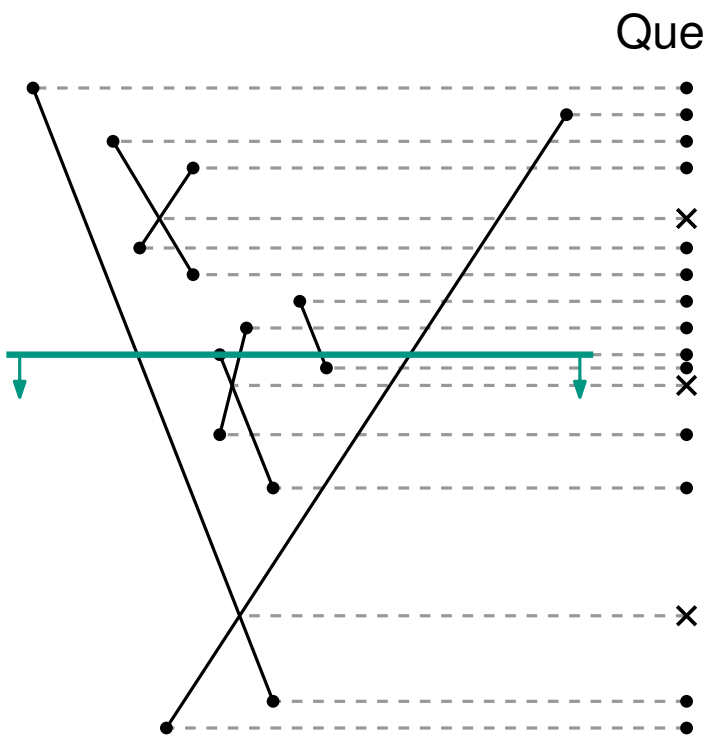
Speicherverbrauch beim Linienschnitt



Speicherverbrauch beim Linienschnitt



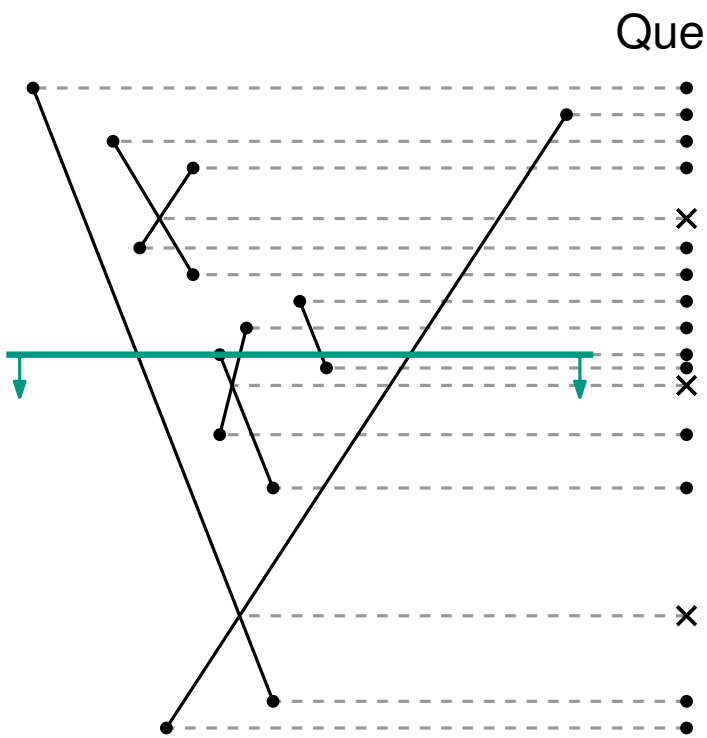
Speicherverbrauch beim Linienschnitt



Speicherverbrauch

- untere Schranke: $\Omega(n)$
  - alleine schon durch die Streckenenden
- obere Schranke: $O(n^2)$
  - es gibt nur $O(n^2)$ viele Schnittpunkte

Speicherverbrauch beim Linienschnitt

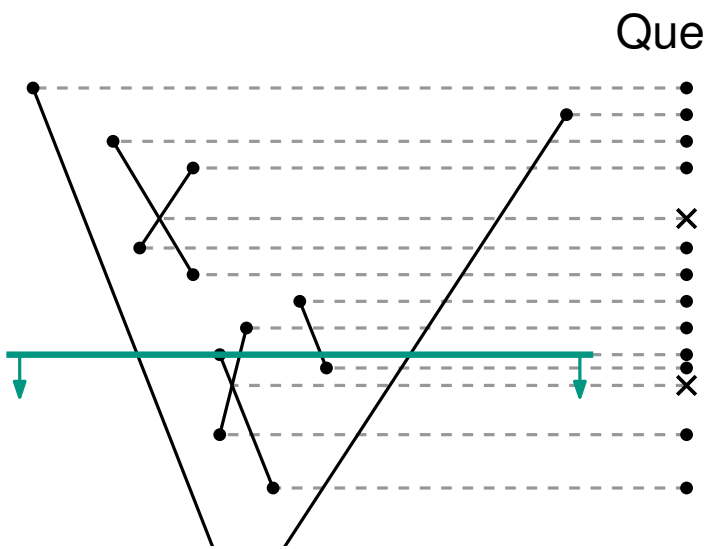


Speicherverbrauch

- untere Schranke: $\Omega(n)$
  - alleine schon durch die Streckenenden
- obere Schranke: $O(n^2)$
  - es gibt nur $O(n^2)$ viele Schnittpunkte

Könnt ihr Beispiele bauen, die tatsächlich $\omega(n)$ Speicher verbrauchen?

Speicherverbrauch beim Linienschnitt



Speicherverbrauch

- untere Schranke: $\Omega(n)$
  - alleine schon durch die Streckenenden
- obere Schranke: $O(n^2)$
  - es gibt nur $O(n^2)$ viele Schnittpunkte

Könnt ihr Beispiele bauen, die tatsächlich $\omega(n)$ Speicher verbrauchen?

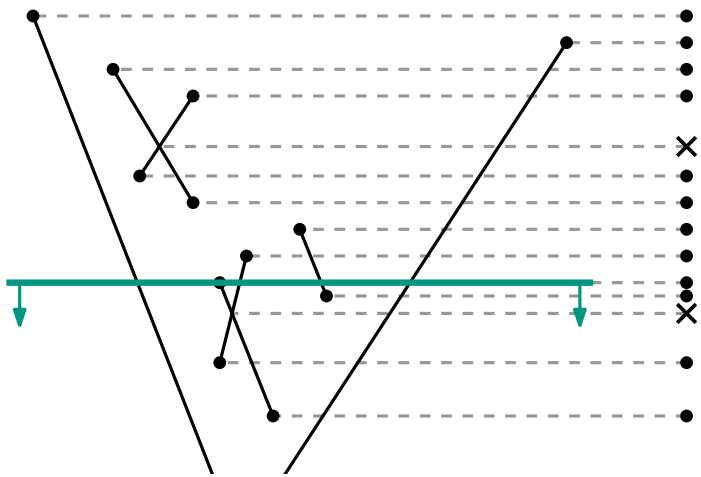
SIAM J. COMPUT.
Vol. 20, No. 3, pp. 460–470, June 1991

004

ON VERTICAL VISIBILITY IN ARRANGEMENTS OF SEGMENTS AND THE QUEUE SIZE IN THE BENTLEY–OTTMANN LINE SWEEPING ALGORITHM*

JÁNOS PACH† AND MICHA SHARIR‡

Speicherverbrauch beim Linienschnitt



Speicherverbrauch

- untere Schranke: $\Omega(n)$
  - alleine schon durch die Streckenenden
- obere Schranke: $O(n^2)$
  - es gibt nur $O(n^2)$ viele Schnittpunkte

Könnt ihr Beispiele bauen, die tatsächlich $\omega(n)$ Speicher verbrauchen?

SIAM J. COMPUT.
Vol. 20, No. 3, pp. 460–470, June 1991

004

ON VERTICAL VISIBILITY IN ARRANGEMENTS OF SEGMENTS AND THE QUEUE SIZE IN THE BENTLEY–OTTMANN LINE SWEEPING ALGORITHM*

JÁNOS PACH† AND MICHA SHARIR‡

Was zeigt das Papier und hilft uns das?

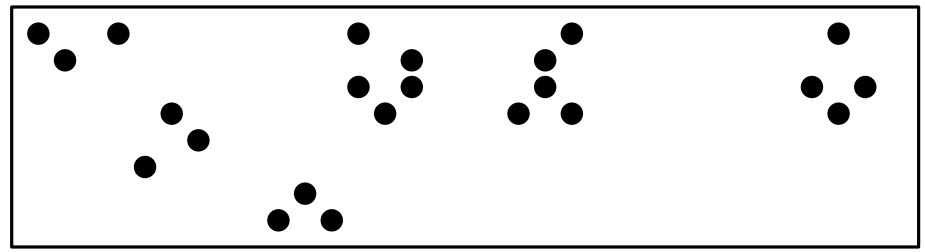
Wie funktioniert die Konstruktion der unteren Schranke?

Wie findet man das heraus, ohne das ganze Papier zu lesen?

Clustering

Ziel

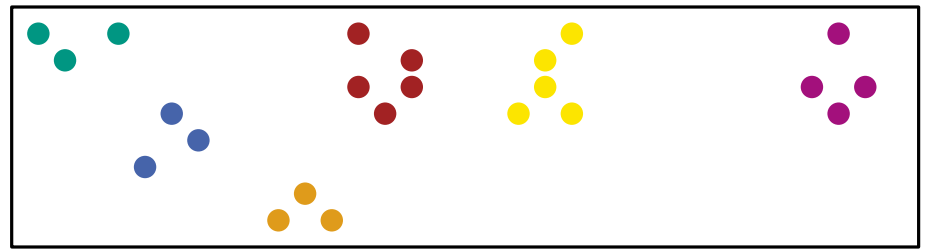
- gegeben eine Punktemenge
- finde ein sinnvolles Clustering



Clustering

Ziel

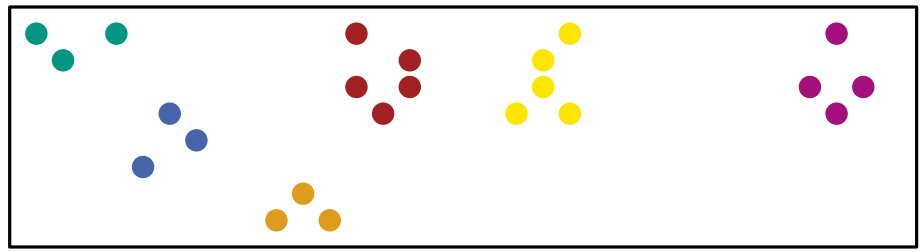
- gegeben eine Punktemenge
- finde ein sinnvolles Clustering

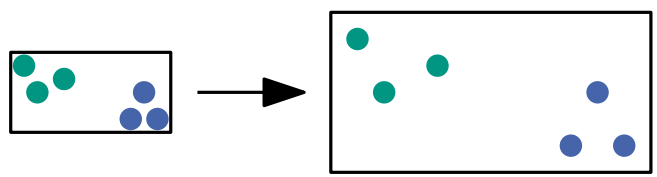


Clustering

Ziel

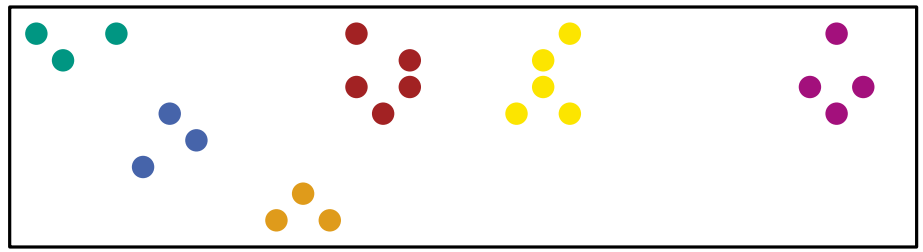
- gegeben eine Punktemenge
- finde ein sinnvolles Clustering
- Anforderungen an den Algorithmus:
  - **Skalierungs-Invarianz:** Skalierung aller Distanzen mit dem selben Faktor ändert das Clustering nicht



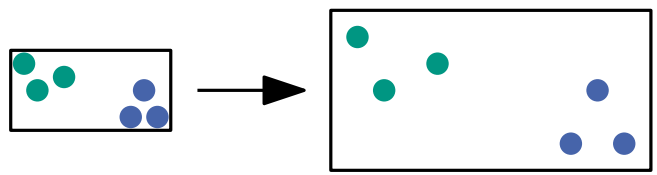


Clustering

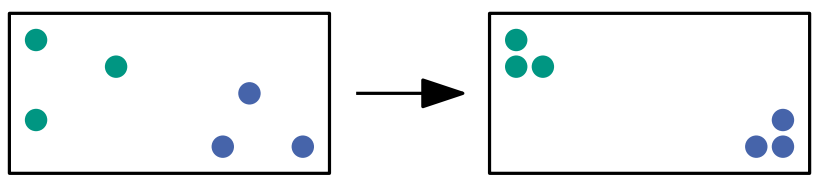
Ziel



- gegeben eine Punktemenge
- finde ein sinnvolles Clustering
- Anforderungen an den Algorithmus:
  - **Skalierungs-Invarianz:** Skalierung aller Distanzen mit dem selben Faktor ändert das Clustering nicht

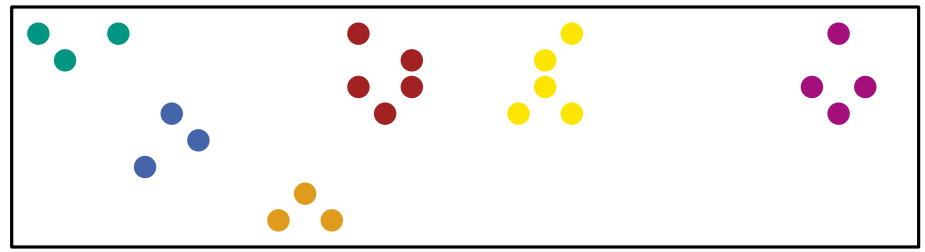


  - **Konsistenz:** Distanzen in Clustern verkleinern und zwischen Clustern vergrößern ändert das Clustering nicht

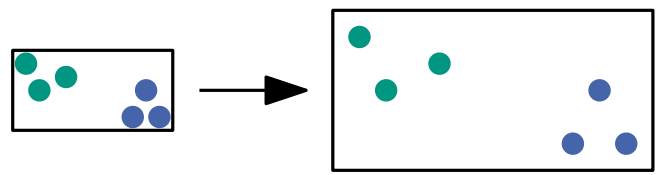


Clustering

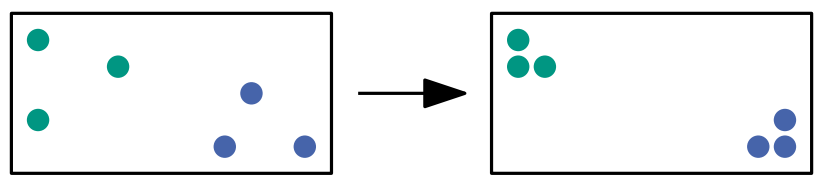
Ziel



- gegeben eine Punktemenge
- finde ein sinnvolles Clustering
- Anforderungen an den Algorithmus:
  - **Skalierungs-Invarianz:** Skalierung aller Distanzen mit dem selben Faktor ändert das Clustering nicht



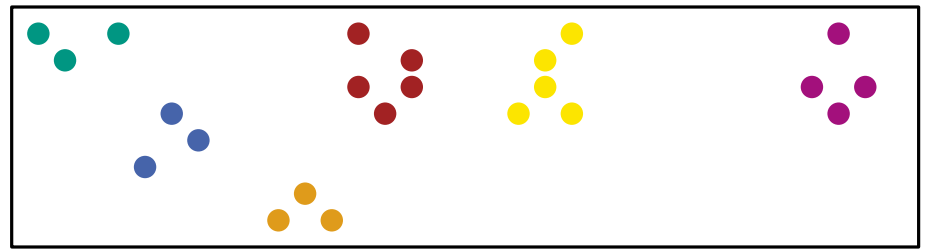
  - **Konsistenz:** Distanzen in Clustern verkleinern und zwischen Clustern vergrößern ändert das Clustering nicht



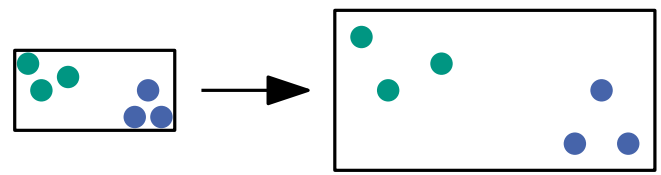
    - **Vielfalt:** jedes Clustering ist im Prinzip möglich

Clustering

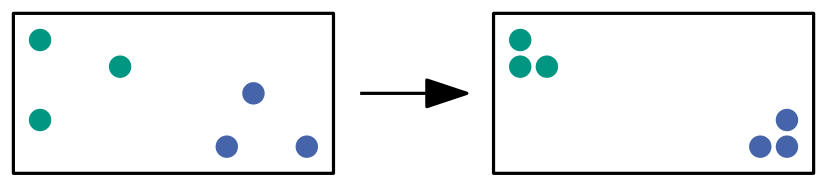
Ziel



- gegeben eine Punktemenge
- finde ein sinnvolles Clustering
- Anforderungen an den Algorithmus:
  - **Skalierungs-Invarianz:** Skalierung aller Distanzen mit dem selben Faktor ändert das Clustering nicht



  - **Konsistenz:** Distanzen in Clustern verkleinern und zwischen Clustern vergrößern ändert das Clustering nicht

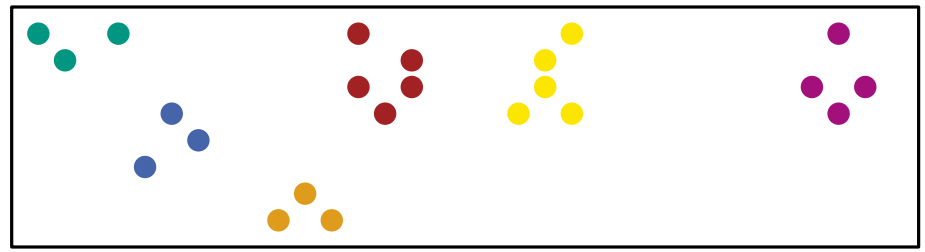


  - **Vielfalt:** jedes Clustering ist im Prinzip möglich

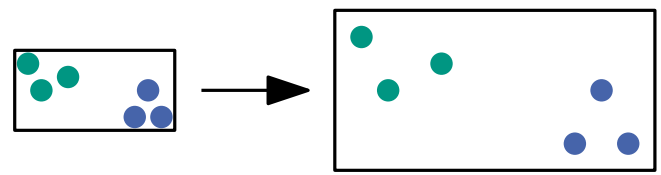
Warum kann es einen solchen Algorithmus nicht geben?

Clustering

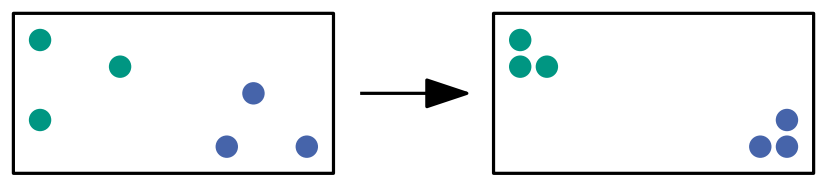
Ziel



- gegeben eine Punktemenge
- finde ein sinnvolles Clustering
- Anforderungen an den Algorithmus:
  - **Skalierungs-Invarianz:** Skalierung aller Distanzen mit dem selben Faktor ändert das Clustering nicht



  - **Konsistenz:** Distanzen in Clustern verkleinern und zwischen Clustern vergrößern ändert das Clustering nicht



  - **Vielfalt:** jedes Clustering ist im Prinzip möglich

Warum kann es einen solchen Algorithmus nicht geben?

Welche der drei Anforderungen würde ihr fallen lassen bzw. aufweichen?