Optimieren von Schnittplänen

Adrian Loy

30. Juli, 2015

Inhaltsverzeichnis

Worum geht es?

- Wir wollen möglichst effizient Druckbögen zerschneiden
- Dazu verwenden wir nur Guillotinen-Schnitte

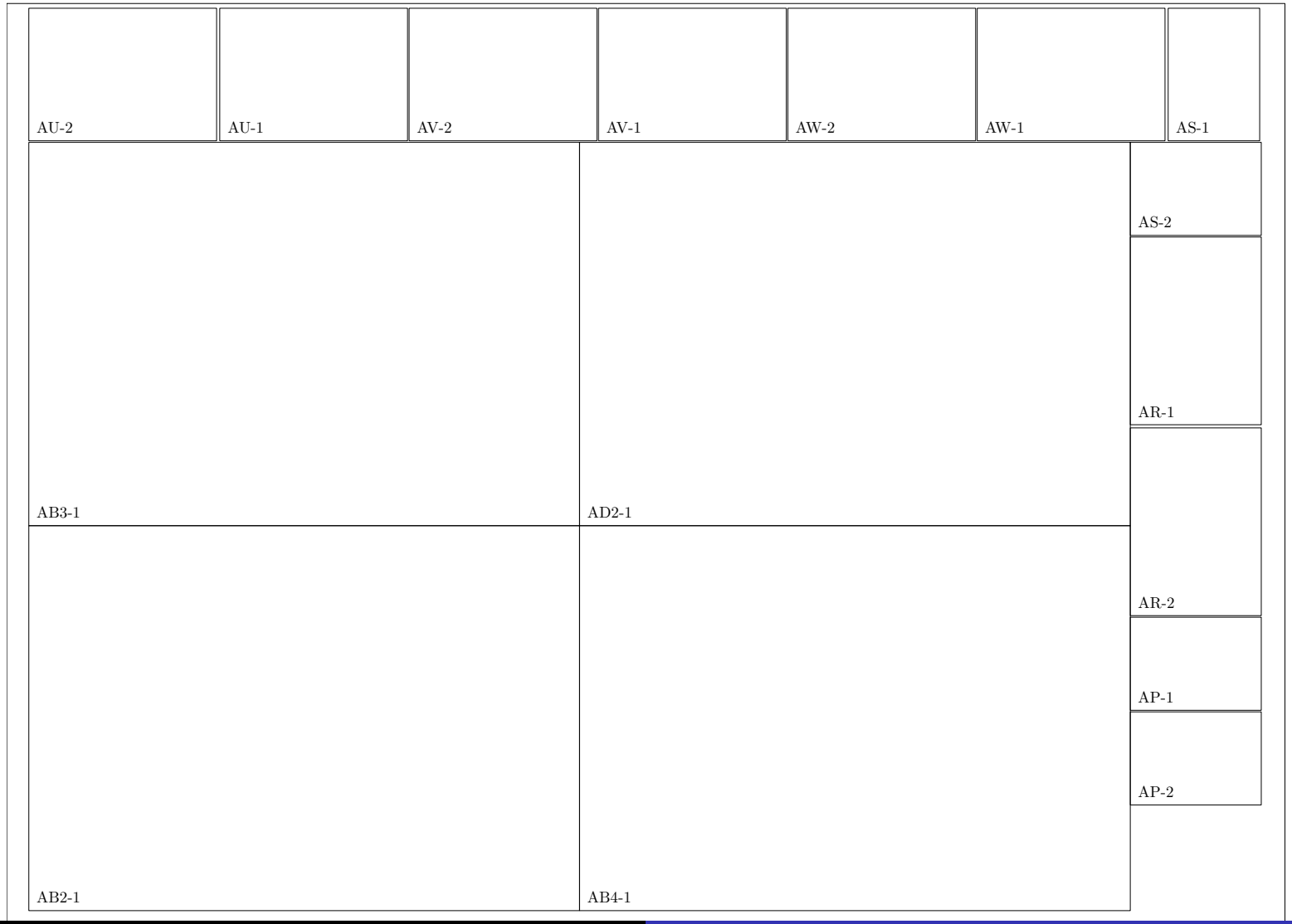
Worum geht es?

- Wir wollen möglichst effizient Druckbögen zerschneiden
- Dazu verwenden wir nur Guillotinen-Schnitte
- Die Anzahl der verwendeten Schnitte soll minimiert werden

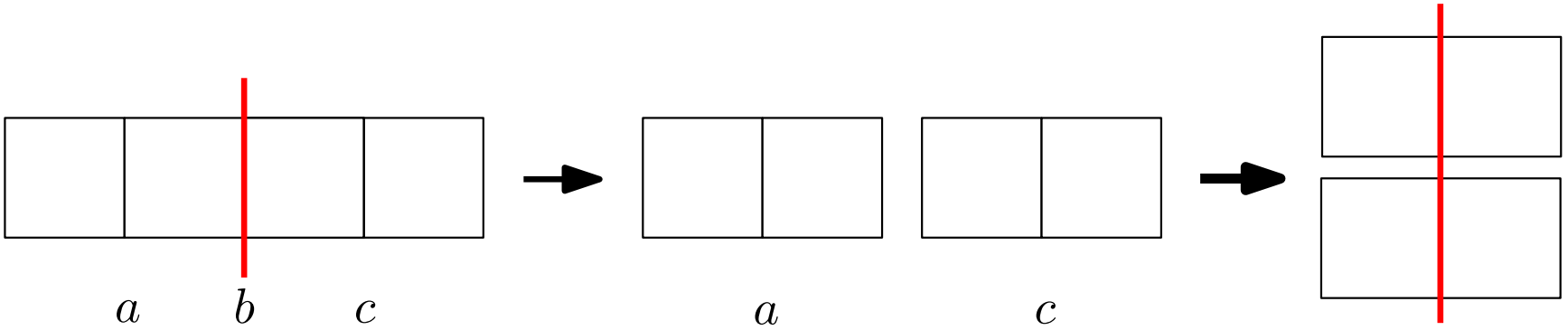
Worum geht es?

- Wir wollen möglichst effizient Druckbögen zerschneiden
- Dazu verwenden wir nur Guillotinen-Schnitte
- Die Anzahl der verwendeten Schnitte soll minimiert werden

Beispiel: Druckbogen

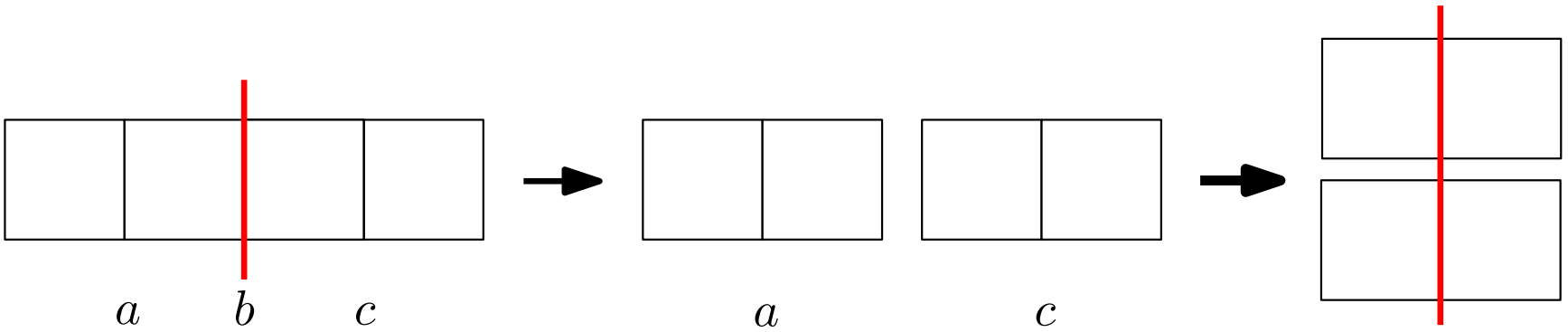


Einsparen von Schnitten



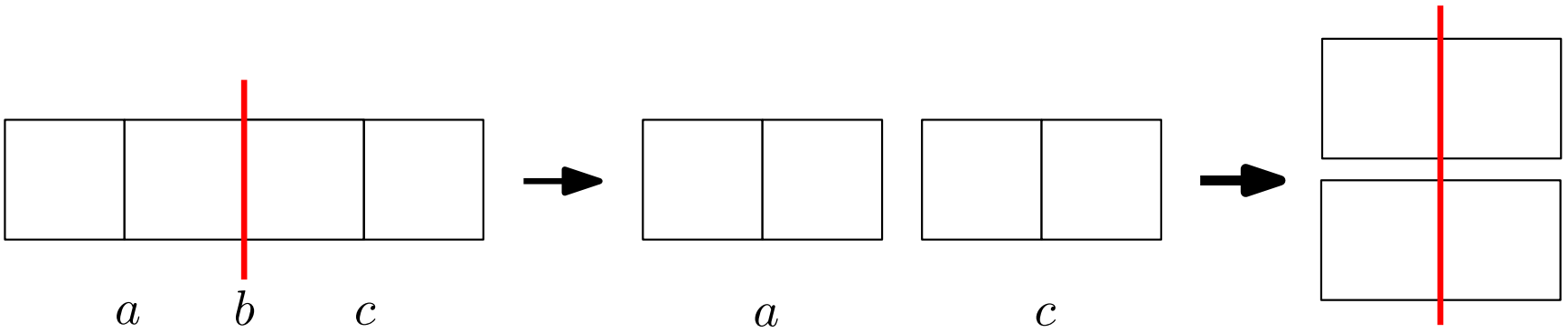
- Wir sprechen von *Schnittelementen*, *Schnittkanten* und *Blöcken*

Einsparen von Schnitten



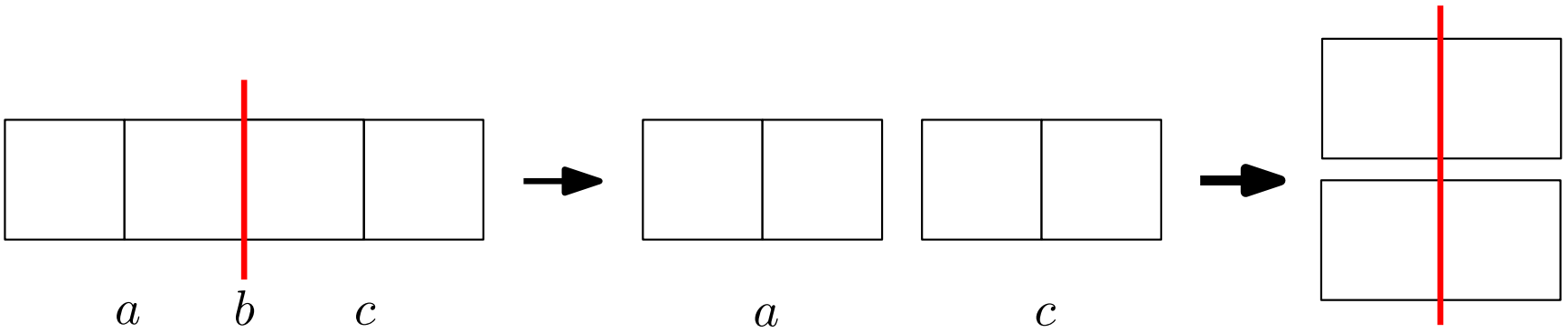
- Wir sprechen von *Schnittelementen*, *Schnittkanten* und *Blöcken*
- Schnittkanten können gemeinsam geschnitten werden, wenn sie in unterschiedlichen Blöcken liegen und mit gleichem Abstand geschnitten werden können

Einsparen von Schnitten



- Wir sprechen von *Schnittelementen*, *Schnittkanten* und *Blöcken*
- Schnittkanten können gemeinsam geschnitten werden, wenn sie in unterschiedlichen Blöcken liegen und mit gleichem Abstand geschnitten werden können

Einsparen von Schnitten



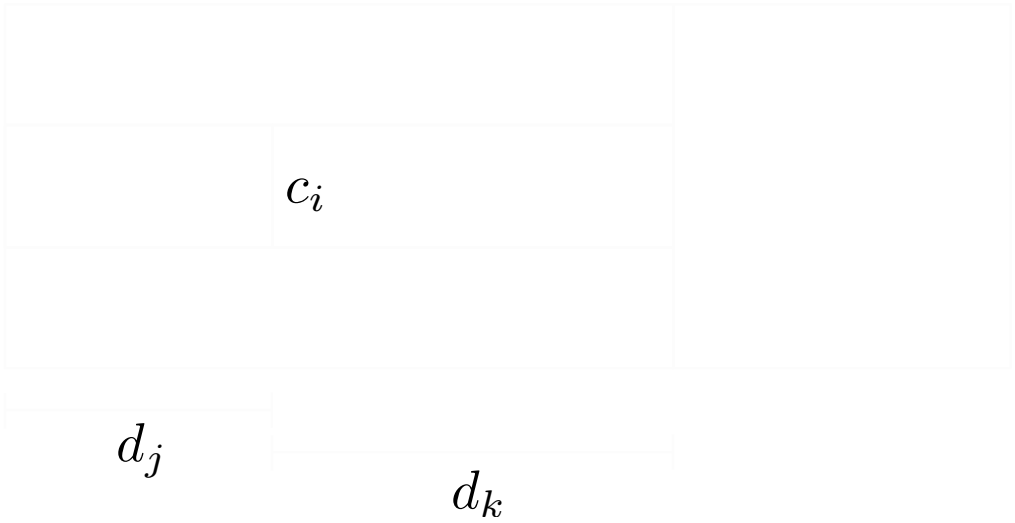
- Wir sprechen von *Schnittelementen*, *Schnittkanten* und *Blöcken*
- Schnittkanten können gemeinsam geschnitten werden, wenn sie in unterschiedlichen Blöcken liegen und mit gleichem Abstand geschnitten werden können

Vertex Cover

- *gegeben:* Ungerichteter Graph $G = (V, E)$
- *gesucht:* Minimale Teilmenge von Knoten $V' \subseteq V$, sodass jede Kante von G einen Knoten aus V' enthält

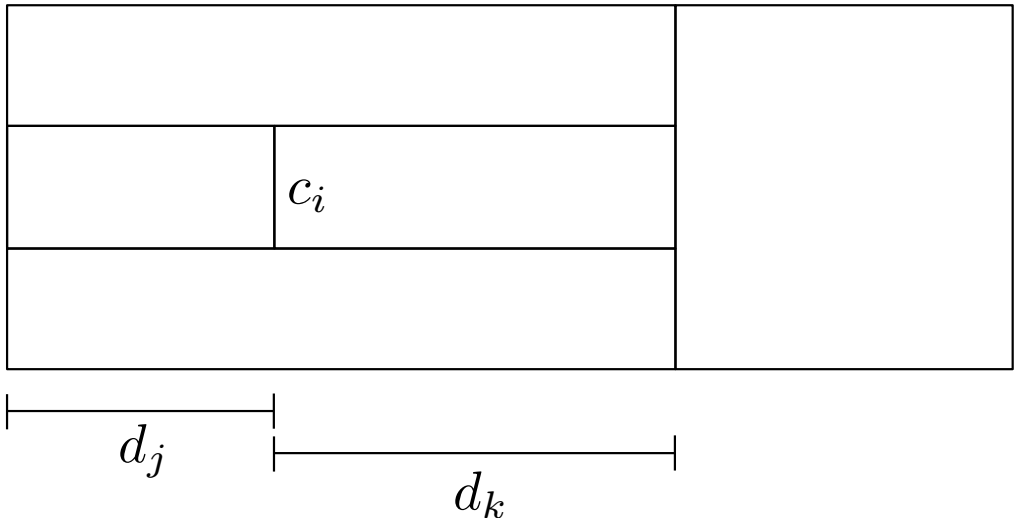
Konstruieren von Teilstreifen

Wir konstruieren für jede Kante $e_i = \{v_j, v_k\}$ einen Teilstreifen:



Konstruieren von Teilstreifen

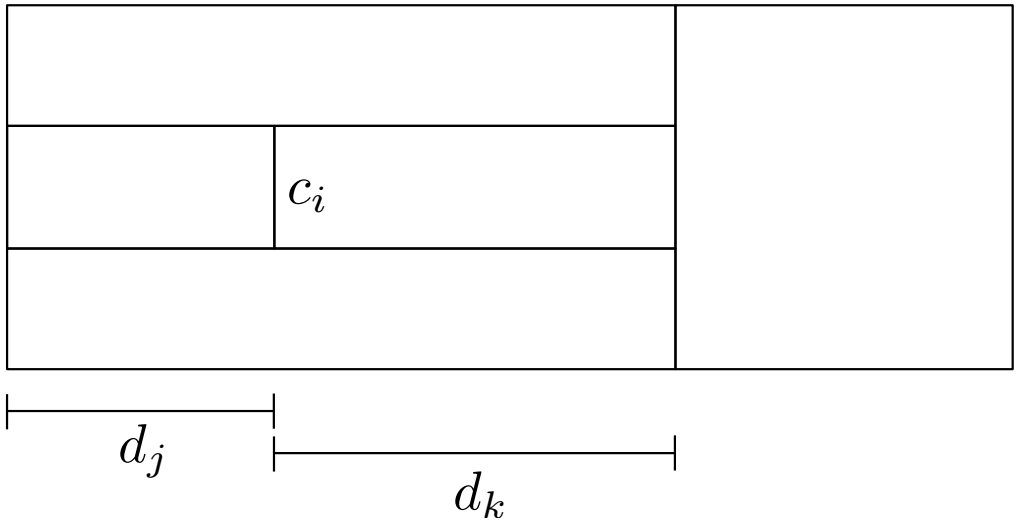
Wir konstruieren für jede Kante $e_i = \{v_j, v_k\}$ einen Teilstreifen:



- Die Kante wird durch eine Schnittkanten im Bogen repräsentiert

Konstruieren von Teilstreifen

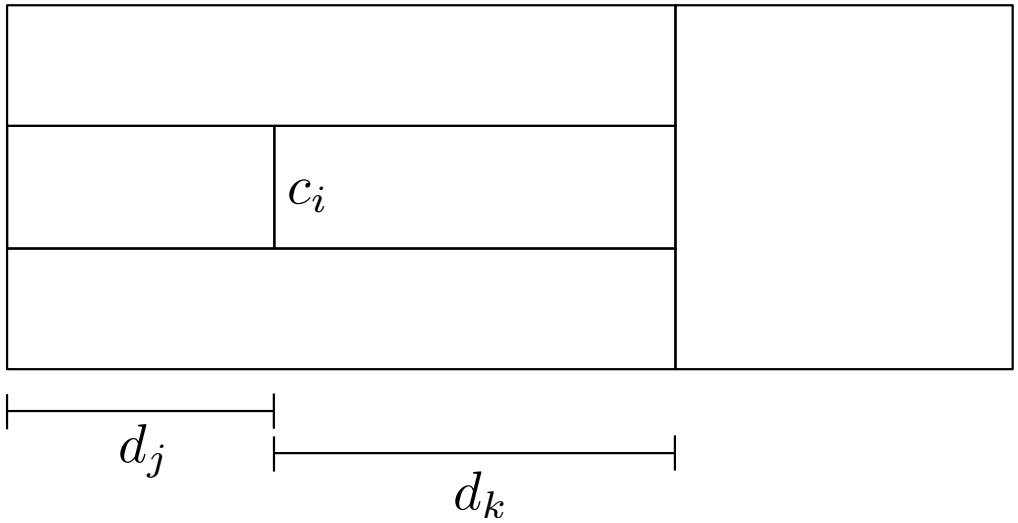
Wir konstruieren für jede Kante $e_i = \{v_j, v_k\}$ einen Teilstreifen:



- Die Kante wird durch eine Schnittkanten im Bogen repräsentiert
- Die repräsentativen Schnittkanten lassen sich mit *genau* zwei möglichen Distanzen schneiden

Konstruieren von Teilstreifen

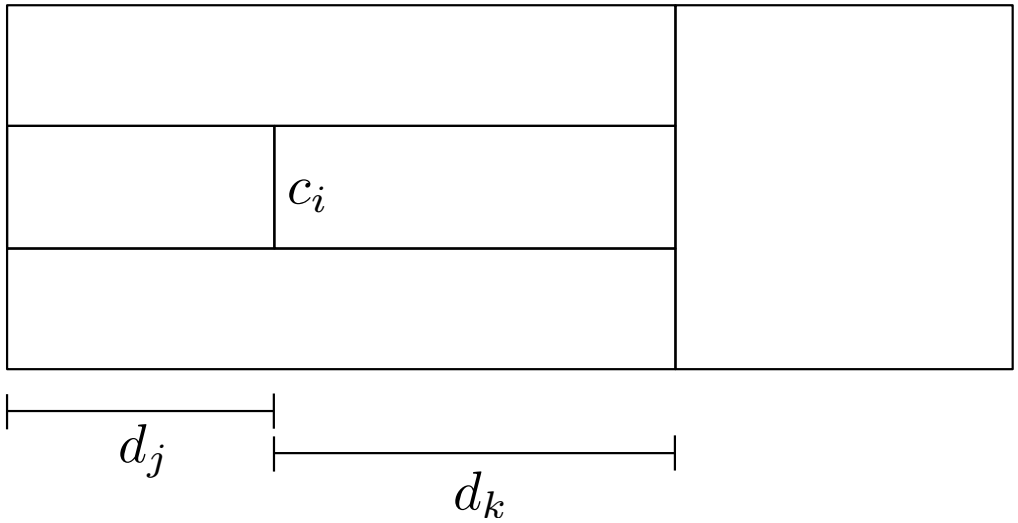
Wir konstruieren für jede Kante $e_i = \{v_j, v_k\}$ einen Teilstreifen:



- Die Kante wird durch eine Schnittkanten im Bogen repräsentiert
- Die repräsentativen Schnittkanten lassen sich mit *genau* zwei möglichen Distanzen schneiden
- Jedem Knoten wird eine eindeutige Distanz zugewiesen

Konstruieren von Teilstreifen

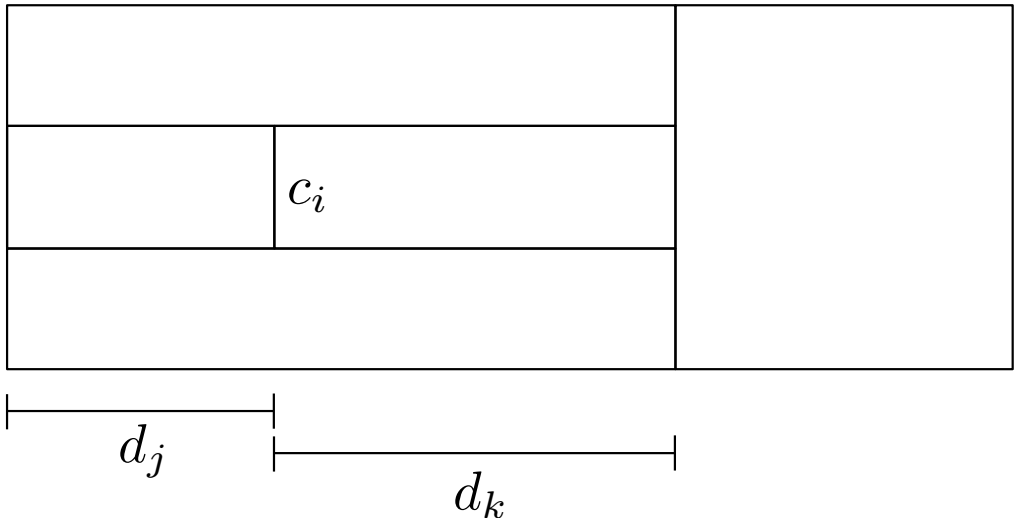
Wir konstruieren für jede Kante $e_i = \{v_j, v_k\}$ einen Teilstreifen:



- Die Kante wird durch eine Schnittkanten im Bogen repräsentiert
- Die repräsentativen Schnittkanten lassen sich mit *genau* zwei möglichen Distanzen schneiden
- Jedem Knoten wird eine eindeutige Distanz zugewiesen
- Diese repräsentieren die Endknoten der Kante

Konstruieren von Teilstreifen

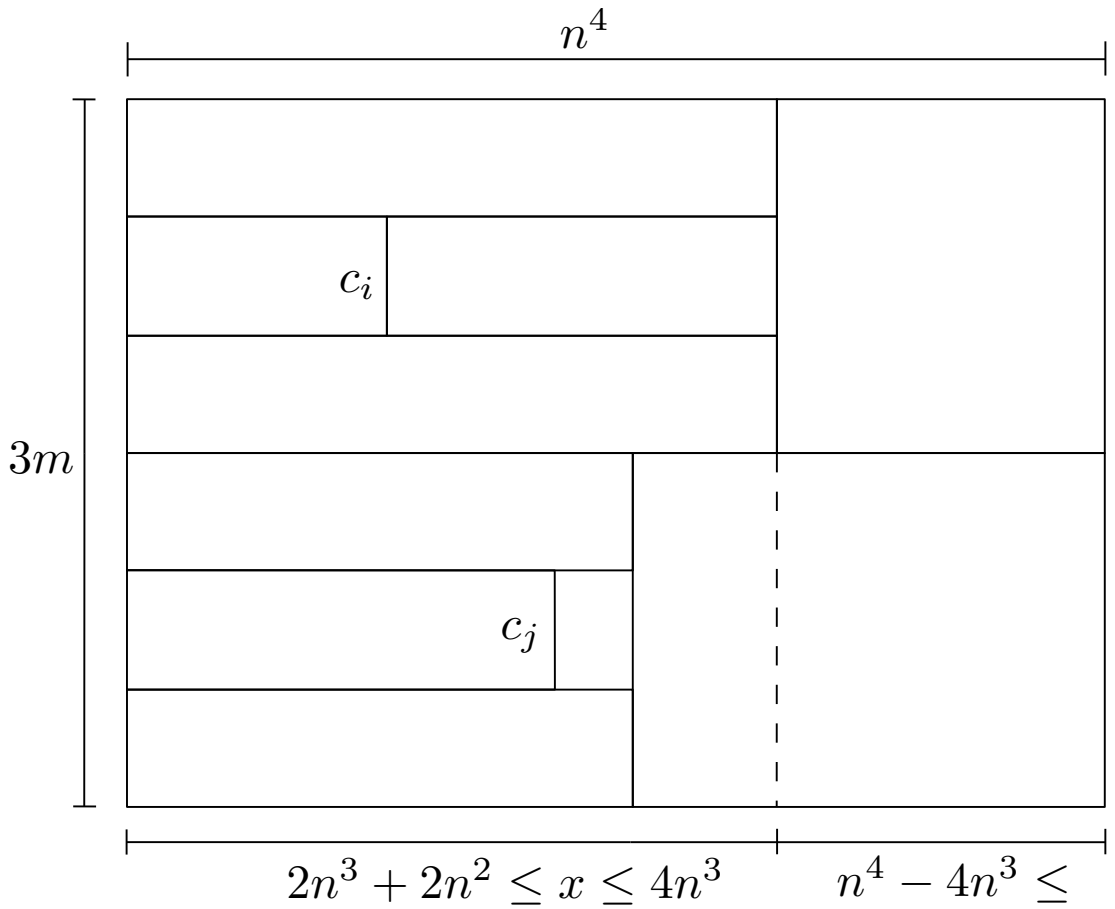
Wir konstruieren für jede Kante $e_i = \{v_j, v_k\}$ einen Teilstreifen:



- Die Kante wird durch eine Schnittkanten im Bogen repräsentiert
- Die repräsentativen Schnittkanten lassen sich mit *genau* zwei möglichen Distanzen schneiden
- Jedem Knoten wird eine eindeutige Distanz zugewiesen
- Diese repräsentieren die Endknoten der Kante

Konstruieren von Bögen

Aufbau des gesamten Bogens. Ein Knoten v_i im Graphen wird durch den Abstand $d_i = (n + i)n^2$ repräsentiert.



Reduktion

- Bei diesen Abständen können repräsentative Kanten nur mit anderen repräsentativen Kanten gemeinsam geschnitten werden
- Ein Schnittplan mit minimaler Anzahl an Schnitten benötigt auch für die repräsentativen Schnittkanten möglichst wenig Schnitte

Reduktion

- Bei diesen Abständen können repräsentative Kanten nur mit anderen repräsentativen Kanten gemeinsam geschnitten werden
- Ein Schnittplan mit minimaler Anzahl an Schnitten benötigt auch für die repräsentativen Schnittkanten möglichst wenig Schnitte
- Die dafür gewählten Abstände entsprechen dann einer minimalen Anzahl an Knoten, welche alle Kanten abdecken

Reduktion

- Bei diesen Abständen können repräsentative Kanten nur mit anderen repräsentativen Kanten gemeinsam geschnitten werden
- Ein Schnittplan mit minimaler Anzahl an Schnitten benötigt auch für die repräsentativen Schnittkanten möglichst wenig Schnitte
- Die dafür gewählten Abstände entsprechen dann einer minimalen Anzahl an Knoten, welche alle Kanten abdecken
- → Das Finden von optimalen Schnittplänen ist ein NP-vollständiges Problem!

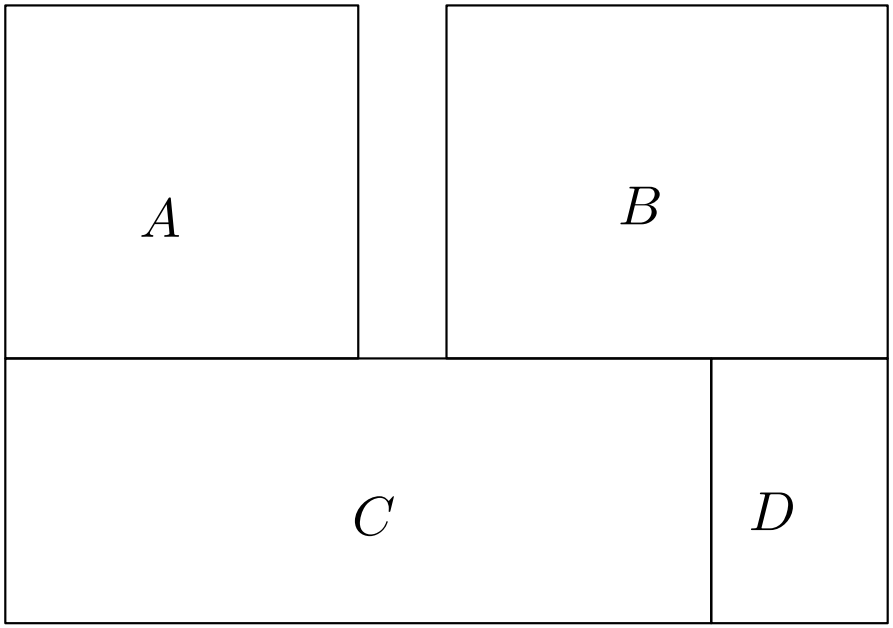
Reduktion

- Bei diesen Abständen können repräsentative Kanten nur mit anderen repräsentativen Kanten gemeinsam geschnitten werden
- Ein Schnittplan mit minimaler Anzahl an Schnitten benötigt auch für die repräsentativen Schnittkanten möglichst wenig Schnitte
- Die dafür gewählten Abstände entsprechen dann einer minimalen Anzahl an Knoten, welche alle Kanten abdecken
- $\rightarrow$ Das Finden von optimalen Schnittplänen ist ein NP-vollständiges Problem!

Überdeckungsfreie Nachbarn

Zwei Elemente sind überdeckungsfrei benachbart, wenn gilt:

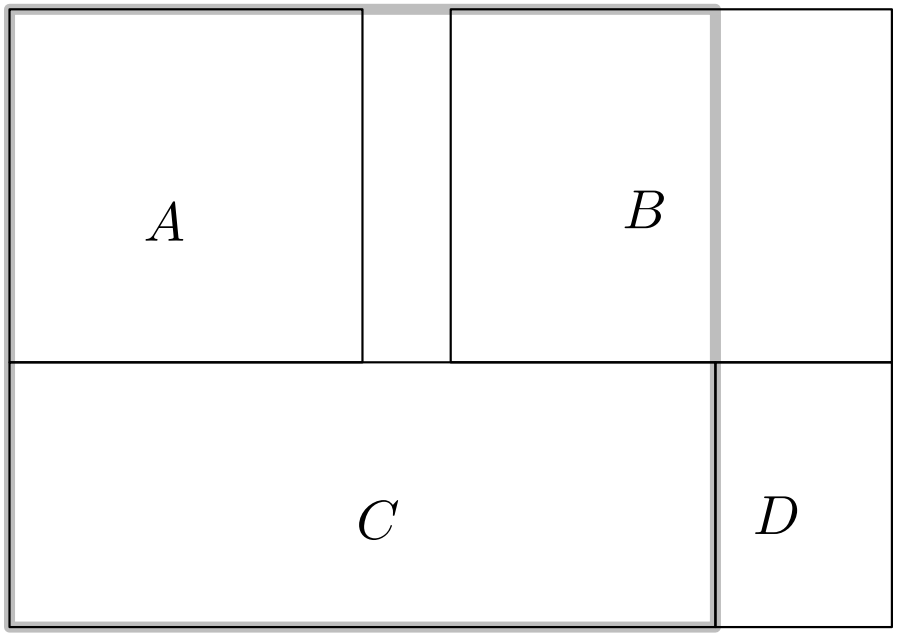
- Das kleinstmögliche, beide überdeckende Rechteck überlappt mit keinem anderen Element



Überdeckungsfreie Nachbarn

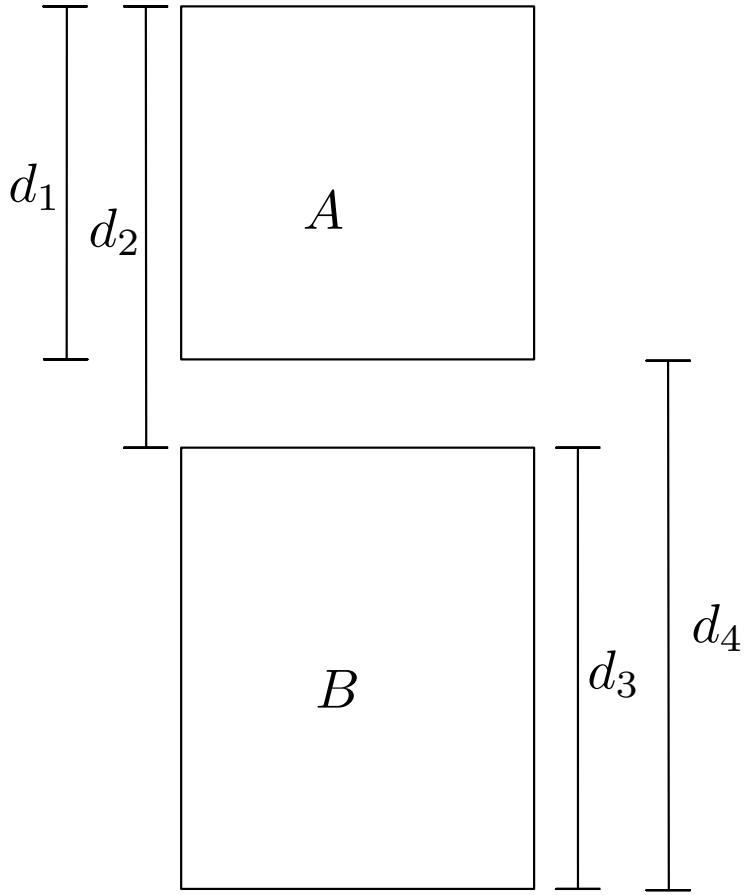
Zwei Elemente sind überdeckungsfrei benachbart, wenn gilt:

- Das kleinstmögliche, beide überdeckende Rechteck überlappt mit keinem anderen Element

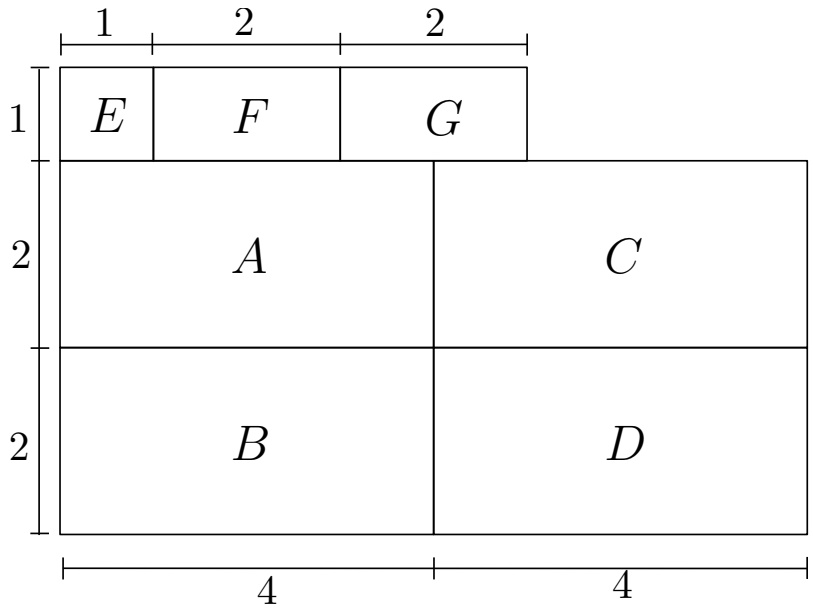


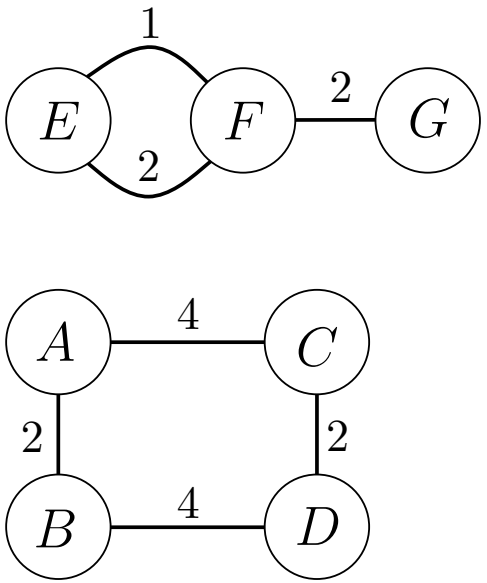
Schnittdistanzen

Falls zwei Elemente einen Block bilden, gibt es vier mögliche Schnittdistanzen:



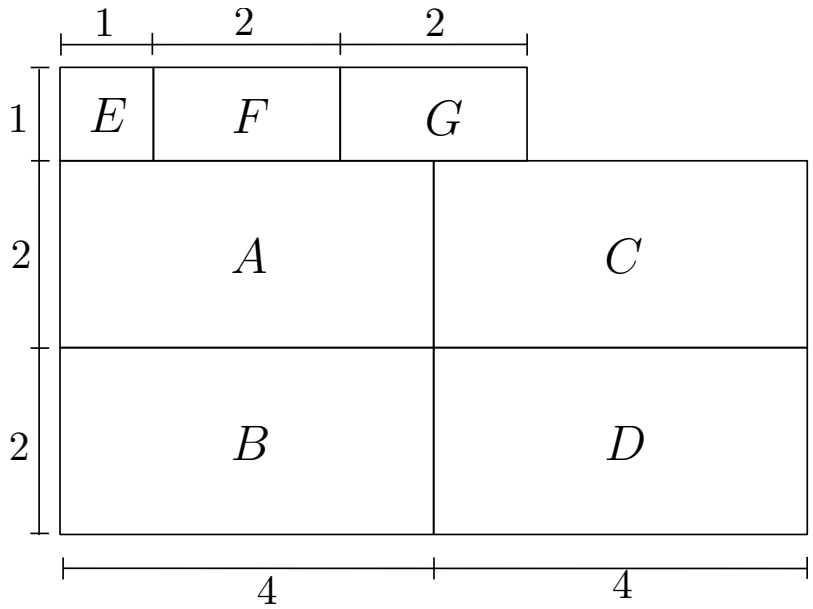
Algorithmus: Vorgehensweise

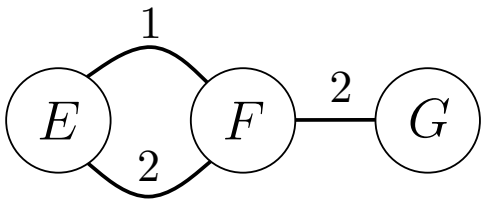


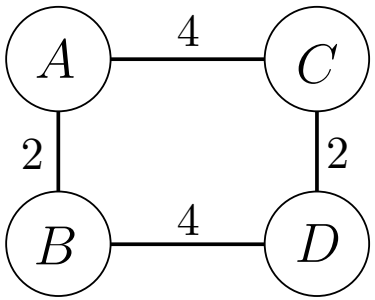


- Kanten verlaufen zwischen überdeckungsfreien Nachbarn
- Kantengewichte stehen für mögliche Distanzen

Algorithmus: Vorgehensweise

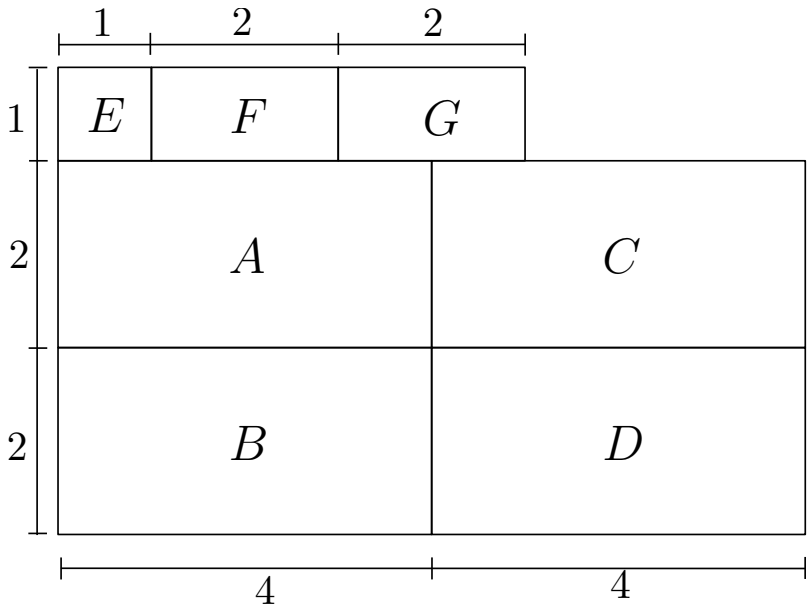


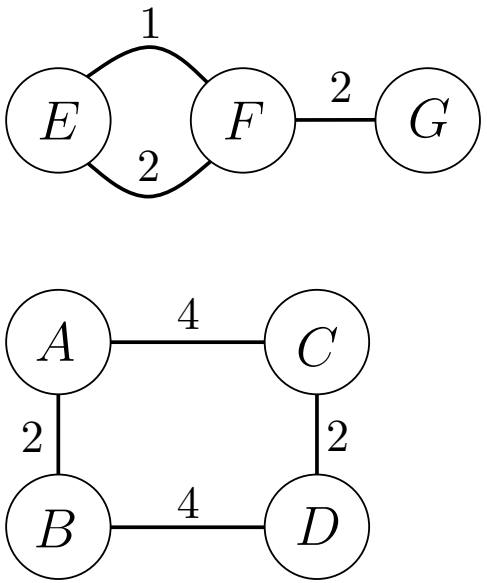




- Kanten verlaufen zwischen überdeckungsfreien Nachbarn
- Kantengewichte stehen für mögliche Distanzen
- Verschmelze iterativ Elemente

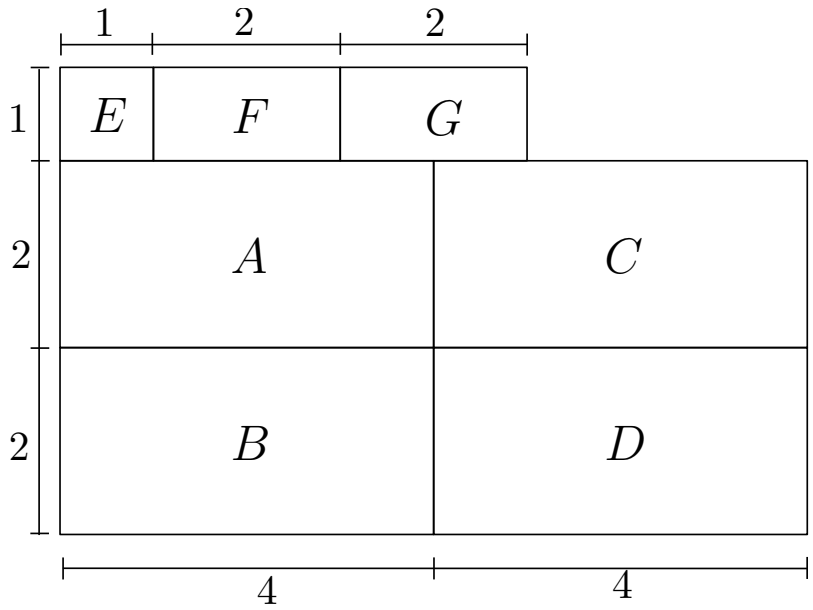
Algorithmus: Vorgehensweise

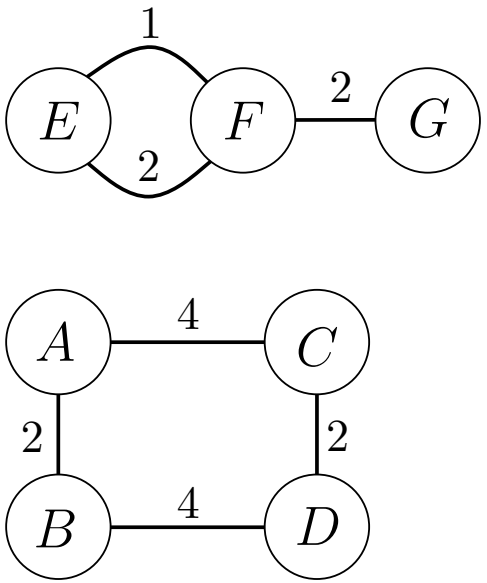




- Kanten verlaufen zwischen überdeckungsfreien Nachbarn
- Kantengewichte stehen für mögliche Distanzen
- Verschmelze iterativ Elemente
- Füge jeweils die Schnitte hinzu, die diese Verschmelzung umkehren

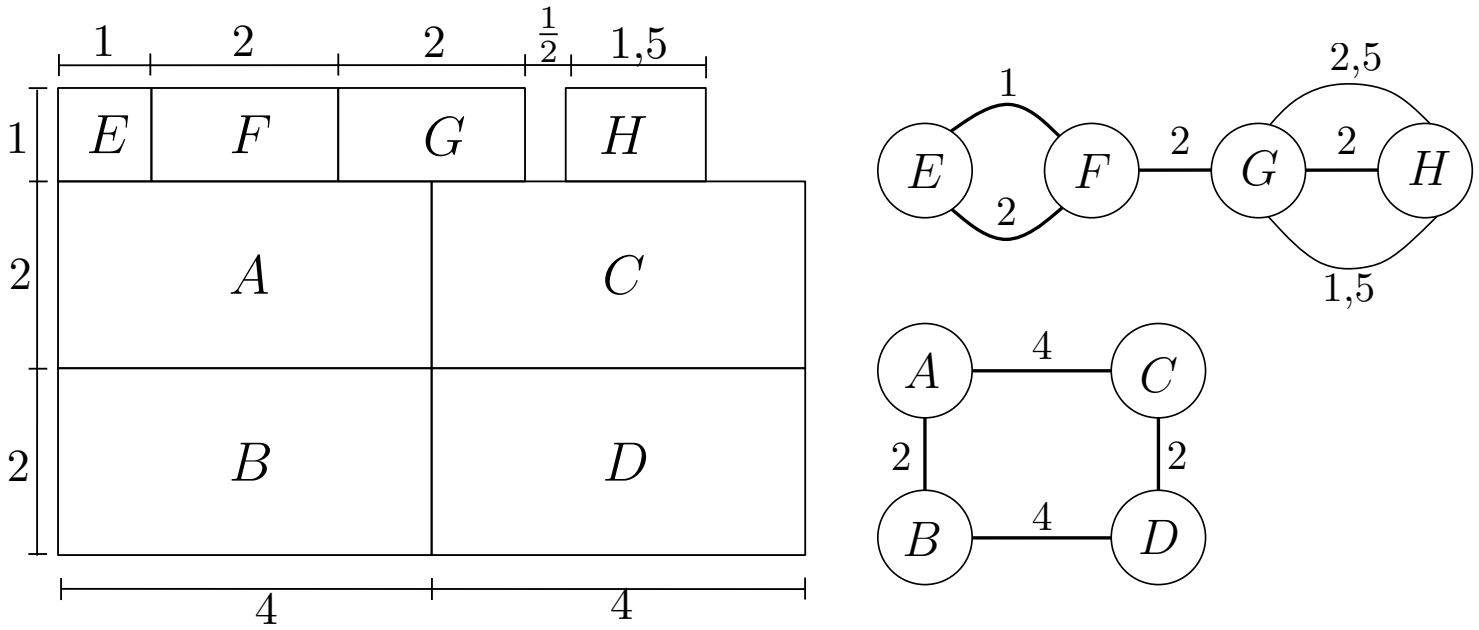
Algorithmus: Vorgehensweise





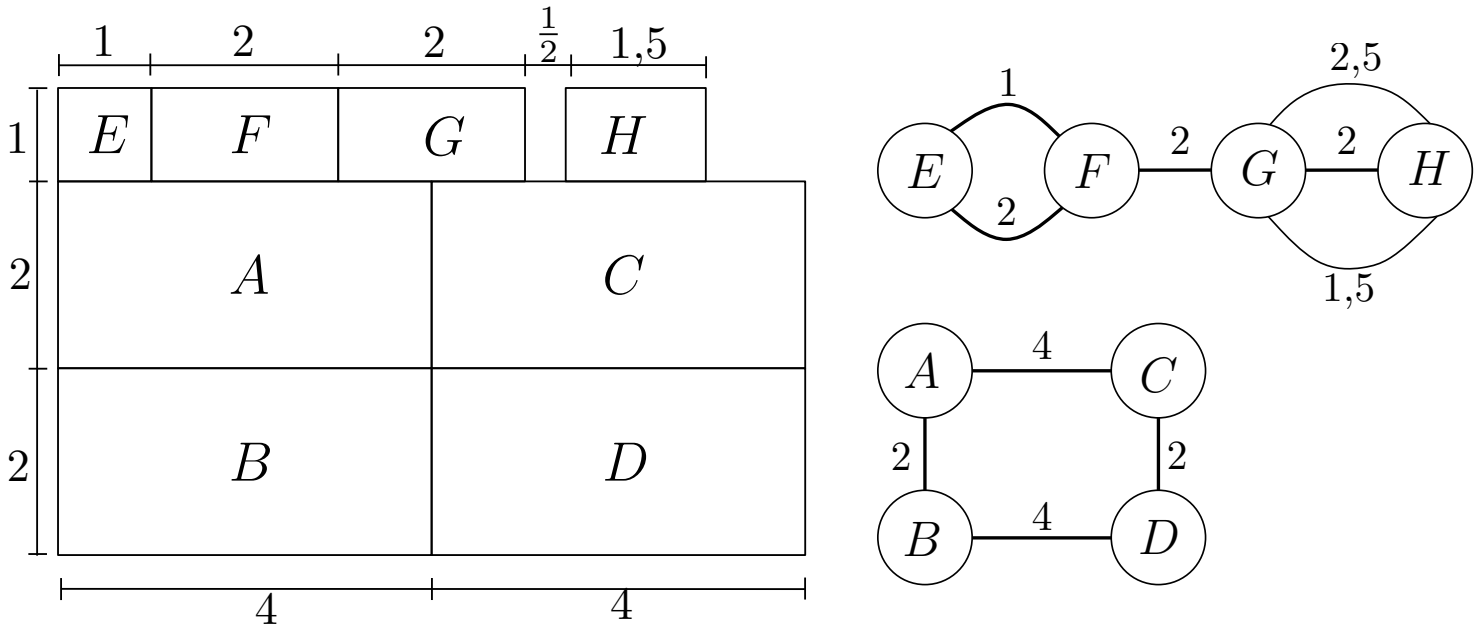
- Kanten verlaufen zwischen überdeckungsfreien Nachbarn
- Kantengewichte stehen für mögliche Distanzen
- Verschmelze iterativ Elemente
- Füge jeweils die Schnitte hinzu, die diese Verschmelzung umkehren

Auswahl der Elemente für Verschmelzung



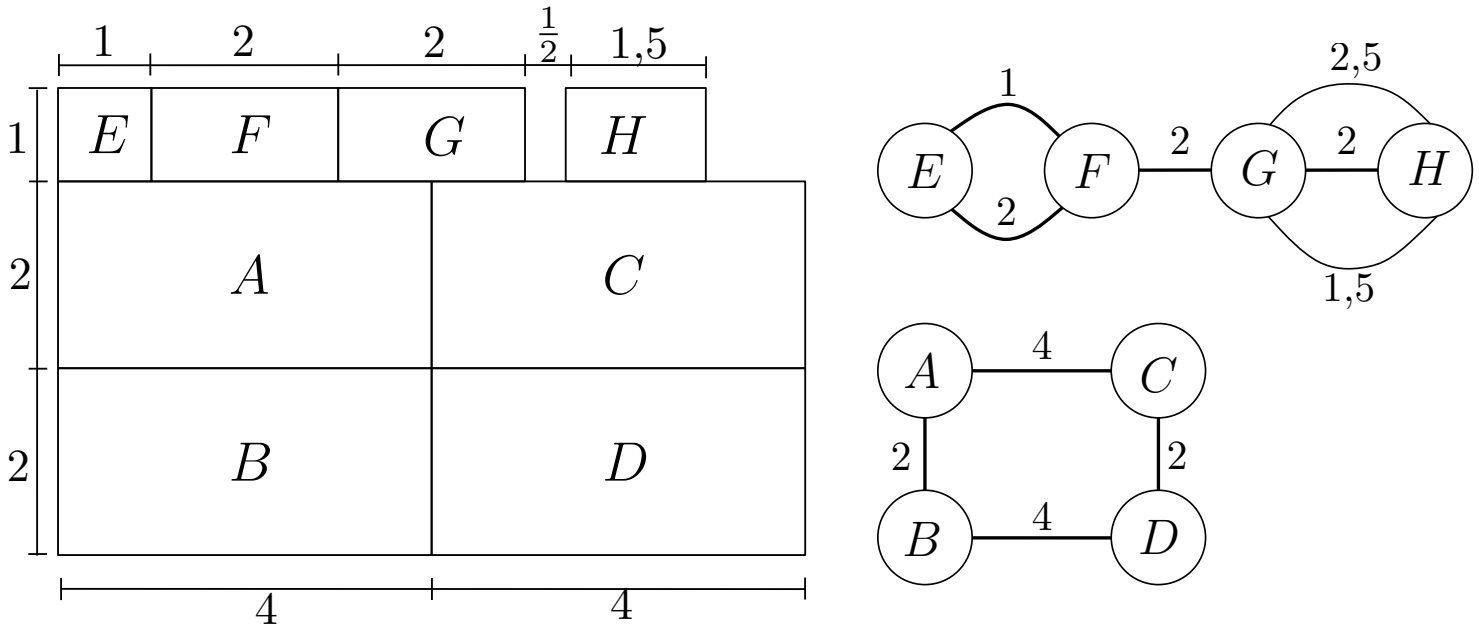
- Versuche Schnitte zu sparen, indem an Kanten mit gleichen Kantengewichten verschmolzen wird
- Dann haben die hinzugefügten Schnitte den gleichen Abstand
  → können gemeinsam geschnitten werden

Auswahl der Elemente für Verschmelzung



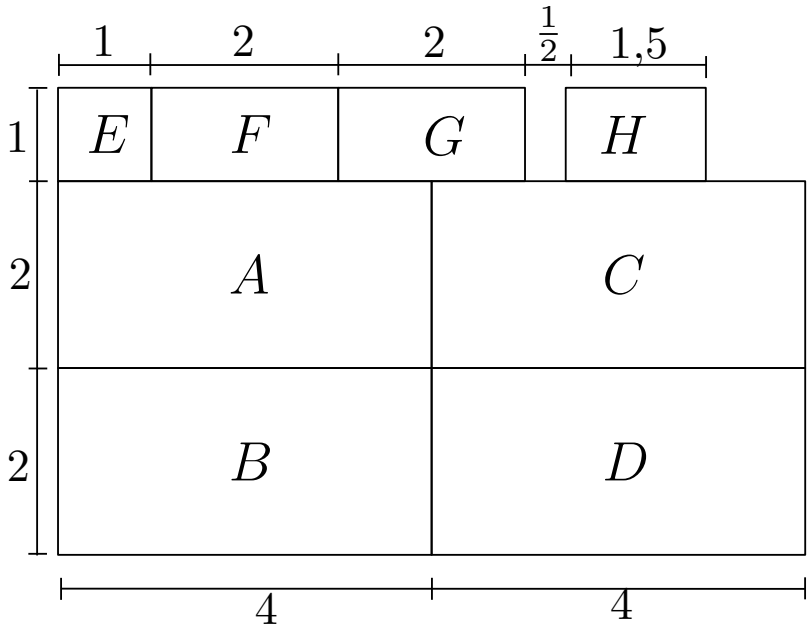
- Versuche Schnitte zu sparen, indem an Kanten mit gleichen Kantengewichten verschmolzen wird
- Dann haben die hinzugefügten Schnitte den gleichen Abstand $\rightarrow$ können gemeinsam geschnitten werden

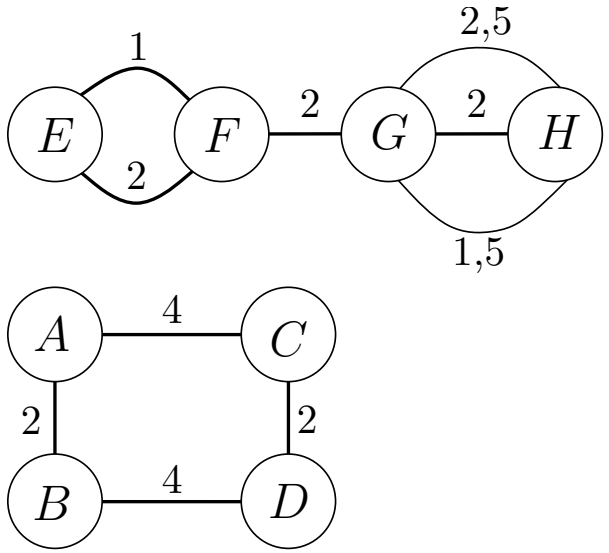
Auswahl der Elemente für Verschmelzung



- Versuche Schnitte zu sparen, indem an Kanten mit gleichen Kantengewichten verschmolzen wird
- Dann haben die hinzugefügten Schnitte den gleichen Abstand $\rightarrow$ können gemeinsam geschnitten werden

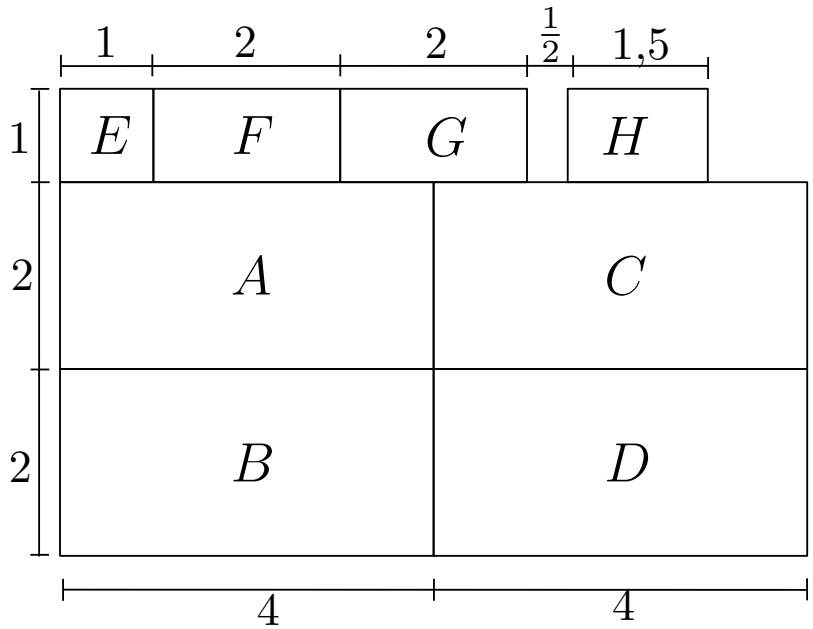
Iteration 1

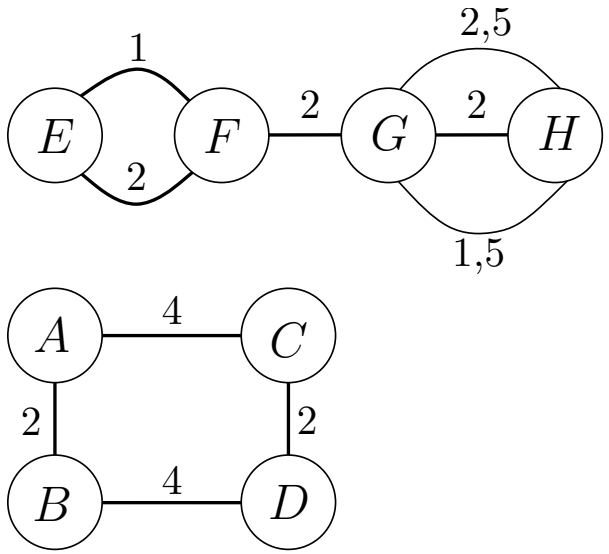




- Entferne alle Kanten, die nicht das häufigste Gewicht haben
- Suche ein größtmögliches Matching, verschmelze entsprechend

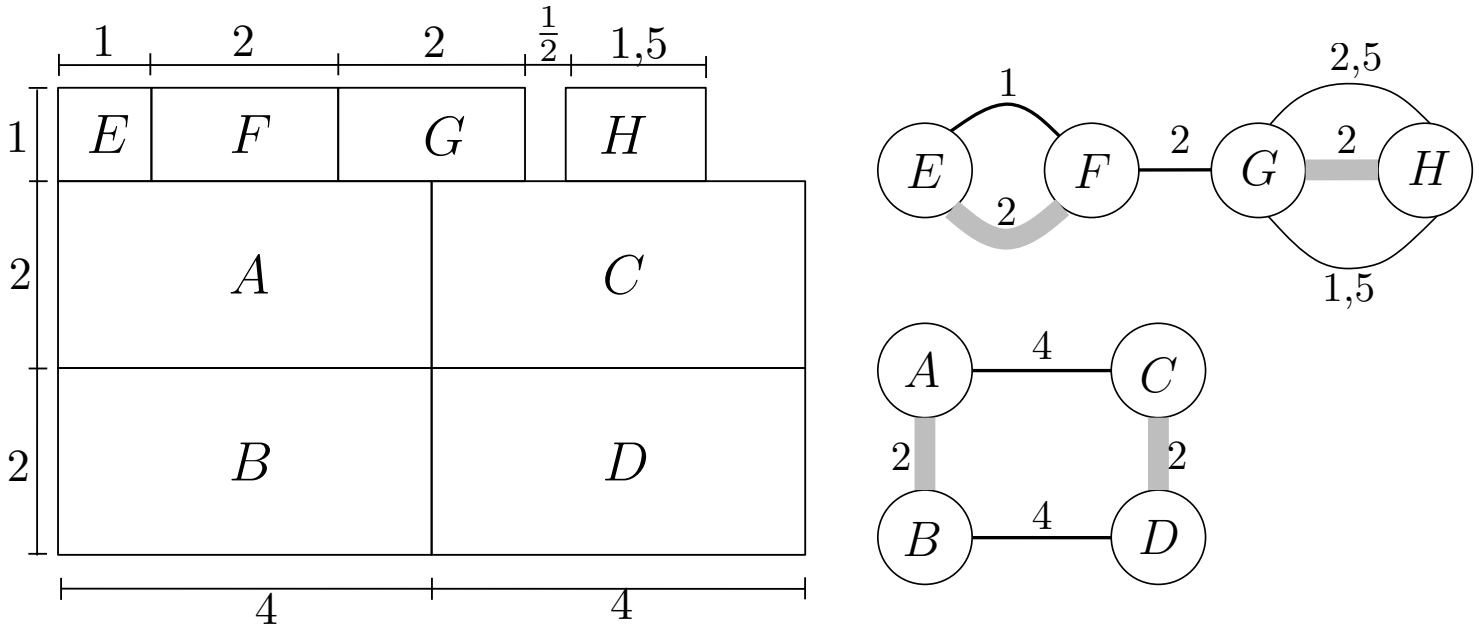
Iteration 1





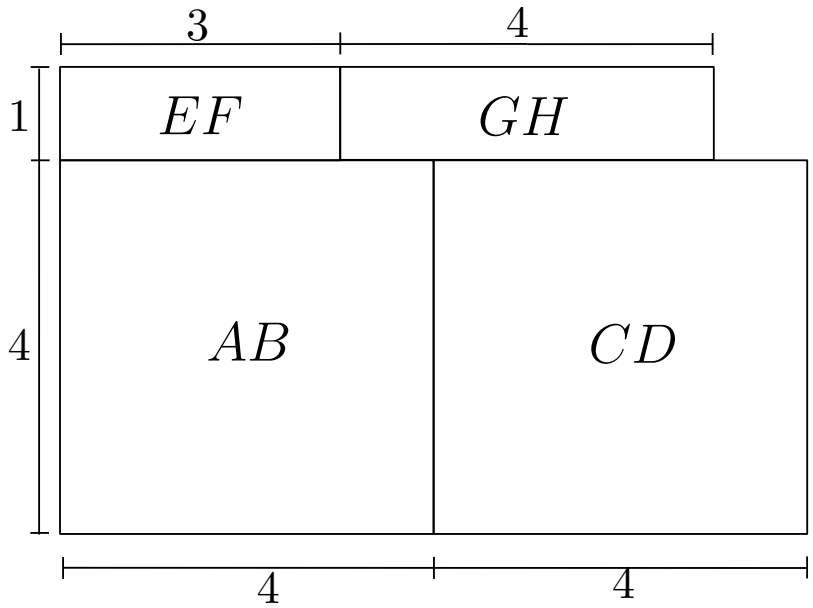
- Entferne alle Kanten, die nicht das häufigste Gewicht haben
- Suche ein größtmögliches Matching, verschmelze entsprechend

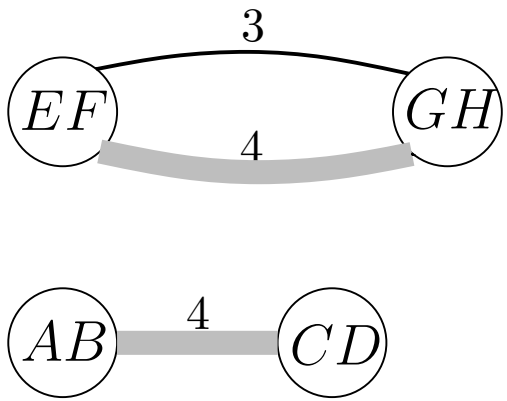
Iteration 1



- Entferne alle Kanten, die nicht das häufigste Gewicht haben
- Suche ein größtmögliches Matching, verschmelze entsprechend

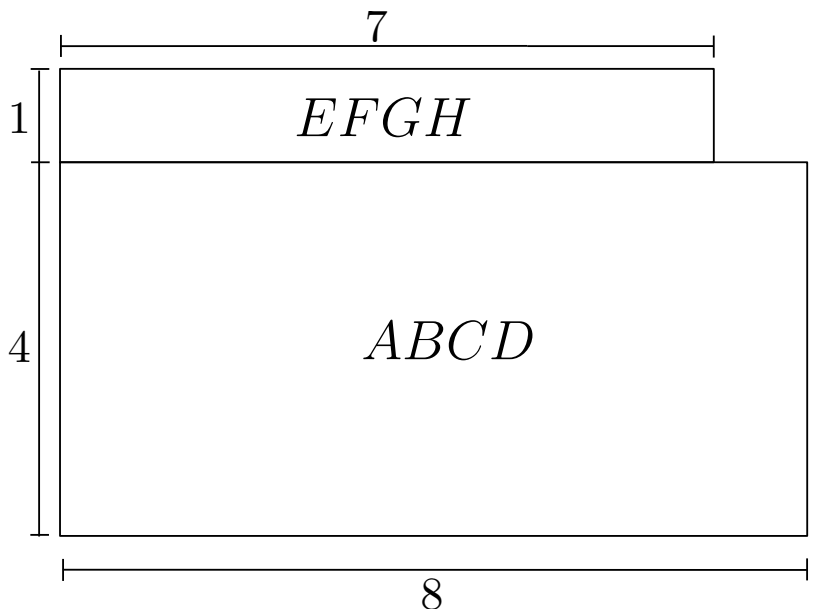
Iteration 2

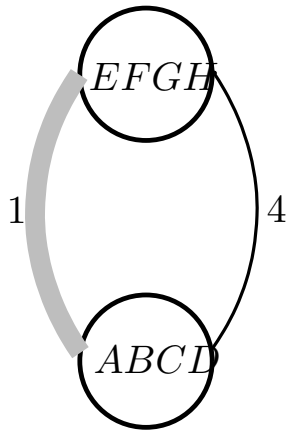




- Entferne alle Kanten, die nicht das häufigste Gewicht haben
- Suche ein größtmögliches Matching, verschmelze entsprechend

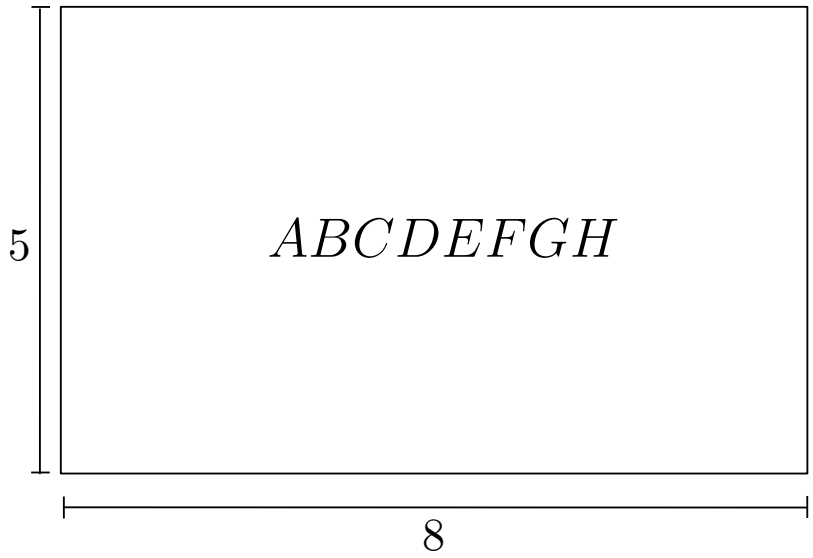
Iteration 3



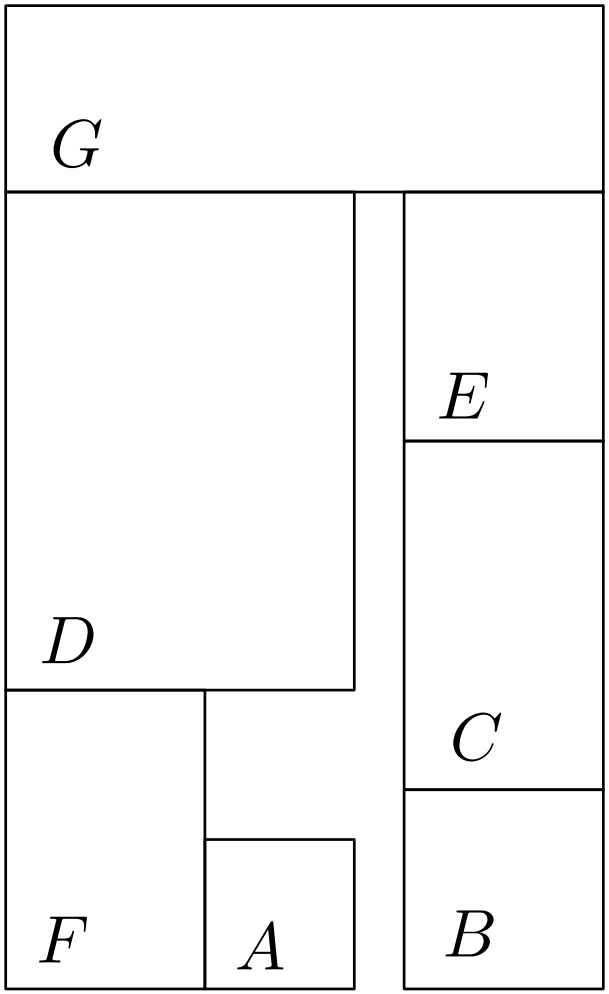


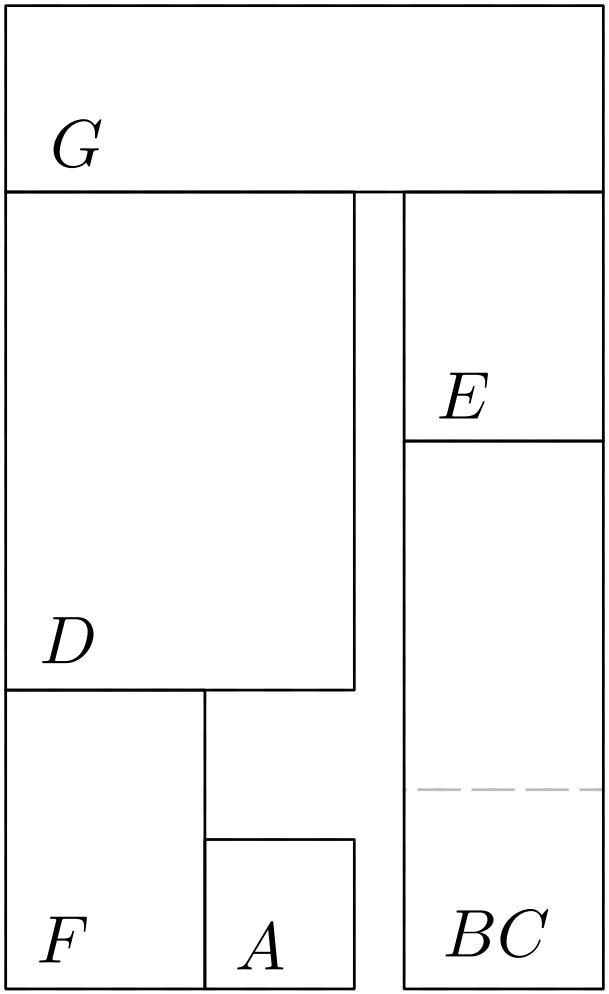
- Entferne alle Kanten, die nicht das häufigste Gewicht haben
- Suche ein größtmögliches Matching, verschmelze entsprechend

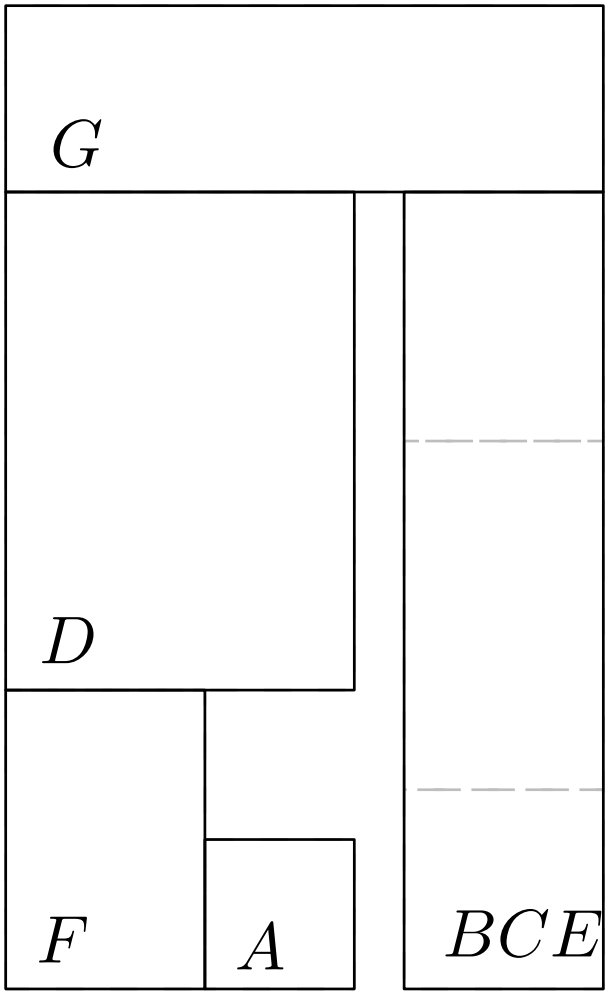
Iteration 4

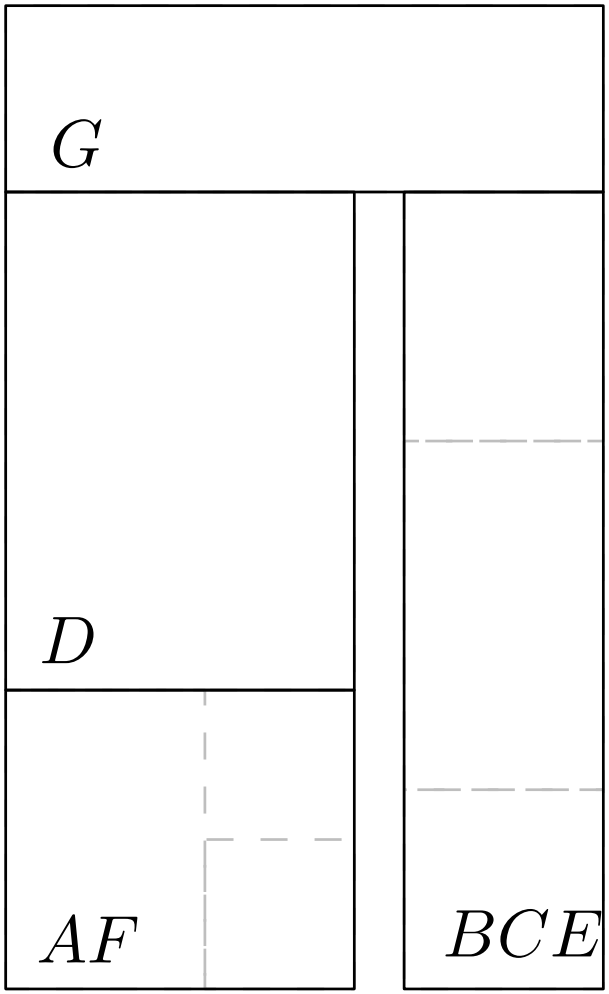


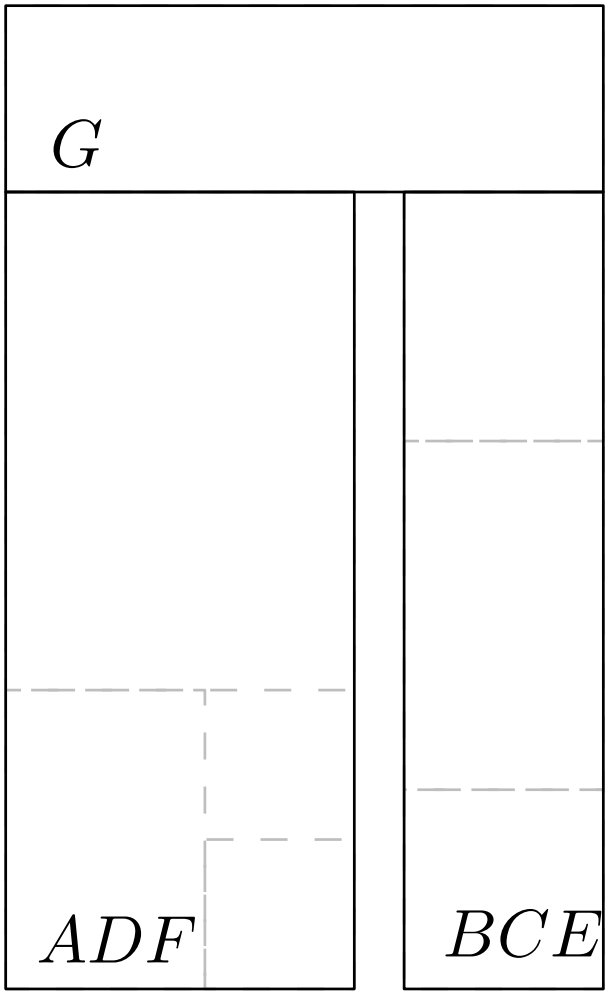
- Entferne alle Kanten, die nicht das häufigste Gewicht haben
- Suche ein größtmögliches Matching, verschmelze entsprechend

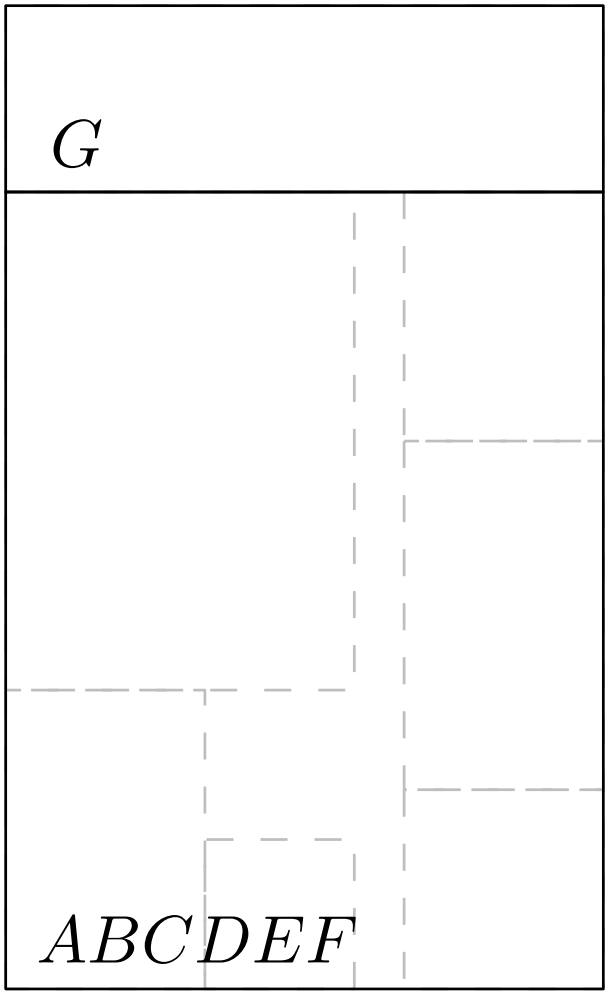

G
E
D
C
F
A
B

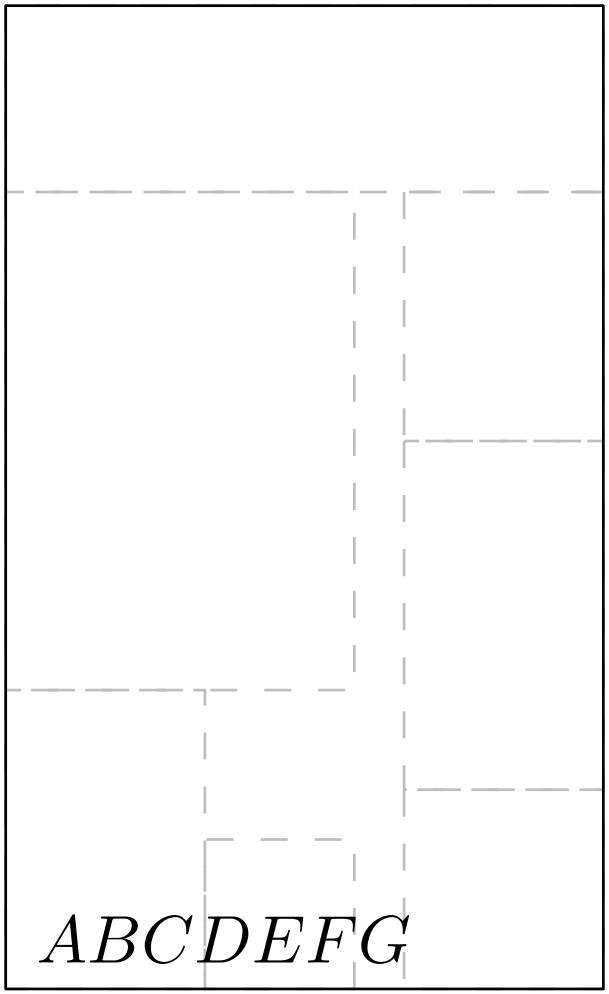

G
E
D
F
A
BC

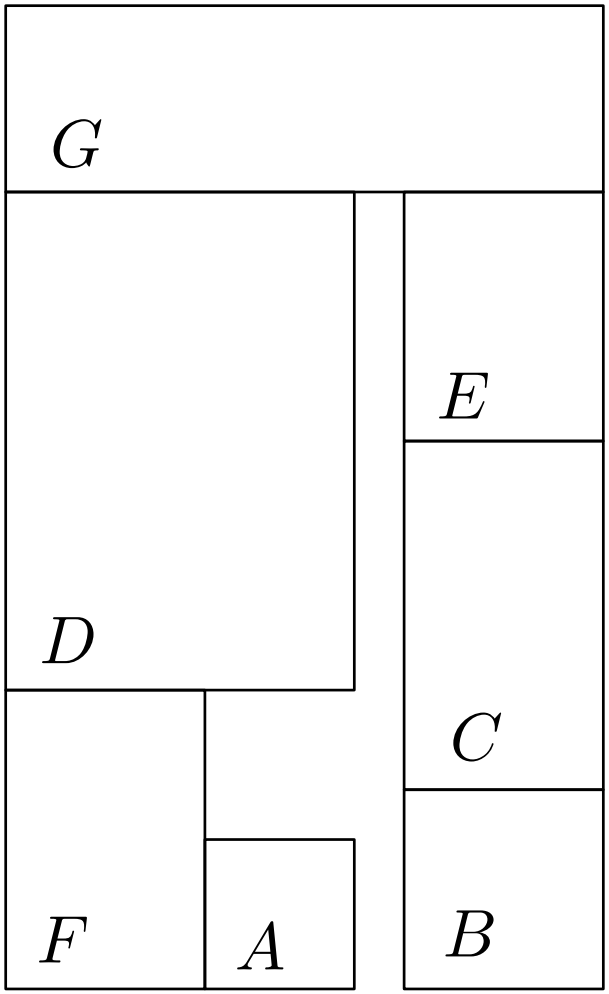

G
D
F
A
BCE

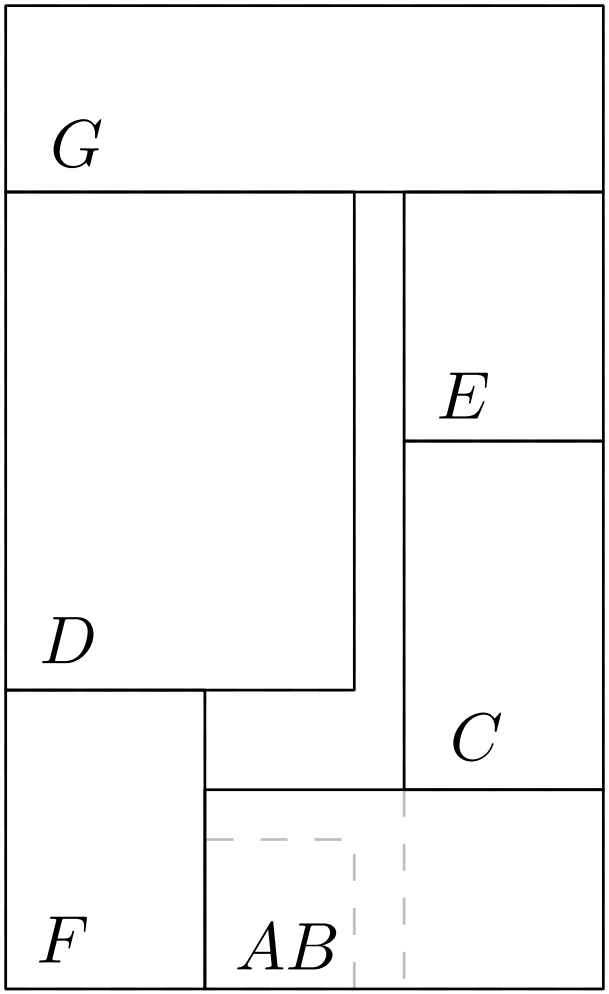

G
D
AF
BCE

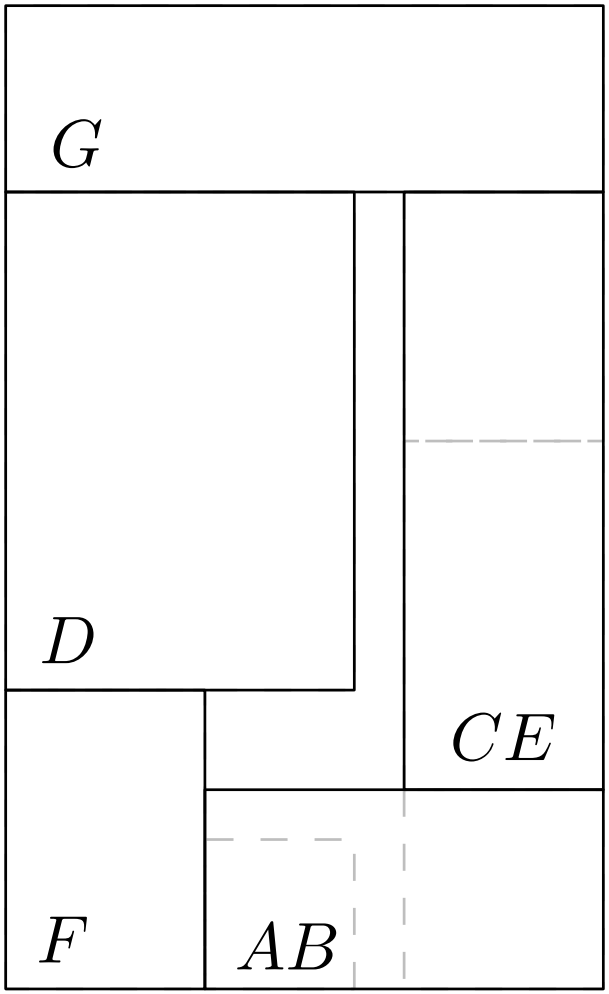

G
ADF
BCE

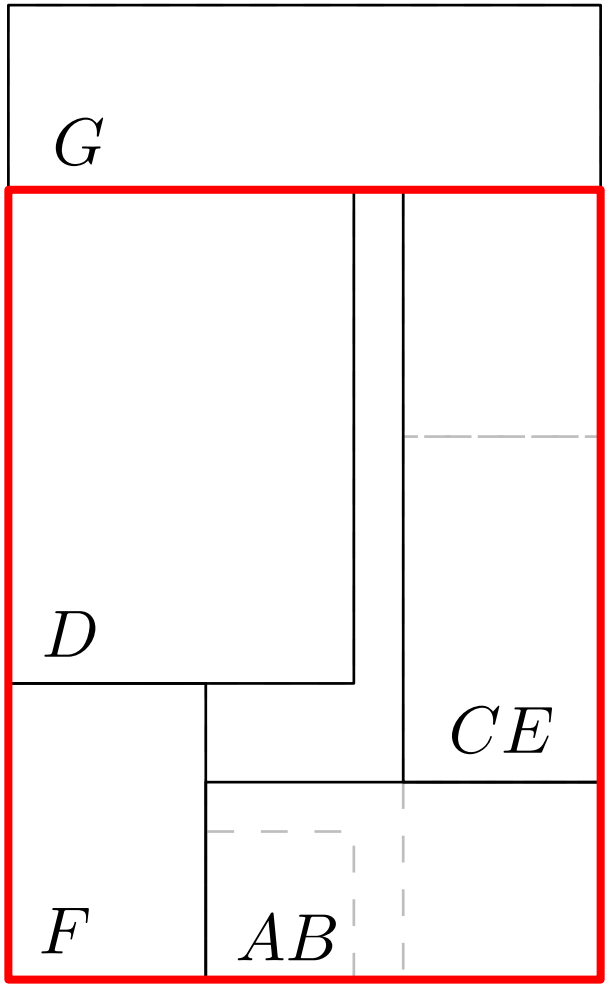

G
ABCDEF

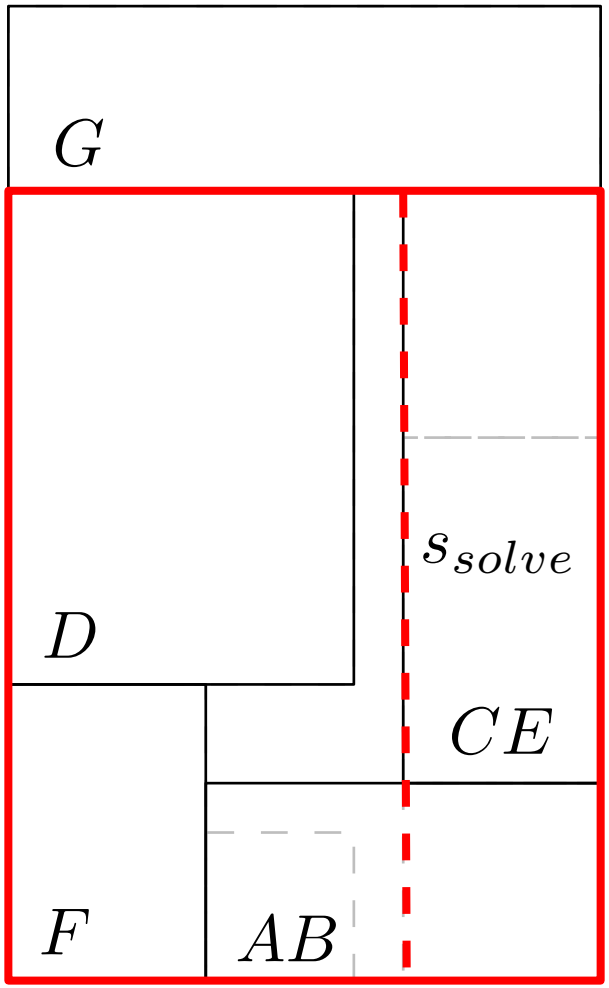

ABCDEFG

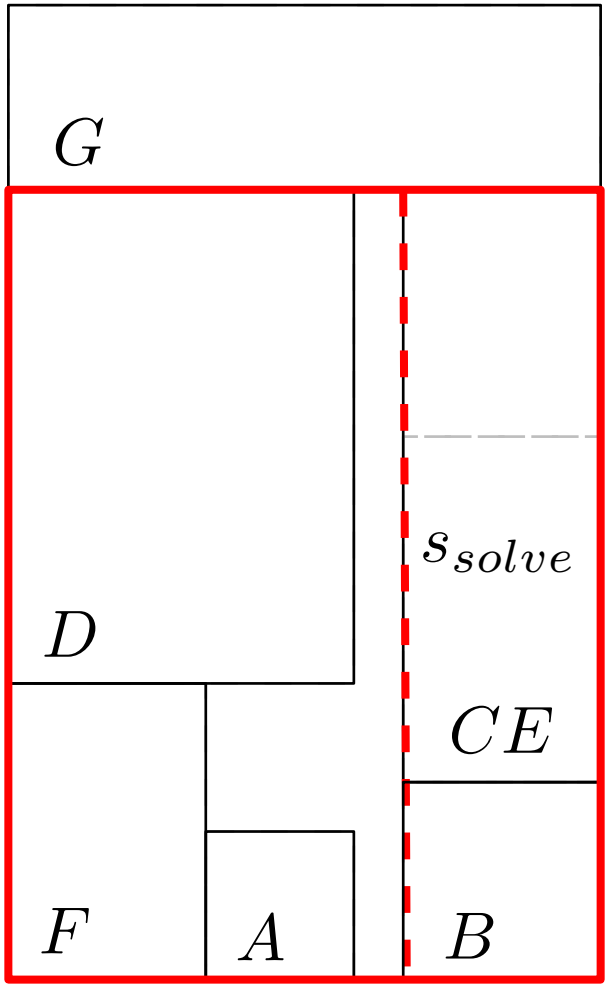

G
E
D
C
F
A
B


G
E
D
C
F
AB


G
D
CE
F
AB


G
D
CE
F
AB


G
D
s_{solve}
CE
F
AB


G
s_{solve}
D
CE
F
A
B

- Konflikte können immer aufgelöst werden
- Die Heuristik findet somit stets eine Lösung, wenn eine existiert

- Konflikte können immer aufgelöst werden
- Die Heuristik findet somit stets eine Lösung, wenn eine existiert
- Laufzeit Best-Case: $O(n^2)$ (n ist die Anzahl der Elemente)

- Konflikte können immer aufgelöst werden
- Die Heuristik findet somit stets eine Lösung, wenn eine existiert
- Laufzeit Best-Case: $O(n^2)$ (n ist die Anzahl der Elemente)
- Laufzeit Worst-Case: nicht bewiesen polynomiell, wegen der Anzahl der möglichen Konflikte!

- Konflikte können immer aufgelöst werden
- Die Heuristik findet somit stets eine Lösung, wenn eine existiert
- Laufzeit Best-Case: $O(n^2)$ (n ist die Anzahl der Elemente)
- Laufzeit Worst-Case: nicht bewiesen polynomiell, wegen der Anzahl der möglichen Konflikte!

Testergebnisse

- Implementierung unter Java
- Test auf realen Bögen aus der Praxis sowie künstlich erzeugten Bögen

Testergebnisse

- Implementierung unter Java
- Test auf realen Bögen aus der Praxis sowie künstlich erzeugten Bögen
- Künstliche Bögen bestehen aus gleichen, gitterartig angeordneten Rechtecken

Testergebnisse

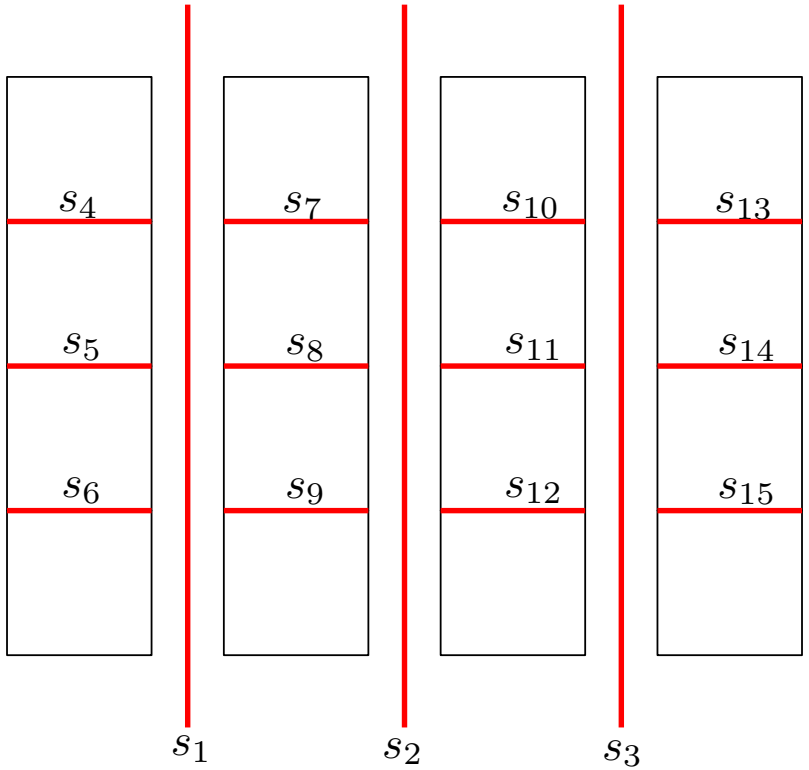
- Implementierung unter Java
- Test auf realen Bögen aus der Praxis sowie künstlich erzeugten Bögen
- Künstliche Bögen bestehen aus gleichen, gitterartig angeordneten Rechtecken
- Auch die künstlichen Bögen sind praxisrelevant

Testergebnisse

- Implementierung unter Java
- Test auf realen Bögen aus der Praxis sowie künstlich erzeugten Bögen
- Künstliche Bögen bestehen aus gleichen, gitterartig angeordneten Rechtecken
- Auch die künstlichen Bögen sind praxisrelevant

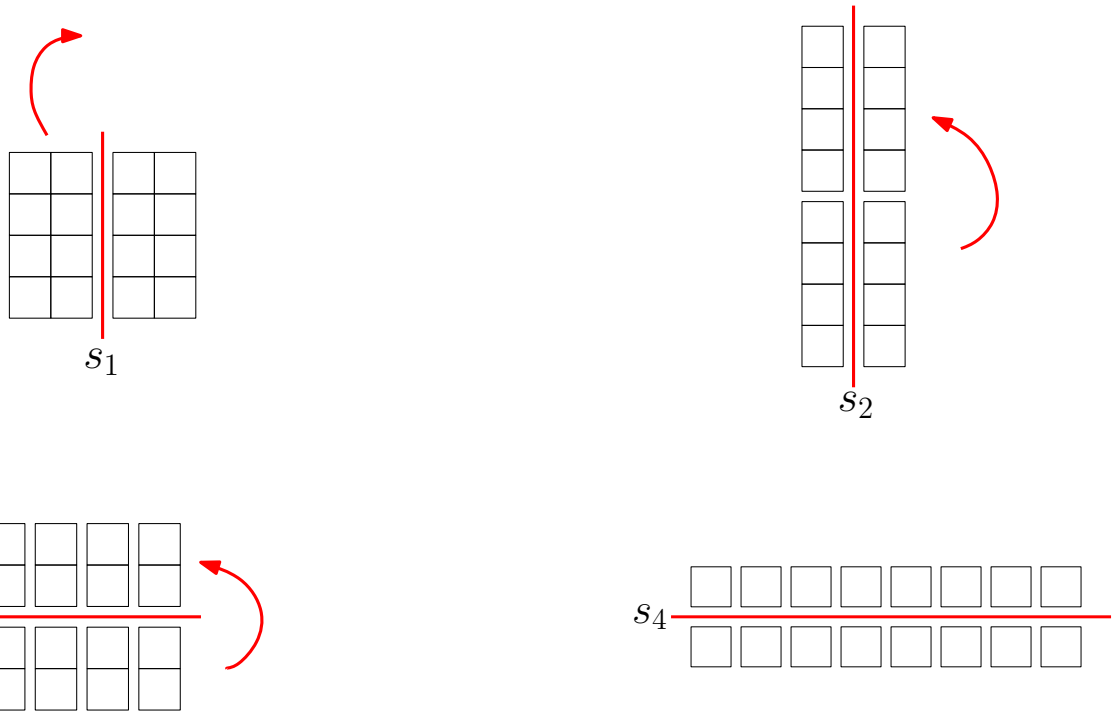
Naive Lösung

Für die gitterartigen Bögen benötigt eine naive Lösung $n - 1$ viele Schnitte.

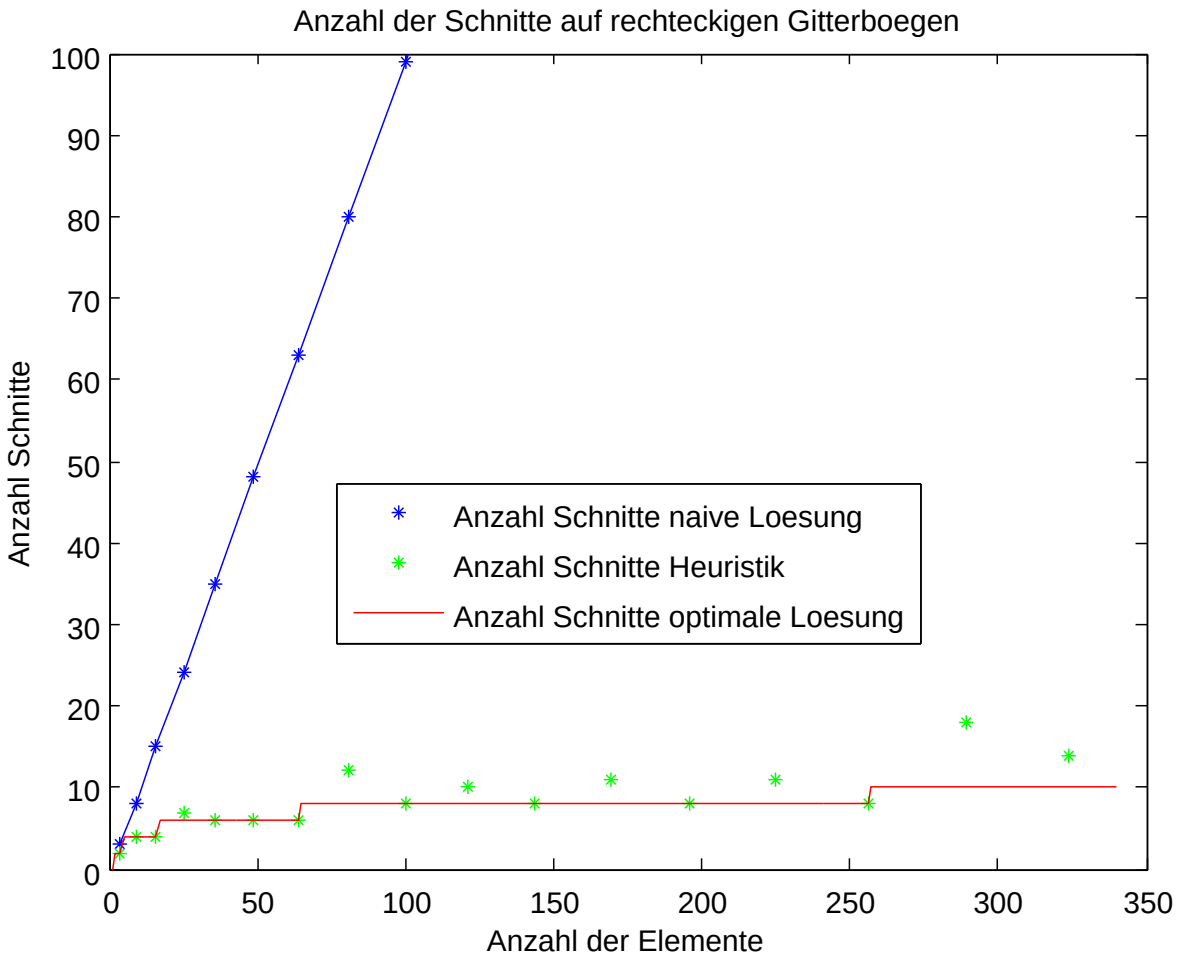


Optimale Lösung

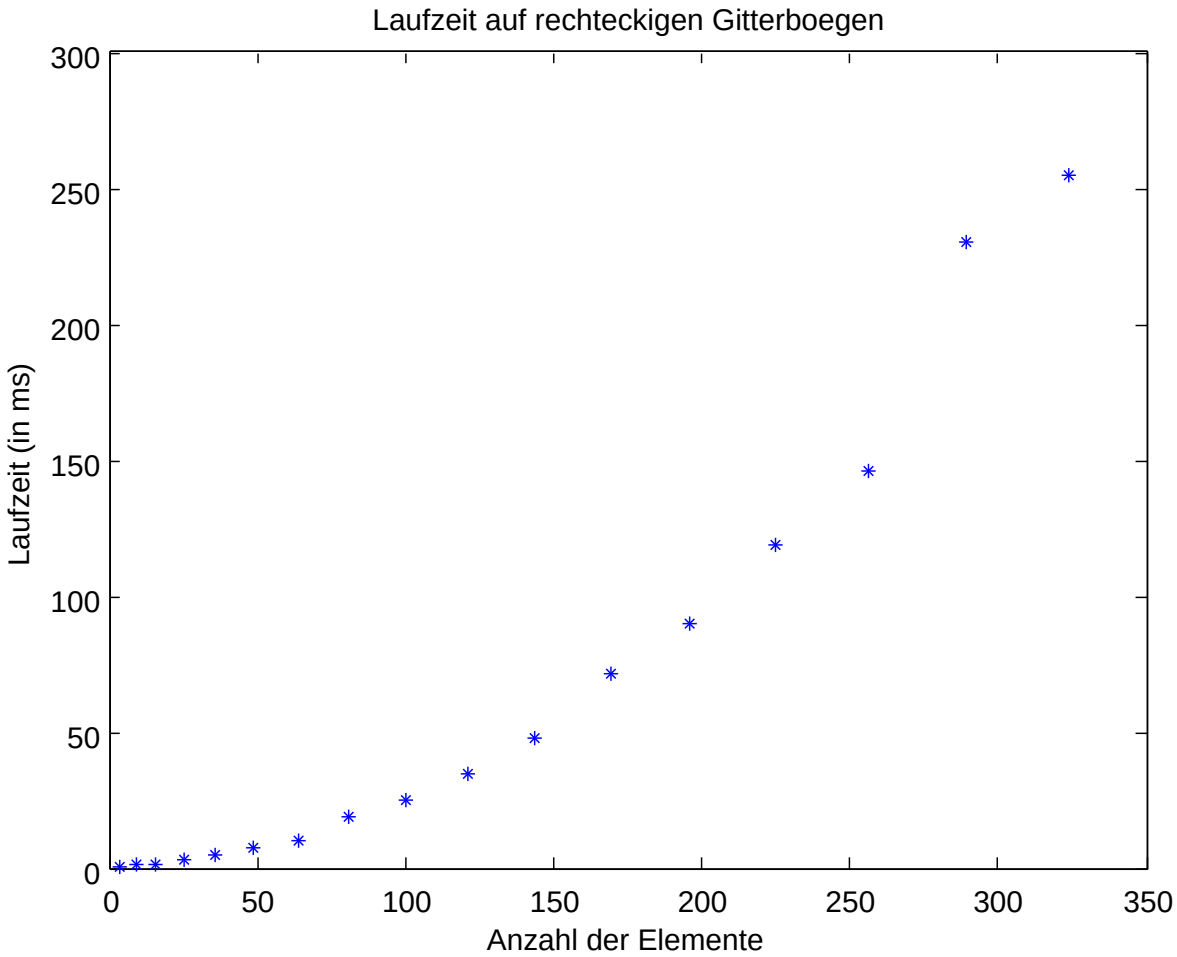
Eine optimale Lösung benötigt nur $2\lceil \log \sqrt{n} \rceil$ viele Schnitte.



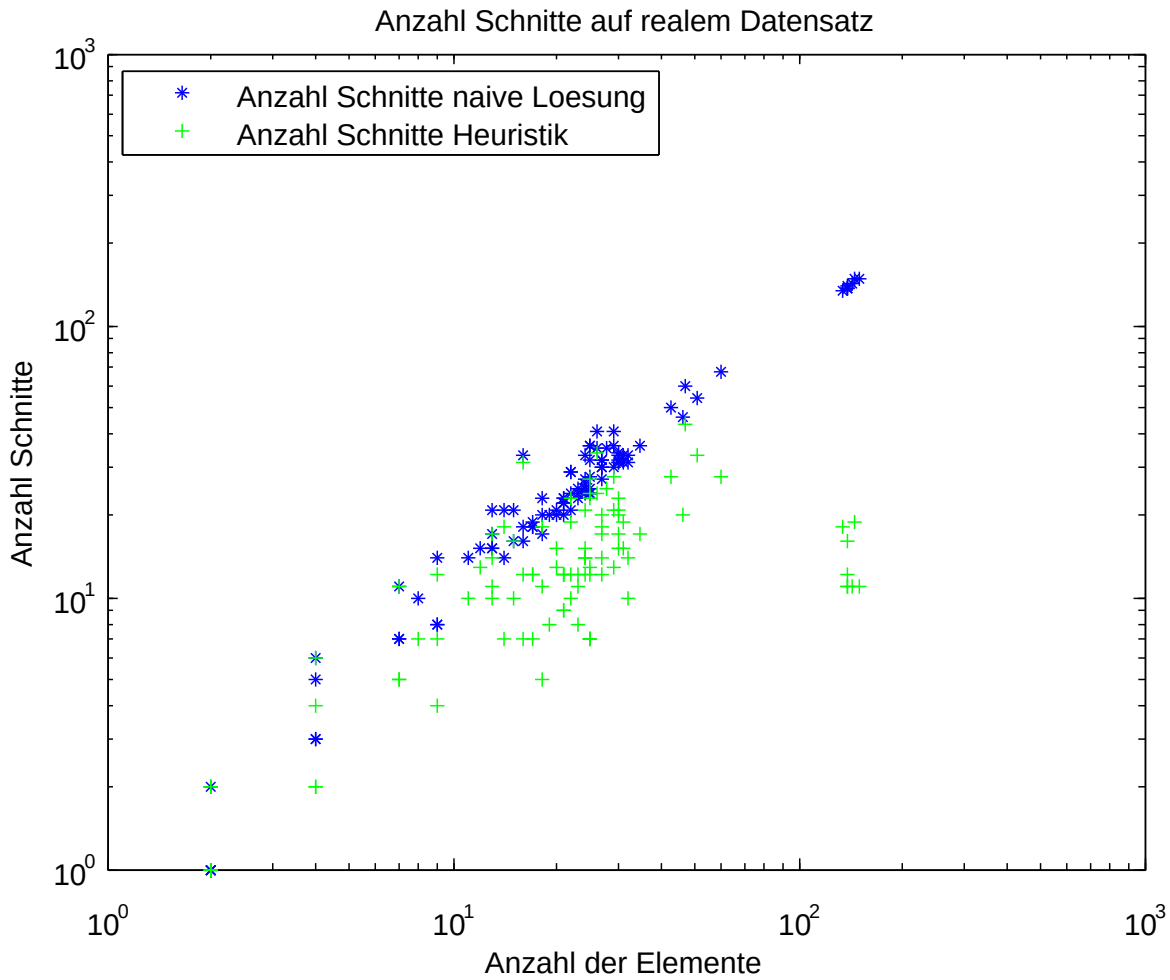
Ergebnisse künstlicher Datensatz



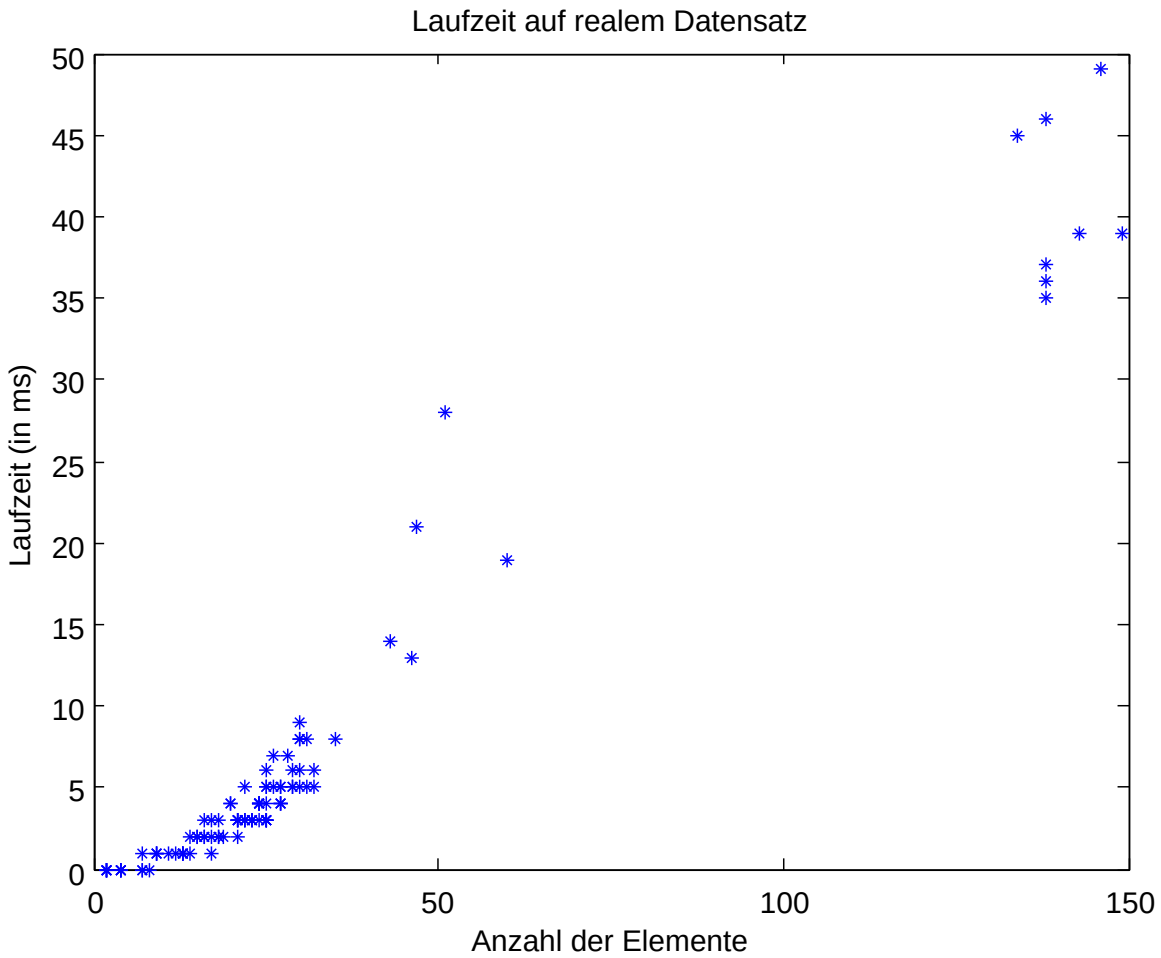
Laufzeiten künstlicher Datensatz



Ergebnisse realer Datensatz



Laufzeiten realer Datensatz



Fazit

- Heuristik bietet noch viele Verbesserungsmöglichkeiten
- Wie nützlich sind die Ergebnisse für die Praxis?

Fazit

- Heuristik bietet noch viele Verbesserungsmöglichkeiten
- Wie nützlich sind die Ergebnisse für die Praxis?
  - Guillotine ist nicht beliebig breit

Fazit

- Heuristik bietet noch viele Verbesserungsmöglichkeiten
- Wie nützlich sind die Ergebnisse für die Praxis?
  - Guillotine ist nicht beliebig breit
  - Beschleunigt das Einsparen von Schnitten überhaupt den Schneideprozess?

Fazit

- Heuristik bietet noch viele Verbesserungsmöglichkeiten
- Wie nützlich sind die Ergebnisse für die Praxis?
  - Guillotine ist nicht beliebig breit
  - Beschleunigt das Einsparen von Schnitten überhaupt den Schneideprozess?

Vielen Dank für ihre Aufmerksamkeit!