Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik Übung

6. Übungstermin · 18. Dezember 2018

Guido Brückner

Institut für Theoretische Informatik · Lehrstuhl Algorithmik

Übersicht

Organisatorisches

- → Unterschiedliche Verfahren in Vorlesung / Übung / Tutorien
- → Abgabetermine Übungsblätter
- → Punktegrenzen Notenbonus

Inhalt

- → NPund coNP
- → Pseudopolynomielle Algorithmen
- → Approximationsalgorithmen
- → Ganzzahlige Programme

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Unterschiedliche Verfahren

Teilweise werden in der Vorlesung / Übung mehrere Lösungsverfahren für dasselbe Problem vorgestellt.

- z.B. zur Konstruktion der Äquivalenzklassen

Eure Tutoren zeigen vielleicht auch noch weitere Verfahren.

Achtet darauf, welches Verfahren ihr benutzt!

In der Klausur zählen nur die Verfahren aus der Vorlesung und aus der Übung!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Abgabetermine Übungsblätter

- Ausgabe des 6. Übungsblattes: 8. Januar
- Abgabe des 5. Übungsblattes: 15. Januar
- Abgabe des 6. Übungsblattes: 22. Januar

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Punktegrenzen Notenbonus

Karlsruher Institut für Technologie

Notenbonus auf bestandene Klausur:

- insgesamt **52 Punkte** oder mehr $\Rightarrow$ **ein** Bonuspunkt
- insgesamt **105 Punkte** oder mehr $\Rightarrow$ **zwei** Bonuspunkte
- insgesamt **157 Punkte** oder mehr $\Rightarrow$ **drei** Bonuspunkte

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Übersicht

Organisatorisches

- Unterschiedliche Verfahren in Vorlesung / Übung / Tutorien
- Abgabetermine Übungsblätter
- Punktegrenzen Notenbonus

Inhalt

- NPund coNP
- Pseudopolynomielle Algorithmen
- Approximationsalgorithmen
- Ganzzahlige Programme

Aufgabe

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt NP = coNP.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt NP = coNP.

Wiederholung:

Die Klasse **NP** ist die Menge aller Sprachen L, für die es eine nichtdeterministische Turing-Maschine gibt, die L in polynomieller Zeit erkennt.

Die Klasse **coNP** ist die Menge aller Sprachen, deren Komplement in NP enthalten ist:

$$\text{coNP} = \{L \subseteq \Sigma^\star \mid \Sigma^\star \setminus L \in \text{NP}\}$$

Aufgabe

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt NP = coNP.

Sei Π_1 ein NP-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt NP = coNP.

Sei Π_1 ein NP-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in \mathrm{NP}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt NP = coNP.

Sei Π_1 ein NP-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in \mathrm{NP}$

Zeige: Komplement Π_2^c eines beliebigen Problems $\Pi_2 \in \mathrm{NP}$ liegt in NP.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt NP = coNP.

Sei Π_1 ein NP-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in \mathrm{NP}$

Zeige: Komplement Π_2^c eines beliebigen Problems $\Pi_2 \in \mathrm{NP}$ liegt in NP.

Π_1 ist NP-vollständig: Es gibt Reduktion φ von Π_2 auf Π_1

φ ist auch Reduktion von Π_2^c auf Π_1^c

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt NP = coNP.

Sei Π_1 ein NP-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in \mathrm{NP}$

Zeige: Komplement Π_2^c eines beliebigen Problems $\Pi_2 \in \mathrm{NP}$ liegt in NP.

Π_1 ist NP-vollständig: Es gibt Reduktion φ von Π_2 auf Π_1

φ ist auch Reduktion von Π_2^c auf Π_1^c

Sei I_2^c eine Instanz von Π_2^c.

Transformiere die Instanz in eine Instanz von Π_1^c: $I_1^c = \varphi(I_2^c)$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt NP = coNP.

Sei Π_1 ein NP-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in$ NP

Zeige: Komplement Π_2^c eines beliebigen Problems $\Pi_2 \in$ NP liegt in NP.

Π_1 ist NP-vollständig: Es gibt Reduktion φ von Π_2 auf Π_1

φ ist auch Reduktion von Π_2^c auf Π_1^c

Sei I_2^c eine Instanz von Π_2^c.

Transformiere die Instanz in eine Instanz von Π_1^c: $I_1^c = \varphi(I_2^c)$.

Da Π_1^c in NPliegt, kann eine NTM $\mathcal{M}$ wie folgt auf I_1^c arbeiten.

1. $\mathcal{M}$ berechnet eine Lösung von I_1^c
2. $\mathcal{M}$ überprüft ob Ja-Instanz
3. Falls ja, dann auch I_2^c sonst ist I_2^c keine Ja-Instanz.

Pseudopolynomielle Algorithmen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wiederholung

Ein Algorithmus, der Problem Π löst, heißt pseudopolynomiell, falls seine Laufzeit durch ein Polynom der beiden Variablen

- Eingabegröße und
- Größe der größten in der Eingabe vorkommenden Zahl

beschränkt ist.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wiederholung

Karlsruher Institut für Technologie

Ein Algorithmus, der Problem Π löst, heißt pseudopolynomiell, falls seine Laufzeit durch ein Polynom der beiden Variablen

- Eingabegröße und
- Größe der größten in der Eingabe vorkommenden Zahl

beschränkt ist.

Äquivalent:

Ein Algorithmus, der Problem Π löst, heißt pseudopolynomiell, falls seine Laufzeit polynomiell durch die Größe der Eingabe **bei unärer Kodierung** beschränkt ist.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem SUBSETSUM:
Gegeben: Menge $M = \{x_1, \dots, x_n\}$, Funktion $w\colon M \to \mathbb{N}_0$ und $K \in \mathbb{N}$.
Frage: Existiert Teilmenge $M' \subseteq M$ mit

$$\sum_{a \in \mathcal{M}'} w(a) = K$$

Aufgabe: Geben Sie einen pseudopolynomiellen Algorithmus an.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem SUBSETSUM:
Gegeben: Menge $M = \{x_1, \dots, x_n\}$, Funktion $w \colon M \to \mathbb{N}_0$ und $K \in \mathbb{N}$.
Frage: Existiert Teilmenge $M' \subseteq M$ mit

$$\sum_{a \in \mathcal{M}'} w(a) = K$$

Aufgabe: Geben Sie einen pseudopolynomiellen Algorithmus an.

Idee Verwende 2-dimensionale Tabelle T der Größe $(n+1) \times (K+1)$, die Wahrheitswerte enthält

	0	1	2	3	4	5
	w					
x_1			f			
x_2						
x_3						

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem SUBSETSUM:
Gegeben: Menge $M = \{x_1, \ldots, x_n\}$, Funktion $w\colon M \to \mathbb{N}_0$ und $K \in \mathbb{N}$.
Frage: Existiert Teilmenge $M' \subseteq M$ mit

$$\sum_{a \in \mathcal{M}'} w(a) = K$$

Aufgabe: Geben Sie einen pseudopolynomiellen Algorithmus an.

Idee Verwende 2-dimensionale Tabelle T der Größe $(n+1) \times (K+1)$, die Wahrheitswerte enthält

Interpretation: $T[i,j] = w \quad \Leftrightarrow$

Es gibt Teilmenge $M' \subseteq \{x_1, \cdots, x_i\}$, sodass

$$\sum_{a \in \mathcal{M}'} w(a) = j$$

	0	1	2	3	4	5
	w					
x_1			f			
x_2						
x_3						

Lösung

Interpretation: $T[i,j] = w \quad \Leftrightarrow$

Es gibt Teilmenge $M' \subseteq \{x_1, \cdots, x_i\}$, sodass

$$\sum_{a \in \mathcal{M}'} w(a) = j$$

	0	1	2	3	4	5
	w					
x_1			f			
x_2						
x_3						

Wie $T[i,j]$ aus vorherigen Einträgen bestimmen?

$$T[i,j] = T[i-1,j] \vee T[i-1, j-x_i]$$

Lösung

Interpretation: $T[i,j] = w \quad \Leftrightarrow$

Es gibt Teilmenge $M' \subseteq \{x_1, \cdots, x_i\}$, sodass

$$\sum_{a \in \mathcal{M}'} w(a) = j$$

	0	1	2	3	4	5
	w					
x_1			f			
x_2						
x_3						

Wie $T[i,j]$ aus vorherigen Einträgen bestimmen?

$$T[i,j] = T[i-1,j] \vee T[i-1,j-x_i]$$

Algorithmus

Initialisiere $T[0,0] = \mathbf{w}$ und $T[i,j] = \mathbf{f}$ sonst

Für $i = 0, \ldots n$

  Für $j = 0, \ldots K$

    $T[i,j] = T[i-1,j] \vee T[i-1,j-x_i]$

Gebe $T[n,K]$ zurück.

Beispiel

$M = \{x_1, x_2, x_3, x_4, x_5\}$

$w(x_1) = 2$ $w(x_2) = 3$ $w(x_3) = 5$ $w(x_4) = 7$ $w(x_5) = 10$

$K = 14$

	0	1	2	3	4	5									14
x_1															
x_2															
x_3															
x_4															
x_5															

$$T[i, j] = T[i-1, j] \vee T[i-1, j-x_i]$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

$M = \{x_1, x_2, x_3, x_4, x_5\}$

$w(x_1) = 2$ $w(x_2) = 3$ $w(x_3) = 5$ $w(x_4) = 7$ $w(x_5) = 10$

$K = 14$

	0	1	2	3	4	5									14
	w	f	f	f	f	f	f	f	f	f	f	f	f	f	f
x_1															
x_2															
x_3															
x_4															
x_5															

$$T[i, j] = T[i-1, j] \vee T[i-1, j-x_i]$$

Beispiel

$M = \{x_1, x_2, x_3, x_4, x_5\}$

$w(x_1) = 2$ $w(x_2) = 3$ $w(x_3) = 5$ $w(x_4) = 7$ $w(x_5) = 10$

$K = 14$

	0	1	2	3	4	5									14
	w	f	f	f	f	f	f	f	f	f	f	f	f	f	f
x_1	w	f	w	f	f	f	f	f	f	f	f	f	f	f	f
x_2															
x_3															
x_4															
x_5															

$$T[i,j] = T[i-1,j] \vee T[i-1, j-x_i]$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

$M = \{x_1, x_2, x_3, x_4, x_5\}$

$w(x_1) = 2 \quad w(x_2) = 3 \quad w(x_3) = 5 \quad w(x_4) = 7 \quad w(x_5) = 10$

$K = 14$

	0	1	2	3	4	5									14
	w	f	f	f	f	f	f	f	f	f	f	f	f	f	f
x_1	w	f	w	f	f	f	f	f	f	f	f	f	f	f	f
x_2	w	f	w	w	f	w	f	f	f	f	f	f	f	f	f
x_3															
x_4															
x_5															

$$T[i, j] = T[i-1, j] \vee T[i-1, j-x_i]$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

$M = \{x_1, x_2, x_3, x_4, x_5\}$

$w(x_1) = 2 \quad w(x_2) = 3 \quad w(x_3) = 5 \quad w(x_4) = 7 \quad w(x_5) = 10$

$K = 14$

	0	1	2	3	4	5									14
	w	f	f	f	f	f	f	f	f	f	f	f	f	f	f
x_1	w	f	w	f	f	f	f	f	f	f	f	f	f	f	f
x_2	w	f	w	w	f	w	f	f	f	f	f	f	f	f	f
x_3	w	f	w	w	f	w	f	w	w	f	w	f	f	f	f
x_4															
x_5															

$$T[i, j] = T[i-1, j] \vee T[i-1, j-x_i]$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beispiel

$M = \{x_1, x_2, x_3, x_4, x_5\}$

$w(x_1) = 2 \qquad w(x_2) = 3 \qquad w(x_3) = 5 \qquad w(x_4) = 7 \qquad w(x_5) = 10$

$K = 14$

	0	1	2	3	4	5									14
	w	f	f	f	f	f	f	f	f	f	f	f	f	f	f
x_1	w	f	w	f	f	f	f	f	f	f	f	f	f	f	f
x_2	w	f	w	w	f	w	f	f	f	f	f	f	f	f	f
x_3	w	f	w	w	f	w	f	w	w	f	w	f	f	f	f
x_4	w	f	w	w	f	w	f	w	w	w	w	f	w	f	w
x_5															

$$T[i,j] = T[i-1,j] \vee T[i-1,j-x_i]$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Approximationsalgorithmen

Wiederholung

Sei Π ein Optimierungsproblem. Ein polynomialer Algorithmus $\mathcal{A}$, der für jedes $I \in D_\Pi$ einen Wert $\mathcal{A}(I)$ liefert, mit

$$|\mathrm{OPT}(I) - \mathcal{A}(I)| \leq K$$

und $K \in \mathbb{N}_0$ konstant, heißt Approximationsalgorithmus mit absoluter Gütegarantie oder absoluter Approximationsalgorithmus.

Aufgabe

Problem CLIQUE:
Gegeben: Ungerichteter Graph $G = (V, E)$.
Gesucht: Möglichst große Clique von G.
Hinweis: $C \subseteq V$ heißt *Clique*, falls für jedes Paar $u, v \in C$ die Kante $\{u, v\} \in E$ existiert.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem $\textsc{Clique}$**:**
Gegeben: Ungerichteter Graph $G = (V, E)$.
Gesucht: Möglichst große Clique von G.
Hinweis: $C \subseteq V$ heißt *Clique*, falls für jedes Paar $u, v \in C$ die Kante $\{u, v\} \in E$ existiert.

Zeige: Es gibt keinen Approximationsalgorithmus.

- **Annahme:** Sei $\mathcal{A}$ absoluter Approxalgo mit $|OPT(I) - \mathcal{A}(I)| \leq K \quad \forall I$
- **Idee:** Konstruiere aus $\mathcal{A}$ polynomiellen exakten Algo für $\textsc{Clique}$
- **Technik:** Nutze Instanz I zum Bau von I', sodass Lösung in I' eine „zu große" Lösung in I induziert.

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Graph G^m ist für $m \in \mathbb{N}$ wie folgt definiert:

1. Kopiere G m-mal
2. Verbinden jeden Knoten einer Kopie mit jedem Knoten aller anderen Kopien

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Graph G^m ist für $m \in \mathbb{N}$ wie folgt definiert:

1. Kopiere G m-mal
2. Verbinden jeden Knoten einer Kopie mit jedem Knoten aller anderen Kopien

Beob.: $\exists$ Clique C der Größe α in G *gdw.* $\exists$ Clique C^m der Größe αm in G^m.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Graph G^m ist für $m \in \mathbb{N}$ wie folgt definiert:

1. Kopiere G m-mal
2. Verbinden jeden Knoten einer Kopie mit jedem Knoten aller anderen Kopien

Beob.: $\exists$ Clique C der Größe α in G *gdw.* $\exists$ Clique C^m der Größe αm in G^m.

Annahme: Es gibt Approximationalg. $\mathcal{A}$ mit $|\mathcal{A}(I) - \mathrm{OPT}(I)| \leq K$ (für alle I)

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Graph G^m ist für $m \in \mathbb{N}$ wie folgt definiert:

1. Kopiere G m-mal
2. Verbinden jeden Knoten einer Kopie mit jedem Knoten aller anderen Kopien

Beob.: $\exists$ Clique C der Größe α in G *gdw.* $\exists$ Clique C^m der Größe αm in G^m.

Annahme: Es gibt Approximationalg. $\mathcal{A}$ mit $|\mathcal{A}(I) - \text{OPT}(I)| \leq K$ (für alle I)

Strategie um größte Clique in G zu finden:

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Graph G^m ist für $m \in \mathbb{N}$ wie folgt definiert:

1. Kopiere G m-mal
2. Verbinden jeden Knoten einer Kopie mit jedem Knoten aller anderen Kopien

Beob.: $\exists$ Clique C der Größe α in G *gdw.* $\exists$ Clique C^m der Größe αm in G^m.

Annahme: Es gibt Approximationalg. $\mathcal{A}$ mit $|\mathcal{A}(I) - \mathrm{OPT}(I)| \leq K$ (für alle I)

Strategie um größte Clique in G zu finden:

1. Erstelle G^{K+1} von G
2. Wende $\mathcal{A}$ auf G^{K+1} an: liefert Clique C^{K+1}
3. Extrahiere aus C^{K+1} eine Clique C für G der Größe $(K + 1) \cdot |C| = |C^{K+1}|$

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Graph G^m ist für $m \in \mathbb{N}$ wie folgt definiert:

1. Kopiere G m-mal
2. Verbinden jeden Knoten einer Kopie mit jedem Knoten aller anderen Kopien

Beob.: $\exists$ Clique C der Größe α in G *gdw.* $\exists$ Clique C^m der Größe αm in G^m.

Annahme: Es gibt Approximationalg. $\mathcal{A}$ mit $|\mathcal{A}(I) - \mathrm{OPT}(I)| \leq K$ (für alle I)

Strategie um größte Clique in G zu finden:

1. Erstelle G^{K+1} von G
2. Wende $\mathcal{A}$ auf G^{K+1} an: liefert Clique C^{K+1}
3. Extrahiere aus C^{K+1} eine Clique C für G der Größe $(K+1) \cdot |C| = |C^{K+1}|$

$$|\mathcal{A}(G^{K+1}) - \mathrm{OPT}(G^{K+1})| \leq K \quad \Leftrightarrow \quad |\mathcal{A}(G^{K+1}) - (K+1)\,\mathrm{OPT}(G)| \leq K$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Graph G^m ist für $m \in \mathbb{N}$ wie folgt definiert:

1. Kopiere G m-mal
2. Verbinden jeden Knoten einer Kopie mit jedem Knoten aller anderen Kopien

Beob.: $\exists$ Clique C der Größe α in G *gdw.* $\exists$ Clique C^m der Größe αm in G^m.

Annahme: Es gibt Approximationalg. $\mathcal{A}$ mit $|\mathcal{A}(I) - \mathrm{OPT}(I)| \leq K$ (für alle I)

Strategie um größte Clique in G zu finden:

1. Erstelle G^{K+1} von G
2. Wende $\mathcal{A}$ auf G^{K+1} an: liefert Clique C^{K+1}
3. Extrahiere aus C^{K+1} eine Clique C für G der Größe $(K + 1) \cdot |C| = |C^{K+1}|$

$$|\mathcal{A}(G^{K+1}) - \mathrm{OPT}(G^{K+1})| \leq K \quad \Leftrightarrow \quad |\mathcal{A}(G^{K+1}) - (K+1)\,\mathrm{OPT}(G)| \leq K$$

$$||C^{K+1}| - (K+1)\,\mathrm{OPT}(G)| \leq K \quad \Leftrightarrow \quad |(K+1)|C| - (K+1)\,\mathrm{OPT}(G)| \leq K$$

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Graph G^m ist für $m \in \mathbb{N}$ wie folgt definiert:

1. Kopiere G m-mal
2. Verbinden jeden Knoten einer Kopie mit jedem Knoten aller anderen Kopien

Beob.: $\exists$ Clique C der Größe α in G *gdw.* $\exists$ Clique C^m der Größe αm in G^m.

Annahme: Es gibt Approximationalg. $\mathcal{A}$ mit $|\mathcal{A}(I) - \mathrm{OPT}(I)| \leq K$ (für alle I)

Strategie um größte Clique in G zu finden:

1. Erstelle G^{K+1} von G
2. Wende $\mathcal{A}$ auf G^{K+1} an: liefert Clique C^{K+1}
3. Extrahiere aus C^{K+1} eine Clique C für G der Größe $(K+1) \cdot |C| = |C^{K+1}|$

$$|\mathcal{A}(G^{K+1}) - \mathrm{OPT}(G^{K+1})| \leq K \quad \Leftrightarrow \quad |\mathcal{A}(G^{K+1}) - (K+1)\,\mathrm{OPT}(G)| \leq K$$

$$||C^{K+1}| - (K+1)\,\mathrm{OPT}(G)| \leq K \quad \Leftrightarrow \quad |(K+1)|C| - (K+1)\,\mathrm{OPT}(G)| \leq K$$

$$|\,|C| - \mathrm{OPT}(G)| \leq \frac{K}{K+1} < 1$$

Aufgabe

Sei $(G = (V, E), K)$ Instanz von CLIQUE

Graph G^m ist für $m \in \mathbb{N}$ wie folgt definiert:

1. Kopiere G m-mal
2. Verbinden jeden Knoten einer Kopie mit jedem Knoten aller anderen Kopien

Beob.: $\exists$ Clique C der Größe α in G *gdw.* $\exists$ Clique C^m der Größe αm in G^m.

Annahme: Es gibt Approximationalg. $\mathcal{A}$ mit $|\mathcal{A}(I) - \mathrm{OPT}(I)| \leq K$ (für alle I)

Strategie um größte Clique in G zu finden:

1. Erstelle G^{K+1} von G
2. Wende $\mathcal{A}$ auf G^{K+1} an: liefert Clique C^{K+1}
3. Extrahiere aus C^{K+1} eine Clique C für G der Größe $(K + 1) \cdot |C| = |C^{K+1}|$

$$|\mathcal{A}(G^{K+1}) - \mathrm{OPT}(G^{K+1})| \leq K \quad \Leftrightarrow \quad |\mathcal{A}(G^{K+1}) - (K+1)\,\mathrm{OPT}(G)| \leq K$$

$$||C^{K+1}| - (K+1)\,\mathrm{OPT}(G)| \leq K \quad \Leftrightarrow \quad |(K+1)|C| - (K+1)\,\mathrm{OPT}(G)| \leq K$$

$$|\,|C| - \mathrm{OPT}(G)| \leq \frac{K}{K+1} < 1 \quad \longrightarrow \quad |C| = \mathrm{OPT}(G)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wiederholung

Sei Π ein Optimierungsproblem. Ein polynomialer Algorithmus $\mathcal{A}$, der für jedes $I \in D_\Pi$ einen Wert $\mathcal{A}(I)$ liefert mit $R_\mathcal{A}(I) \leq K$, wobei $K \geq 1$ eine Konstante, und

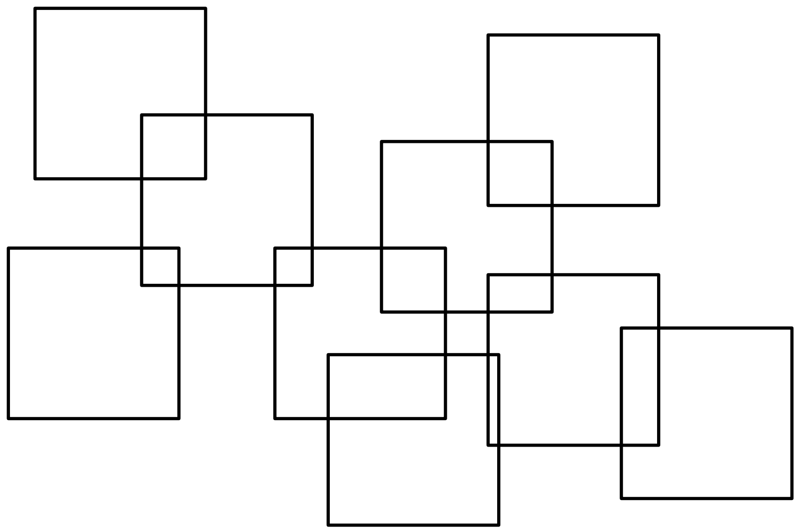
$$\mathcal{R}_\mathcal{A}(I) := \begin{cases} \frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} & \text{falls } \Pi \text{ Minimierungsproblem} \\ \frac{\mathrm{OPT}(I)}{\mathcal{A}(I)} & \text{falls } \Pi \text{ Maximierungsproblem} \end{cases}$$

heißt Approximationsalgorithmus mit relativer Gütegarantie. $\mathcal{A}$ heißt ε–approximativ, falls $\mathcal{R}_\mathcal{A}(I) \leq 1 + \varepsilon$ für alle $I \in D_\Pi$.

Problem **INDEPENDENTSQUARES:**

Gegeben: Menge $Q = \{q_1, \ldots, q_n\}$ gleichgroßer, achsenparalleler Quadrate in der Ebene.

Gesucht: Möglichst große unabhängige Menge $S \subseteq Q$. Dabei heißt $S \subseteq Q$ *unabhängig*, falls für alle $q_i, q_j \in S$ mit $i \neq j$ gilt, dass q_i und q_j sich nicht schneiden.

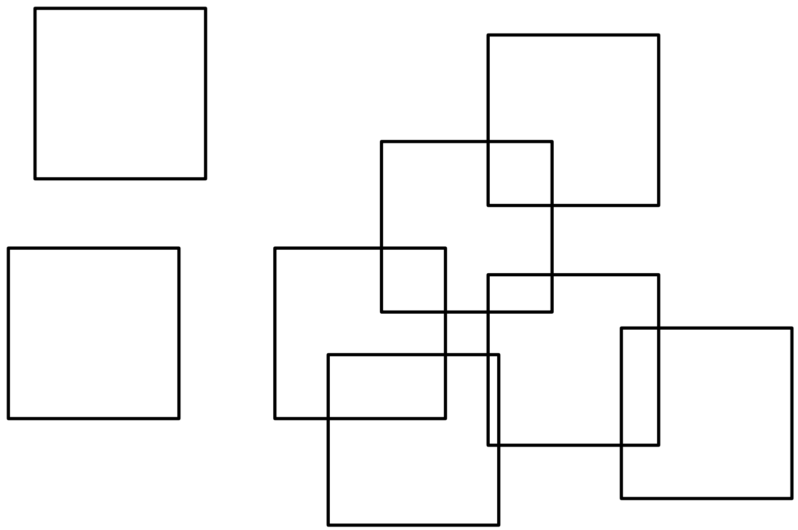


Aufgabe

Problem INDEPENDENTSQUARES:

Gegeben: Menge $Q = \{q_1, \dots, q_n\}$ gleichgroßer, achsenparalleler Quadrate in der Ebene.

Gesucht: Möglichst große unabhängige Menge $S \subseteq Q$. Dabei heißt $S \subseteq Q$ *unabhängig*, falls für alle $q_i, q_j \in S$ mit $i \neq j$ gilt, dass q_i und q_j sich nicht schneiden.



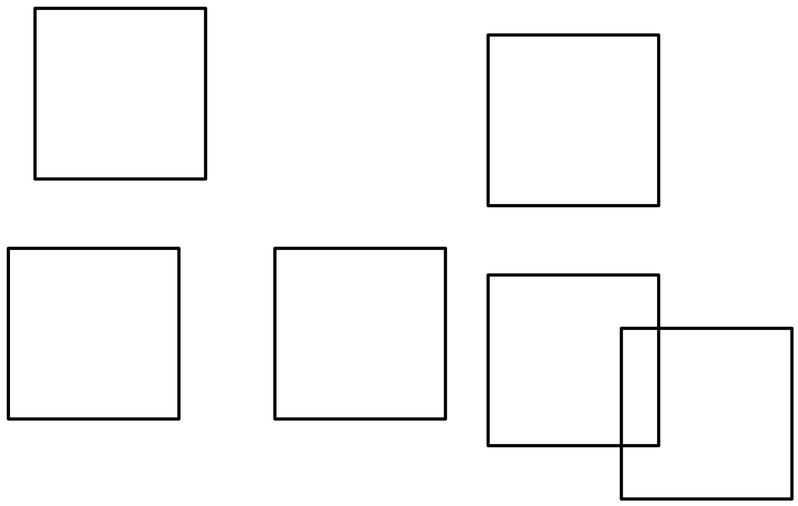
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem INDEPENDENTSQUARES:

Gegeben: Menge $Q = \{q_1, \ldots, q_n\}$ gleichgroßer, achsenparalleler Quadrate in der Ebene.

Gesucht: Möglichst große unabhängige Menge $S \subseteq Q$. Dabei heißt $S \subseteq Q$ *unabhängig*, falls für alle $q_i, q_j \in S$ mit $i \neq j$ gilt, dass q_i und q_j sich nicht schneiden.



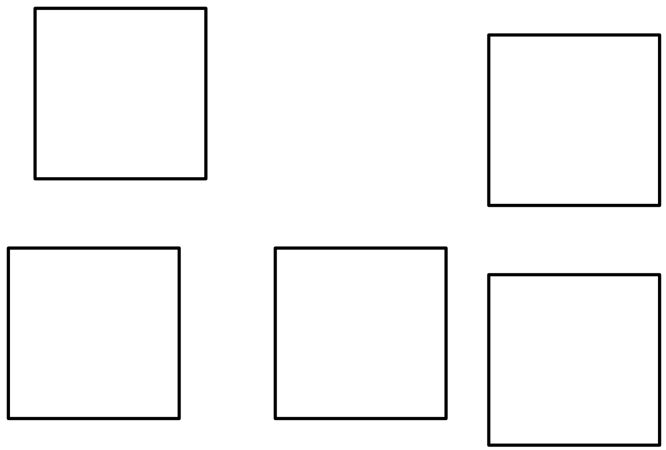
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem INDEPENDENTSQUARES:

Gegeben: Menge $Q = \{q_1, \ldots, q_n\}$ gleichgroßer, achsenparalleler Quadrate in der Ebene.

Gesucht: Möglichst große unabhängige Menge $S \subseteq Q$. Dabei heißt $S \subseteq Q$ *unabhängig*, falls für alle $q_i, q_j \in S$ mit $i \neq j$ gilt, dass q_i und q_j sich nicht schneiden.



Aufgabe

Problem $\textsc{IndependentSquares}$**:**

Gegeben: Menge $Q = \{q_1, \ldots, q_n\}$ gleichgroßer, achsenparalleler Quadrate in der Ebene.

Gesucht: Möglichst große unabhängige Menge $S \subseteq Q$. Dabei heißt $S \subseteq Q$ *unabhängig*, falls für alle $q_i, q_j \in S$ mit $i \neq j$ gilt, dass q_i und q_j sich nicht schneiden.

Eingabe: $Q = \{q_1, \ldots, q_n\}$ gleichgroßer, achsenparalleler Quadrate mit Mittelpunkten $c_1, \ldots, c_n$, sodass für die x-Koordinaten der Mittelpunkte gilt $x(c_1) \leq \ldots \leq x(c_n)$.

Ausgabe: Unabhängige Menge $S \subseteq Q$.

$S \leftarrow \emptyset$;

für $i = 1, \ldots, n$ **tue**

  wenn $q_i \in Q$ **dann**

    $S \leftarrow S \cup \{q_i\}$;

    $Q \leftarrow Q \setminus (\{q_i\} \cup \{q_j \in Q \mid q_j \text{ und } q_i \text{ schneiden sich.}\})$

return S;

Algorithmus 1: $\textsc{SweepLine}$

Aufgabe

Gesucht: Familie $Q_1, Q_2, Q_3, \ldots$ gleichgroßer, achsenparalleler Quadrate an, sodass gilt

$$\left|Q_n\right| \in \Theta(n) \text{ und } \left|\textsc{SweepLine}(Q_n)\right| = \frac{1}{2}\left|\mathrm{OPT}(Q_n)\right|$$

für alle $n \in \mathbb{N}$. Dabei bezeichnet $\mathrm{OPT}(Q)$ die kardinalitätsmaximale unabhängige Menge von Q. Begründen Sie Ihre Antwort.

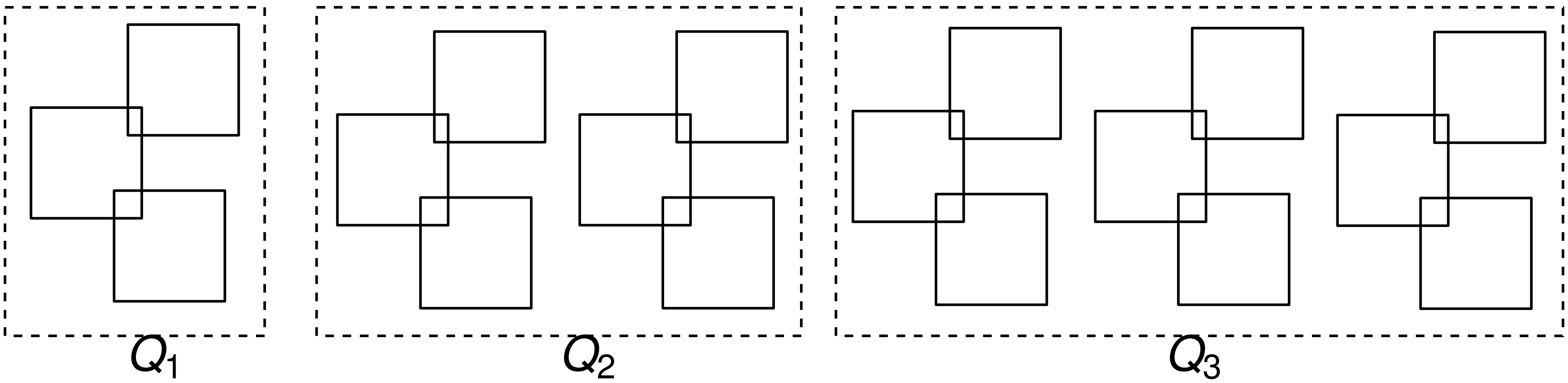
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Gesucht: Familie $Q_1, Q_2, Q_3, \ldots$ gleichgroßer, achsenparalleler Quadrate an, sodass gilt

$$|Q_n| \in \Theta(n) \text{ und } |\textsc{SweepLine}(Q_n)| = \frac{1}{2}|\,\mathrm{OPT}(Q_n)|$$

für alle $n \in \mathbb{N}$. Dabei bezeichnet $\mathrm{OPT}(Q)$ die kardinalitätsmaximale unabhängige Menge von Q. Begründen Sie Ihre Antwort.



Aufgabe

Zeigen Sie, dass SweepLine für IndependentSquares ein Approximationsalgorithmus mit relativer Gütegarantie 2 ist.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Zeigen Sie, dass SweepLine für IndependentSquares ein Approximationsalgorithmus mit relativer Gütegarantie 2 ist.

Betrachte den Fall, dass q_i im i-ten Schritt in S eingefügt wird.

Bezeichne Q_i die Menge Q direkt vor dem i-ten Schritt.

Sei $K = \{q_j \in Q_i \mid q_j \text{ und } q_i \text{ schneiden sich}\}$.

Aufgabe

Zeigen Sie, dass SWEEPLINE für INDEPENDENTSQUARES ein Approximationsalgorithmus mit relativer Gütegarantie 2 ist.

Betrachte den Fall, dass q_i im i-ten Schritt in S eingefügt wird.

Bezeichne Q_i die Menge Q direkt vor dem i-ten Schritt.

Sei $K = \{q_j \in Q_i \mid q_j \text{ und } q_i \text{ schneiden sich}\}$.

Alle Quadrate gleichgroß und achsenparallel

$\longrightarrow$ Jedes Quadrat $q_j \in K$ überdeckt mindestens eine Ecke von q_i.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Zeigen Sie, dass SweepLine für IndependentSquares ein Approximationsalgorithmus mit relativer Gütegarantie 2 ist.

Betrachte den Fall, dass q_i im i-ten Schritt in S eingefügt wird.

Bezeichne Q_i die Menge Q direkt vor dem i-ten Schritt.

Sei $K = \{q_j \in Q_i \mid q_j \text{ und } q_i \text{ schneiden sich}\}$.

Alle Quadrate gleichgroß und achsenparallel

$\longrightarrow$ Jedes Quadrat $q_j \in K$ überdeckt mindestens eine Ecke von q_i.

$x(c_j) \geq x(c_i)$ für alle $q_j \in Q$:

$\longrightarrow$ Obere-rechte oder untere-rechte Ecke von q_i muss überdeckt sein.

$\longrightarrow$ $|\mathrm{OPT}(K)| \leq 2$

Aufgabe

Zeigen Sie, dass SweepLine für IndependentSquares ein Approximationsalgorithmus mit relativer Gütegarantie 2 ist.

Betrachte den Fall, dass q_i im i-ten Schritt in S eingefügt wird.

Bezeichne Q_i die Menge Q direkt vor dem i-ten Schritt.

Sei $K = \{q_j \in Q_i \mid q_j$ und q_i schneiden sich$\}$.

Alle Quadrate gleichgroß und achsenparallel

$\longrightarrow$ Jedes Quadrat $q_j \in K$ überdeckt mindestens eine Ecke von q_i.

$x(c_j) \geq x(c_i)$ für alle $q_j \in Q$:

$\longrightarrow$ Obere-rechte oder untere-rechte Ecke von q_i muss überdeckt sein.

$\longrightarrow$ $|\mathrm{OPT}(K)| \leq 2$

Schlimmster Fall: Zwei Quadrate der optimalen Lösung gehen verloren, während eins zur Lösung hinzugenommen wird.

$\longrightarrow$ Relative Gütegarantie

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Sei $G = (V, E)$ ein gerichteter Graph und sei $G_1 = (V, E_1 \subseteq E)$ ein inklusionsmaximaler azyklischer Teilgraph von G. Zudem sei $G_2 = (V, E_2 = E \setminus E_1)$ das Komplement zu G_1.

Zeige: Für jede Kante $(u, v) \in E_2$ gibt es in G_1 einen gerichteten Pfad von v nach u.

Aufgabe

Sei $G = (V, E)$ ein gerichteter Graph und sei $G_1 = (V, E_1 \subseteq E)$ ein inklusionsmaximaler azyklischer Teilgraph von G. Zudem sei $G_2 = (V, E_2 = E \setminus E_1)$ das Komplement zu G_1.

Zeige: Für jede Kante $(u, v) \in E_2$ gibt es in G_1 einen gerichteten Pfad von v nach u.

Annahme: Es gibt Kante $(u, v) \in E_2$, sodass es keinen gerichteten Pfad von v nach u in G_1 gibt.

$\longrightarrow$ Kante (u, v) kann zu E_1 hinzu genommen werden, ohne das ein gerichteter Kreis entsteht.

$\lightning$ G_1 inklusionsmaximal, azyklischer Graph ist.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Sei $G = (V, E)$ ein gerichteter Graph und sei $G_1 = (V, E_1 \subseteq E)$ ein inklusionsmaximaler azyklischer Teilgraph von G. Zudem sei $G_2 = (V, E_2 = E \setminus E_1)$ das Komplement zu G_1.

Zeige: G_2 ist azyklisch.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

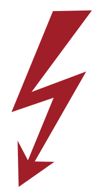
Sei $G = (V, E)$ ein gerichteter Graph und sei $G_1 = (V, E_1 \subseteq E)$ ein inklusionsmaximaler azyklischer Teilgraph von G. Zudem sei $G_2 = (V, E_2 = E \setminus E_1)$ das Komplement zu G_1.

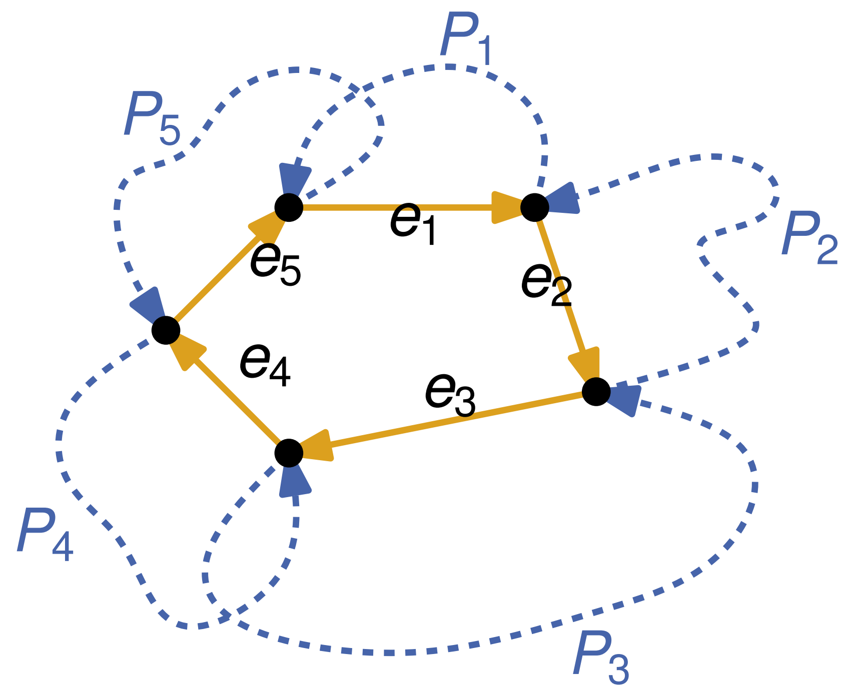
Zeige: G_2 ist azyklisch.

Annahme: G_2 enthält einen Kreis $P = (e_1, e_2, \ldots, e_k)$.

Teilaufagabe (a): Für jede dieser Kanten $e_i \in P$ gibt es gerichteten Pfad P_i in G_1, der vom Zielknoten von e_i zum Startknoten von e_i führt.

Verbinde Pfade zu einem Kreis.





Maximum Acyclic Graph

Karlsruher Institut für Technologie

Problem MAXIMUM ACYCLIC GRAPH:

> **Gegeben:** Gerichteter Graph $G = (V, E)$.
> **Gesucht:** Kardinalitätsmaximaler azyklischer Teilgraph von G.

Gesucht: Approx.algo. für MAXIMUM ACYCLIC GRAPH mit relativer Gütegarantie 2.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Maximum Acyclic Graph

Karlsruher Institut für Technologie

Problem MAXIMUM ACYCLIC GRAPH:

Gegeben: Gerichteter Graph $G = (V, E)$.
Gesucht: Kardinalitätsmaximaler azyklischer Teilgraph von G.

Gesucht: Approx.algo. für MAXIMUM ACYCLIC GRAPH mit relativer Gütegarantie 2.

Berechne inklusionsmaximalen azyklischen Teilgraph $G_1 = (V, E_1)$ von G.;
Berechne Komplementgraph $G_2 = (V, E \setminus E_1)$ zu G_1.;
wenn $|E_1| \geq |E_2|$ **dann**
  return G_1;
sonst
  return G_2;

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Maximum Acyclic Graph

Problem MAXIMUM ACYCLIC GRAPH:

Gegeben: Gerichteter Graph $G = (V, E)$.
Gesucht: Kardinalitätsmaximaler azyklischer Teilgraph von G.

Gesucht: Approx.algo. für MAXIMUM ACYCLIC GRAPH mit relativer Gütegarantie 2.

Berechne inklusionsmaximalen azyklischen Teilgraph $G_1 = (V, E_1)$ von G.;
Berechne Komplementgraph $G_2 = (V, E \setminus E_1)$ zu G_1.;
wenn $|E_1| \geq |E_2|$ **dann**
  return G_1;
sonst
  return G_2;

- G_2 ist azyklisch.
- $|E_1| + |E_2| = |E|$ und $E_1 \cap E_2 \neq \emptyset$

Maximum Acyclic Graph

Problem MAXIMUM ACYCLIC GRAPH:

Gegeben: Gerichteter Graph $G = (V, E)$.
Gesucht: Kardinalitätsmaximaler azyklischer Teilgraph von G.

Gesucht: Approx.algo. für MAXIMUM ACYCLIC GRAPH mit relativer Gütegarantie 2.

Berechne inklusionsmaximalen azyklischen Teilgraph $G_1 = (V, E_1)$ von G.;
Berechne Komplementgraph $G_2 = (V, E \setminus E_1)$ zu G_1.;
wenn $|E_1| \geq |E_2|$ **dann**
  return G_1;
sonst
  return G_2;

- G_2 ist azyklisch.
- $|E_1| + |E_2| = |E|$ und $E_1 \cap E_2 \neq \emptyset$

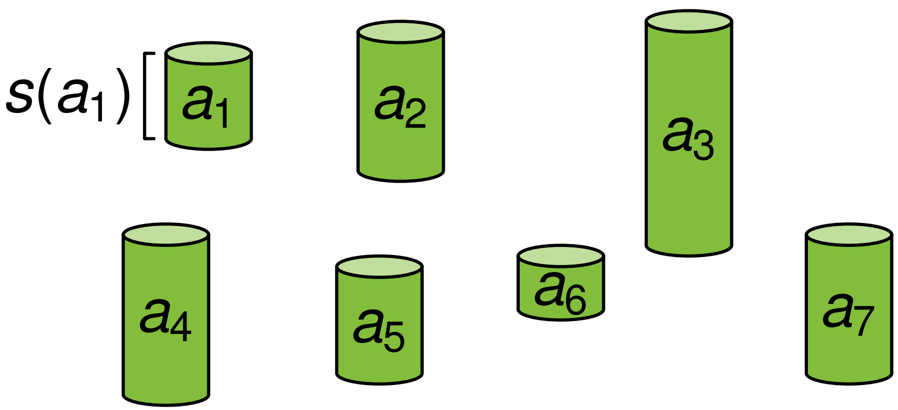
  → $|E_1| \geq \frac{1}{2}|E|$ oder $|E_2| \geq \frac{1}{2}|E|$

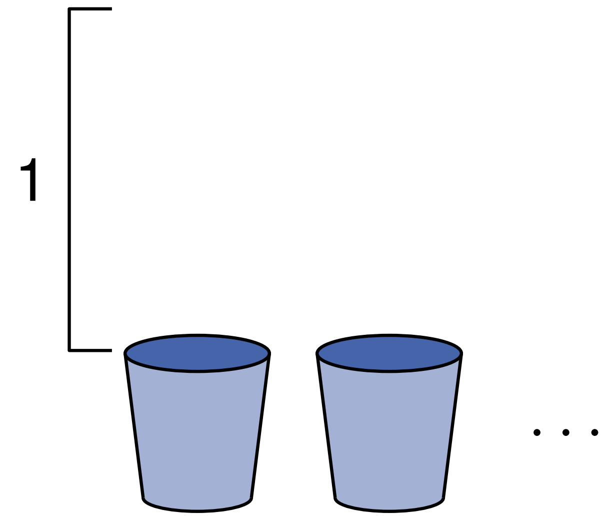
  → Optimale Lösung kann nicht mehr als $|E|$ Kanten enthalten.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bin Packing – Definition

endliche Menge $M = \{a_1, \ldots, a_n\}$
mit Gewichtsfunktion
$s\colon M \longrightarrow (0, 1]$





Eimer (Bins) mit Fassungsvermögen 1

Problem BIN PACKING:
Weise die Elemente in M einer minimalen Anzahl an Bins $B_1, \ldots, B_m$ zu, sodass für jeden Bin B gilt:

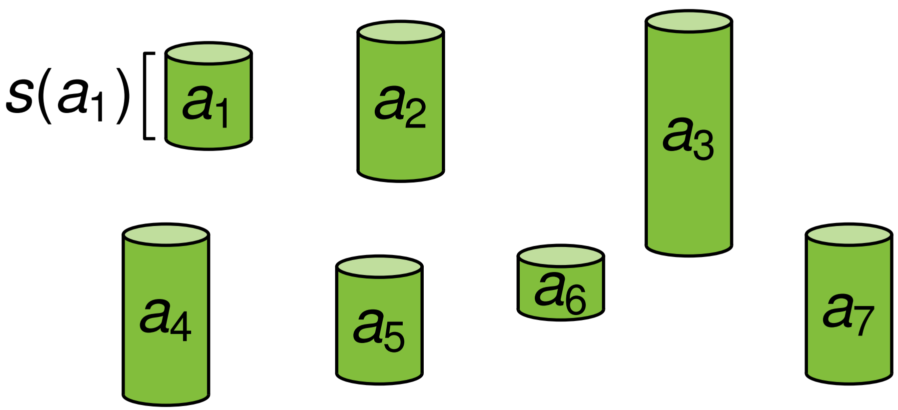
$$\sum_{a_i \in B} s(a_i) \leq 1$$

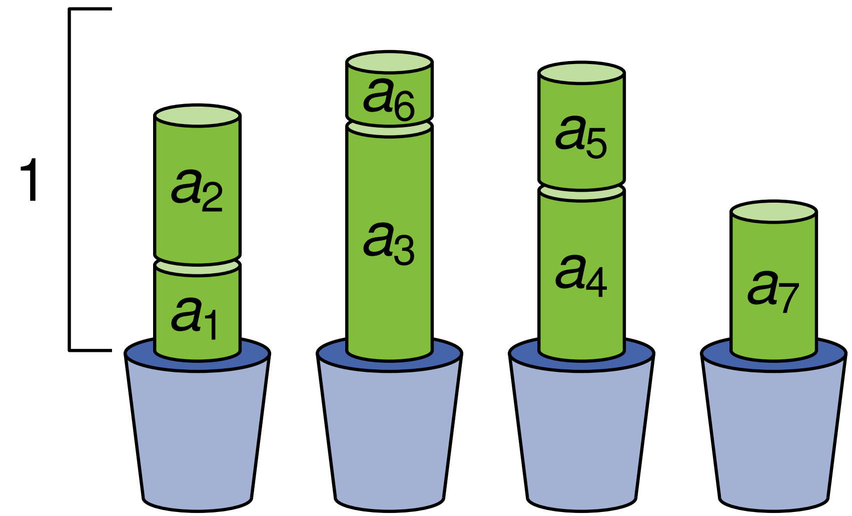
Bin Packing – Definition

endliche Menge $M = \{a_1, \ldots, a_n\}$

mit Gewichtsfunktion

$s\colon M \longrightarrow (0, 1]$





Eimer (Bins) mit Fassungsvermögen 1

4 Bins

Problem BIN PACKING:

Weise die Elemente in M einer minimalen Anzahl an Bins $B_1, \ldots, B_m$ zu, sodass für jeden Bin B gilt:

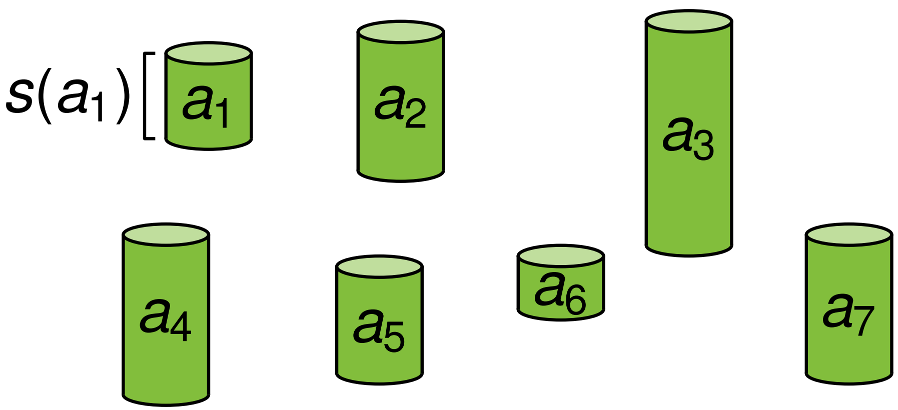
$$\sum_{a_i \in B} s(a_i) \leq 1$$

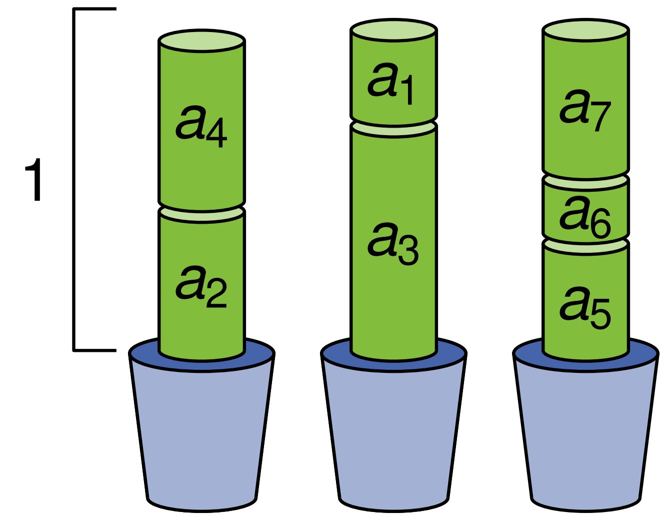
Bin Packing – Definition

endliche Menge $M = \{a_1, \dots, a_n\}$

mit Gewichtsfunktion

$$s\colon M \longrightarrow (0, 1]$$





Eimer (Bins) mit Fassungsvermögen 1

3 Bins

Problem Bin Packing:

Weise die Elemente in M einer minimalen Anzahl an Bins $B_1, \dots, B_m$ zu, sodass für jeden Bin B gilt:

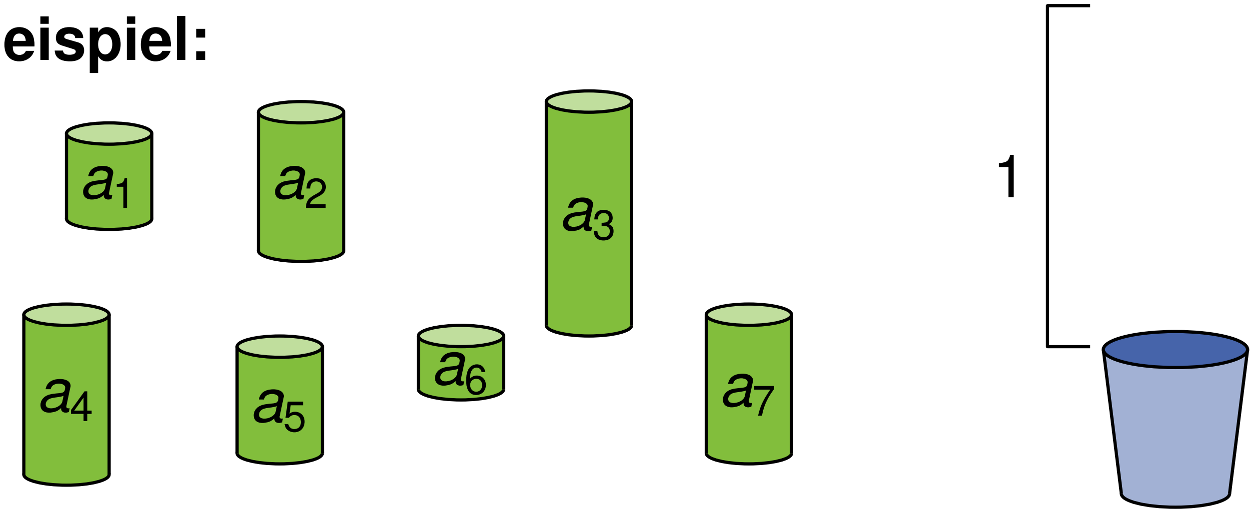
$$\sum_{a_i \in B} s(a_i) \leq 1$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bin-Packing

Strategie:
Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

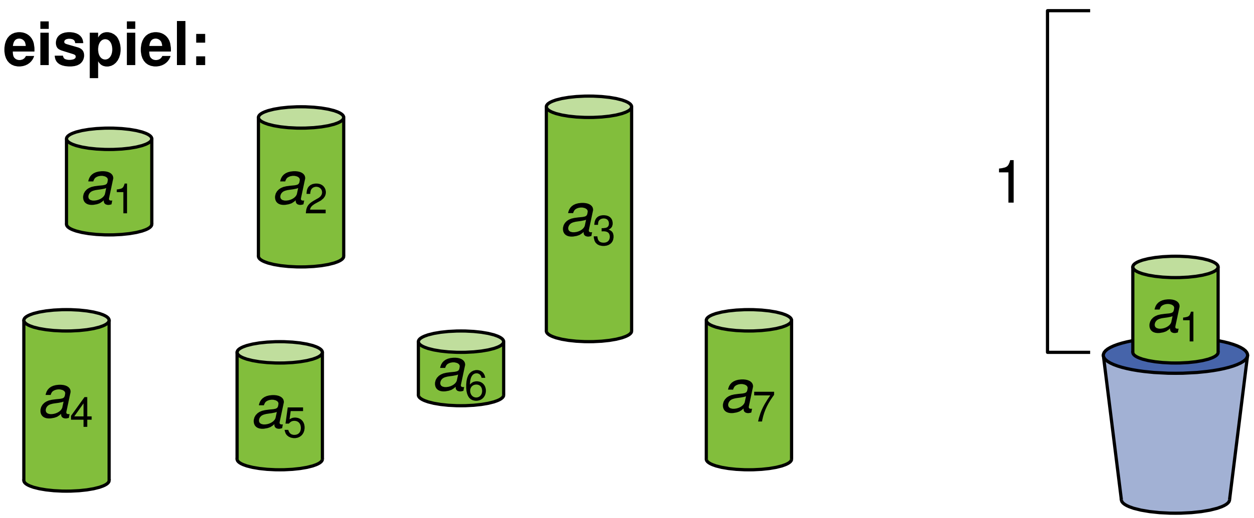
Beispiel:



Bin-Packing

Strategie:
Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

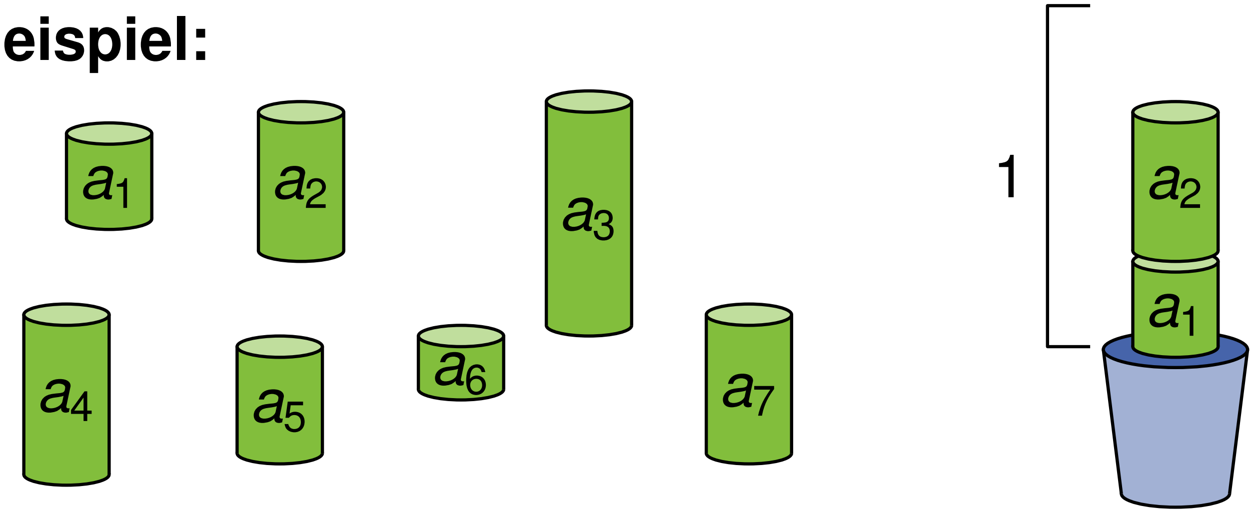
Beispiel:



Bin-Packing

Strategie:
Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

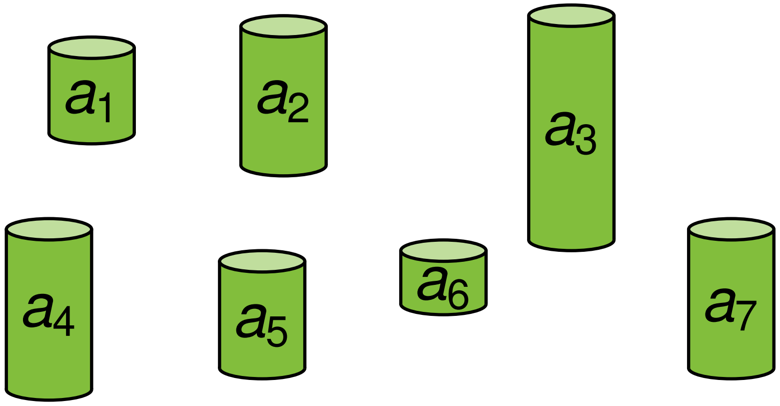
Beispiel:

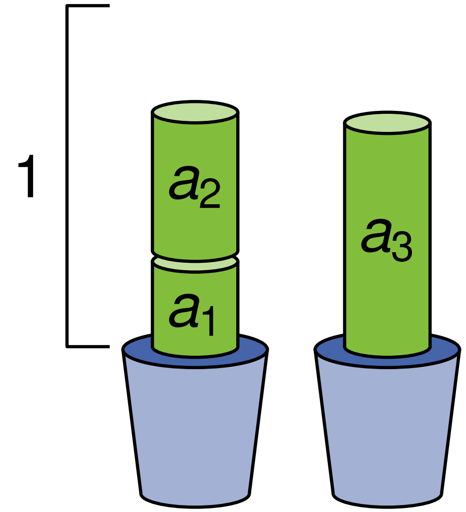


Bin-Packing

Strategie:
Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

Beispiel:

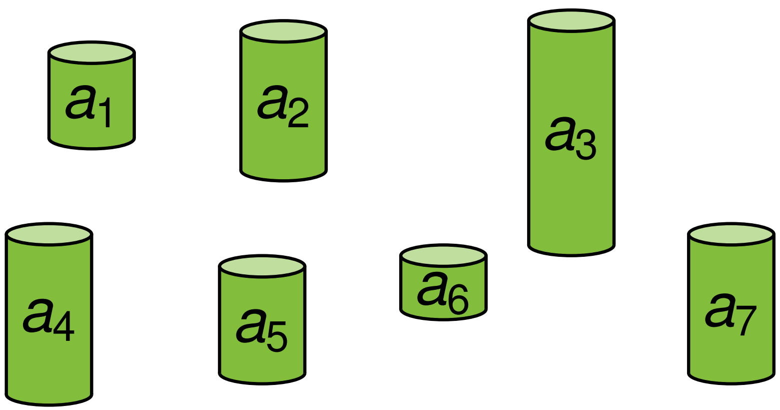


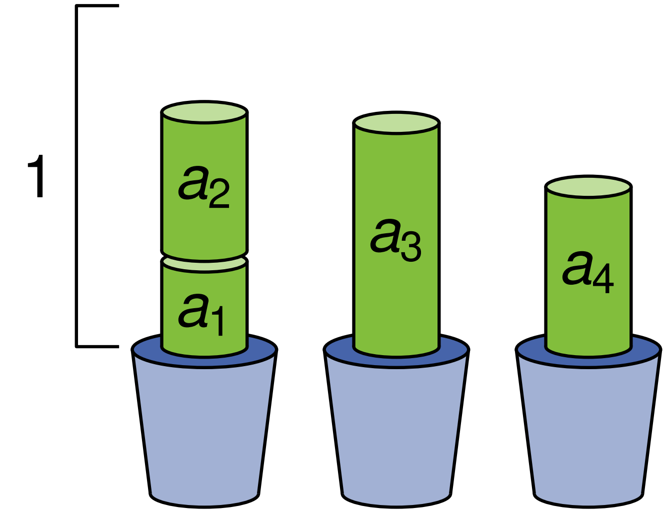


Bin-Packing

Strategie:
Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

Beispiel:



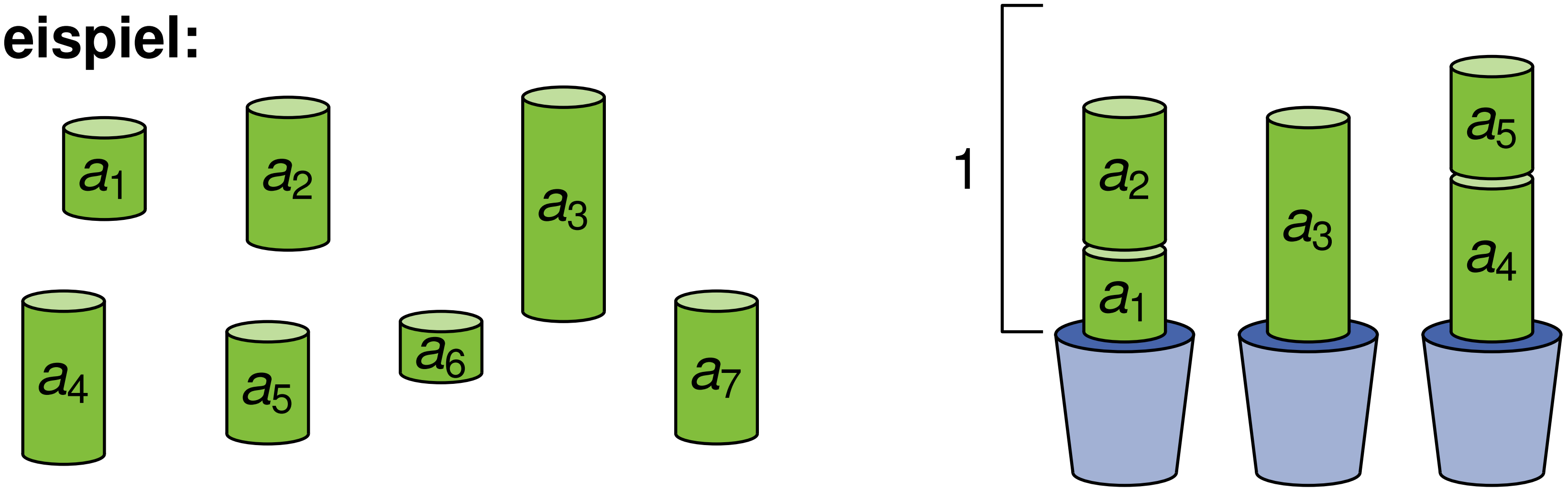


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bin-Packing

Strategie:
Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

Beispiel:



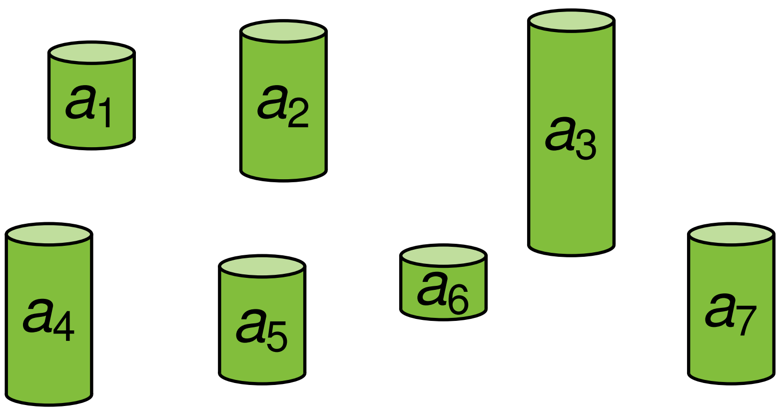
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

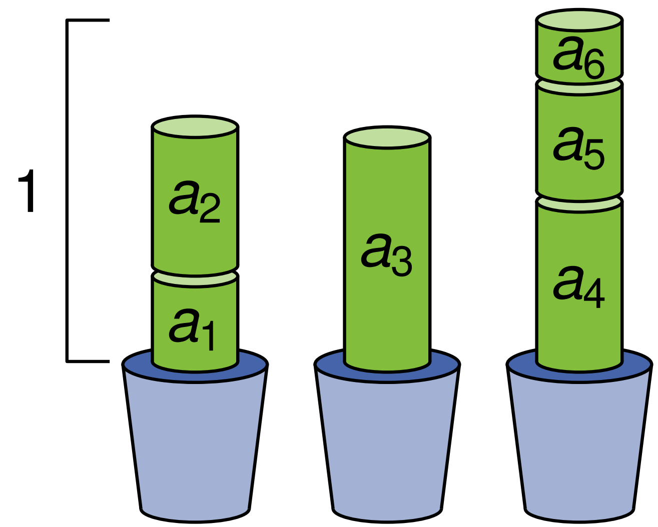
Bin-Packing

Strategie:

Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

Beispiel:

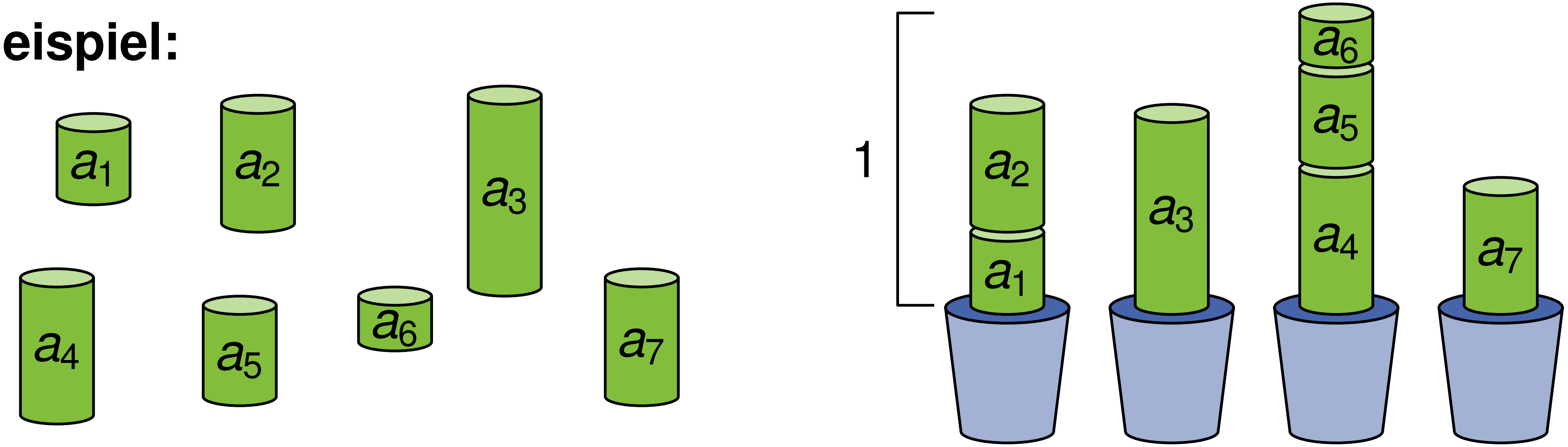




Bin-Packing

Strategie:
Füge Elemente nacheinander in den aktuellen Bin ein. Wenn ein Element nicht mehr passt, schließe den Bin ab und nimm einen neuen.

Beispiel:

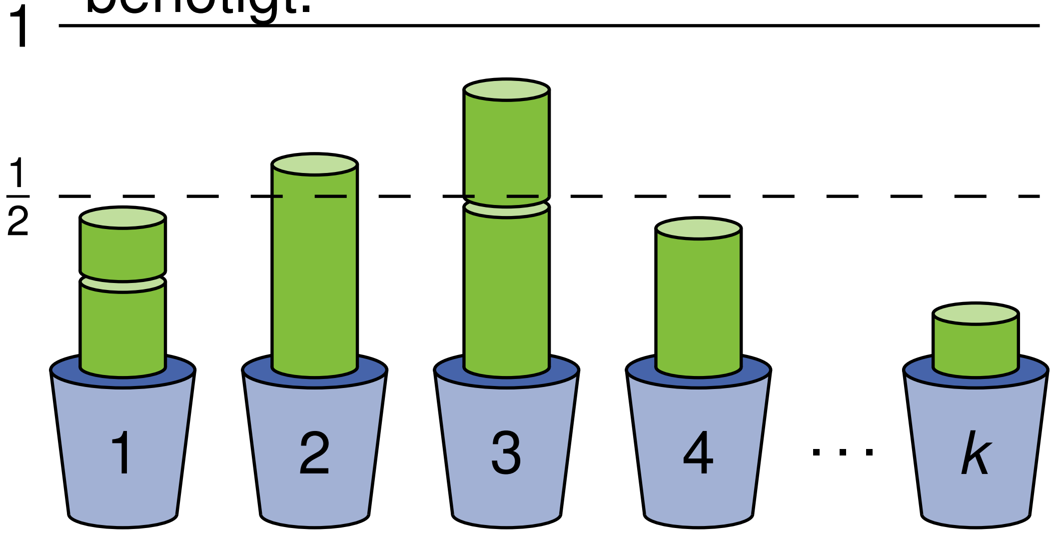


Next Fit – Approximation

Satz: Approximation
NEXT FIT erfüllt $\mathcal{R}_{\text{NF}} \leq 2$.

Beweis:

- Sei $k = \text{NF}(I)$ die Anzahl an Bins, die NEXTFIT für die Instanz I benötigt.



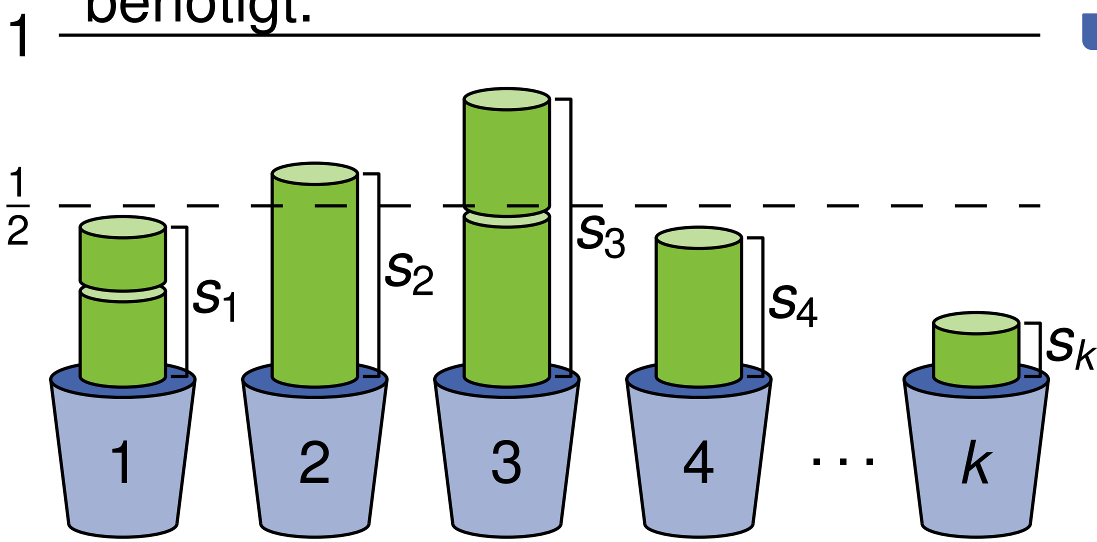
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Next Fit – Approximation

Satz: Approximation
NEXT FIT erfüllt $\mathcal{R}_{\text{NF}} \leq 2$.

Beweis:

- Sei k = NF(I) die Anzahl an Bins, die NEXTFIT für die Instanz I benötigt.



- Sei s_i die Größe der Elemente in Bin i.

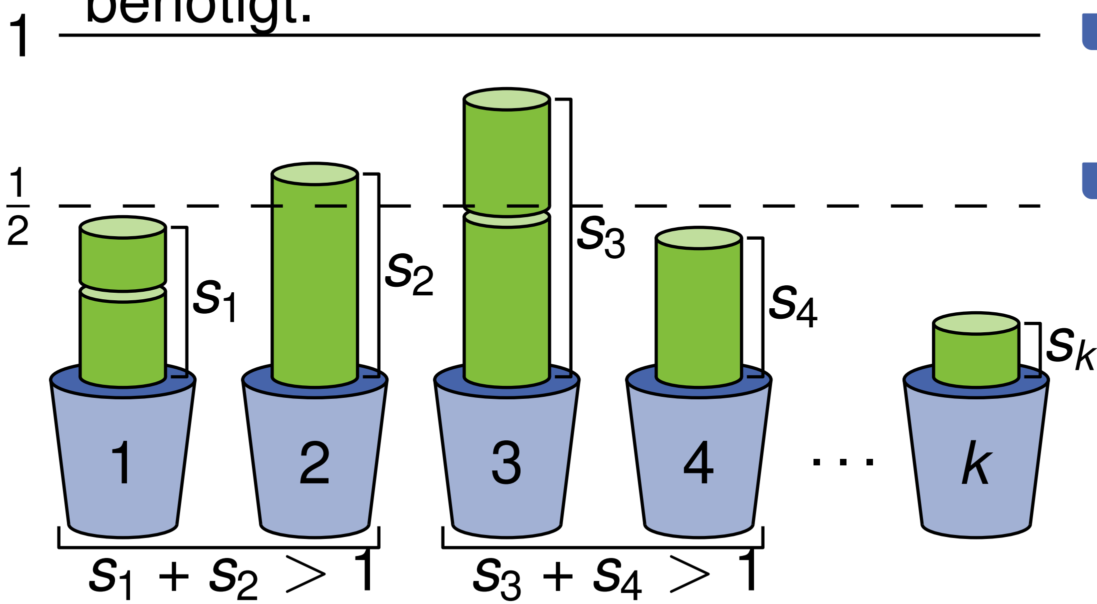
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Next Fit – Approximation

Satz: Approximation
NEXT FIT erfüllt $\mathcal{R}_{\text{NF}} \leq 2$.

Beweis:

- Sei k = NF(I) die Anzahl an Bins, die NEXTFIT für die Instanz I benötigt.



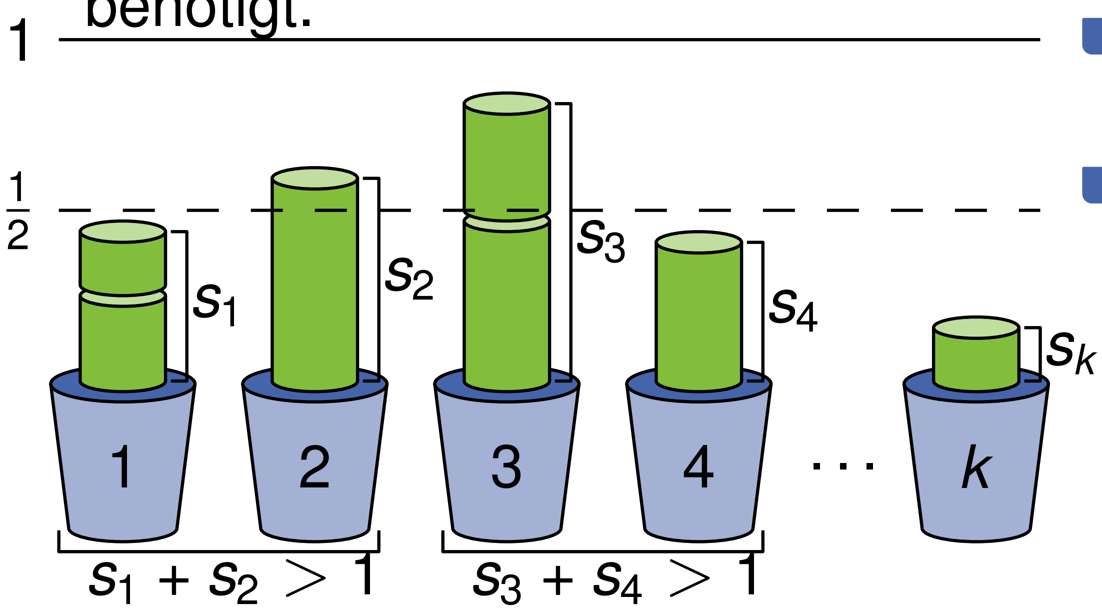
- Sei s_i die Größe der Elemente in Bin i.
- Für zwei aufeinanderfolgende Bins gilt: $s_i + s_{i+1} > 1$
  (sonst hätten die Elemente in Bin $i + 1$ noch in Bin i gepasst)

Next Fit – Approximation

Satz: Approximation
Next Fit erfüllt $\mathcal{R}_{\text{NF}} \leq 2$.

Beweis:

- Sei $k = \text{NF}(I)$ die Anzahl an Bins, die NextFit für die Instanz I benötigt.



- Sei s_i die Größe der Elemente in Bin i.
- Für zwei aufeinanderfolgende Bins gilt: $s_i + s_{i+1} > 1$
  (sonst hätten die Elemente in Bin $i+1$ noch in Bin i gepasst)

$\Rightarrow \sum_{i=1}^{k} s_i > \frac{k}{2}$ falls k gerade bzw. $\sum_{i=1}^{k-1} s_i > \frac{k-1}{2}$ falls k ungerade

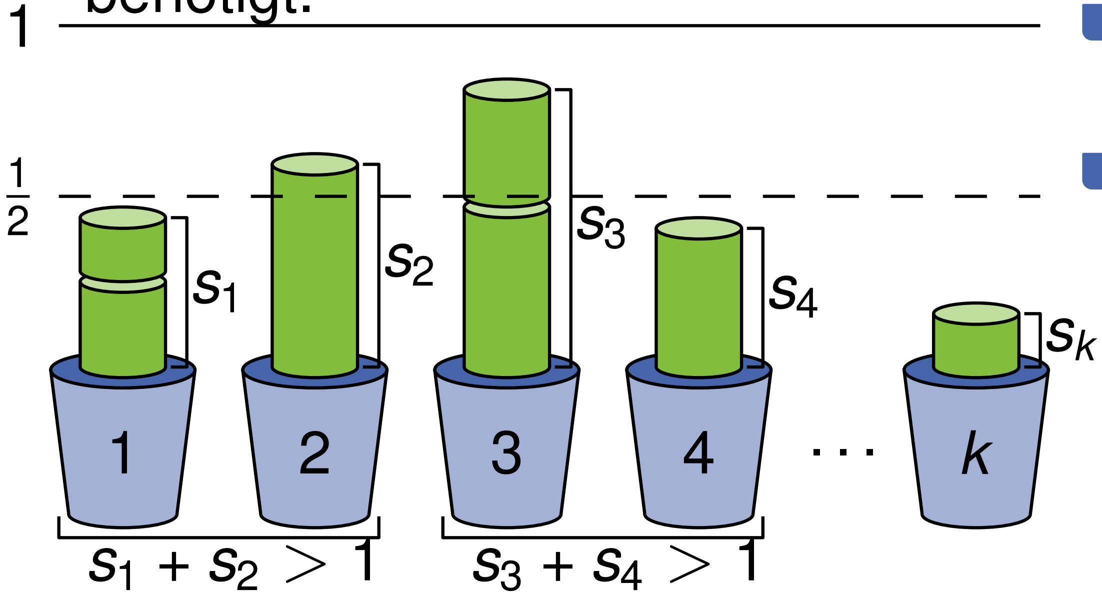
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Next Fit – Approximation

Satz: Approximation
NEXT FIT erfüllt $\mathcal{R}_{\text{NF}} \leq 2$.

Beweis:

- Sei $k = \text{NF}(I)$ die Anzahl an Bins, die NEXTFIT für die Instanz I benötigt.



- Sei s_i die Größe der Elemente in Bin i.
- Für zwei aufeinanderfolgende Bins gilt: $s_i + s_{i+1} > 1$ (sonst hätten die Elemente in Bin $i+1$ noch in Bin i gepasst)

$$\Rightarrow \sum_{i=1}^{k} s_i > \frac{k}{2} \text{ falls } k \text{ gerade bzw. } \sum_{i=1}^{k-1} s_i > \frac{k-1}{2} \text{ falls } k \text{ ungerade}$$

$$\Rightarrow \text{OPT}(I) > \frac{k-1}{2} \Rightarrow \text{NF}(I) = k < 2\text{OPT}(I) + 1 \Rightarrow \text{NF}(I) \leq 2\text{OPT}(I)$$

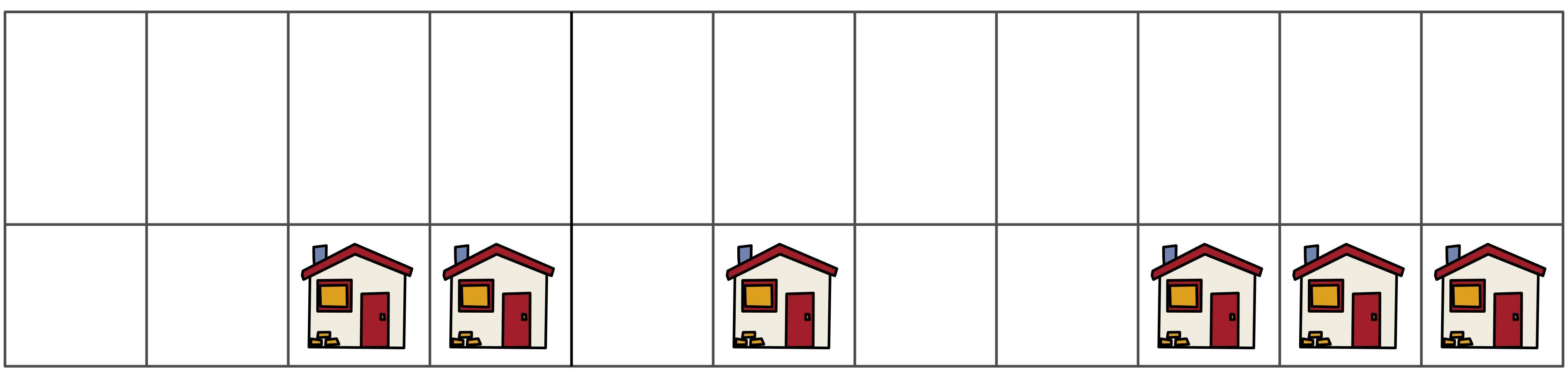
Klausuraufgabe WS16/17

Gegeben:

- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:

Abdeckung aller Häuser mit möglichst wenig Sendern



Klausuraufgabe WS16/17

Gegeben:

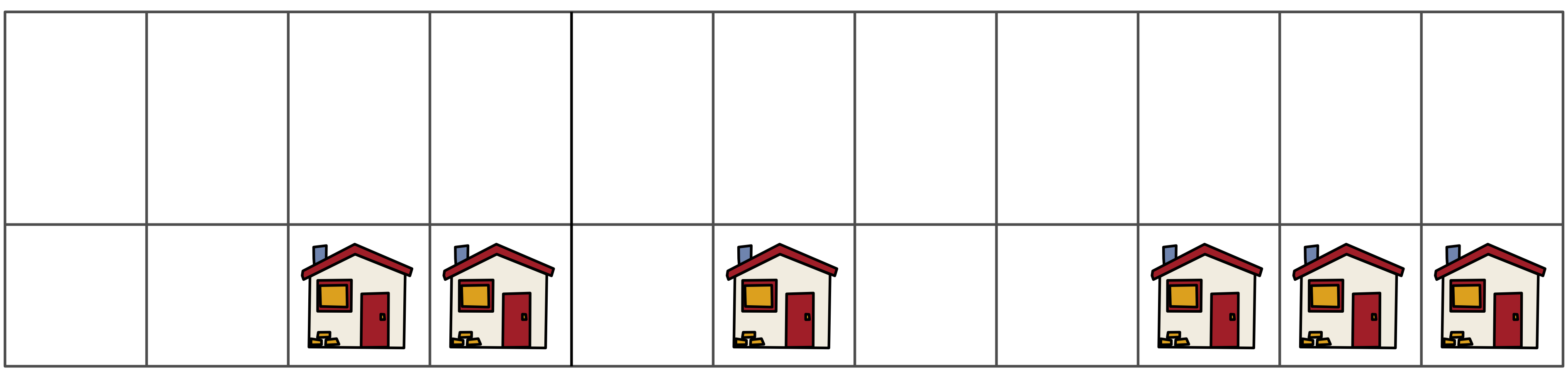
- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:
Abdeckung aller Häuser mit möglichst wenig Sendern

Greedy-Algorithmus $\mathcal{A}$:

solange nicht jedes Haus abgedeckt:

- wähle nicht abgedecktes Haus mit kleinster Zelle i
- platziere Sender auf Haus in größter Zelle im Intervall $\{i, i + 1, \ldots, i + k\}$



Klausuraufgabe WS16/17

Gegeben:

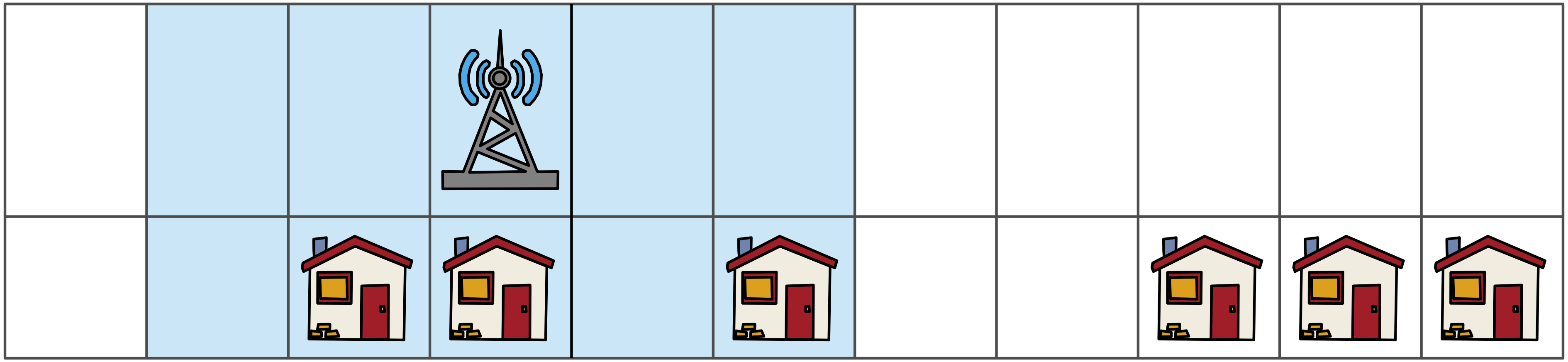
- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:
Abdeckung aller Häuser mit möglichst wenig Sendern

Greedy-Algorithmus $\mathcal{A}$:

solange nicht jedes Haus abgedeckt:

- wähle nicht abgedecktes Haus mit kleinster Zelle i
- platziere Sender auf Haus in größter Zelle im Intervall $\{i, i+1, \ldots, i+k\}$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Gegeben:

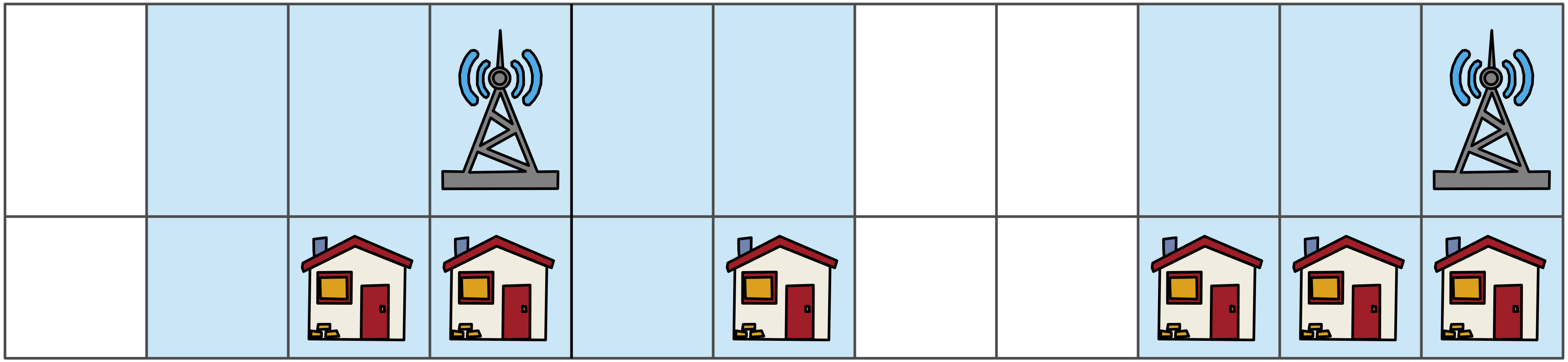
- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:
Abdeckung aller Häuser mit möglichst wenig Sendern

Greedy-Algorithmus $\mathcal{A}$:

solange nicht jedes Haus abgedeckt:

- wähle nicht abgedecktes Haus mit kleinster Zelle i
- platziere Sender auf Haus in größter Zelle im Intervall $\{i, i+1, \ldots, i+k\}$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Gegeben:

- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:
Abdeckung aller Häuser mit möglichst wenig Sendern

Zeigen Sie: $\mathcal{A}$ ist optimal

Greedy-Algorithmus $\mathcal{A}$:

solange nicht jedes Haus abgedeckt:

- wähle nicht abgedecktes Haus mit kleinster Zelle i
- platziere Sender auf Haus in größter Zelle in $\{i, i+1, \ldots, i+k\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Gegeben:

- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:
Abdeckung aller Häuser mit möglichst wenig Sendern

Greedy-Algorithmus $\mathcal{A}$:

solange nicht jedes Haus abgedeckt:

- wähle nicht abgedecktes Haus mit kleinster Zelle i
- platziere Sender auf Haus in größter Zelle in $\{i, i+1, \ldots, i+k\}$

Zeigen Sie: $\mathcal{A}$ ist optimal

- sei $(a_1, a_2, \ldots, a_t)$ $\mathcal{A}$-Lösung
- sei $(o_1, o_2, \ldots, o_{t'})$ beliebige aber feste optimale Lösung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Gegeben:

- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:
Abdeckung aller Häuser mit möglichst wenig Sendern

Greedy-Algorithmus $\mathcal{A}$:

solange nicht jedes Haus abgedeckt:

- wähle nicht abgedecktes Haus mit kleinster Zelle i
- platziere Sender auf Haus in größter Zelle in $\{i, i+1, \ldots, i+k\}$

Zeigen Sie: $\mathcal{A}$ ist optimal

- sei $(a_1, a_2, \ldots, a_t)$ $\mathcal{A}$-Lösung
- sei $(o_1, o_2, \ldots, o_{t'})$ beliebige aber feste optimale Lösung
- vergleiche a_1, o_1: es ist $a_1 \geq o_1$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Gegeben:

- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:
Abdeckung aller Häuser mit möglichst wenig Sendern

Greedy-Algorithmus $\mathcal{A}$:

solange nicht jedes Haus abgedeckt:

- wähle nicht abgedecktes Haus mit kleinster Zelle i
- platziere Sender auf Haus in größter Zelle in $\{i, i+1, \ldots, i+k\}$

Zeigen Sie: $\mathcal{A}$ ist optimal

- sei $(a_1, a_2, \ldots, a_t)$ $\mathcal{A}$-Lösung
- sei $(o_1, o_2, \ldots, o_{t'})$ beliebige aber feste optimale Lösung
- vergleiche a_1, o_1: es ist $a_1 \geq o_1$
- dann ist auch $(a_1, o_2, o_3, \ldots, o_{t'})$ eine optimale Lösung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Gegeben:

- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:
Abdeckung aller Häuser mit möglichst wenig Sendern

Greedy-Algorithmus $\mathcal{A}$:

solange nicht jedes Haus abgedeckt:

- wähle nicht abgedecktes Haus mit kleinster Zelle i
- platziere Sender auf Haus in größter Zelle in $\{i, i+1, \ldots, i+k\}$

Zeigen Sie: $\mathcal{A}$ ist optimal

- sei $(a_1, a_2, \ldots, a_t)$ $\mathcal{A}$-Lösung
- sei $(o_1, o_2, \ldots, o_{t'})$ beliebige aber feste optimale Lösung
- vergleiche a_1, o_1: es ist $a_1 \geq o_1$
- dann ist auch $(a_1, o_2, o_3, \ldots, o_{t'})$ eine optimale Lösung
- Induktionsschritt: $(a_1, a_2, \ldots, a_i, o_{i+1}, o_{i+2}, \ldots, o_{t'})$ ist optimale Lösung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Gegeben:

- Straße aus Zellen $\{1, 2, \ldots, n\}$
- Häuser in gewissen Zellen
- Sender mit Reichweite k

Gesucht:
Abdeckung aller Häuser mit möglichst wenig Sendern

Greedy-Algorithmus $\mathcal{A}$:

solange nicht jedes Haus abgedeckt:

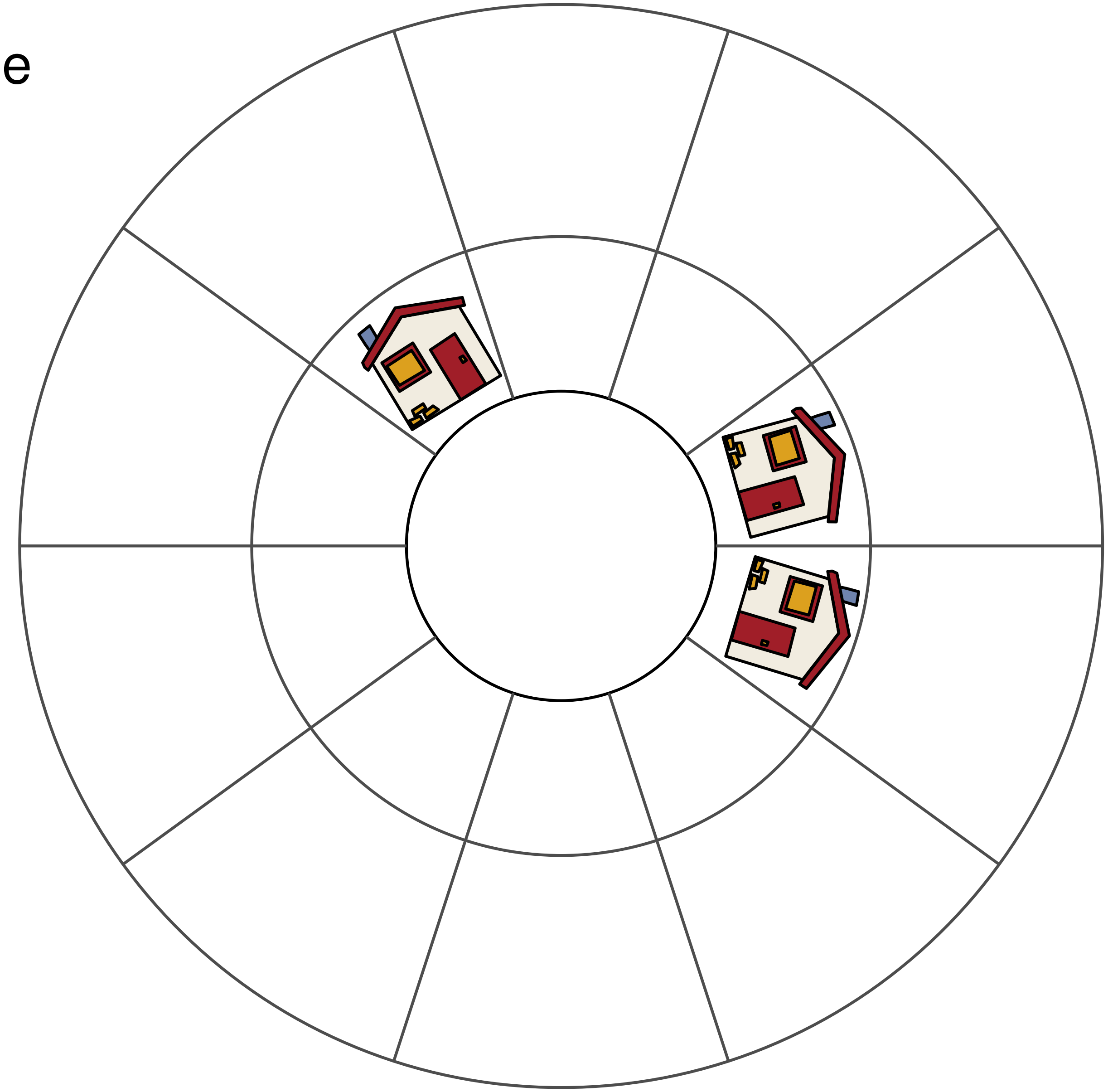
- wähle nicht abgedecktes Haus mit kleinster Zelle i
- platziere Sender auf Haus in größter Zelle in $\{i, i+1, \ldots, i+k\}$

Zeigen Sie: $\mathcal{A}$ ist optimal

- sei $(a_1, a_2, \ldots, a_t)$ $\mathcal{A}$-Lösung
- sei $(o_1, o_2, \ldots, o_{t'})$ beliebige aber feste optimale Lösung
- vergleiche a_1, o_1: es ist $a_1 \geq o_1$
- dann ist auch $(a_1, o_2, o_3, \ldots, o_{t'})$ eine optimale Lösung
- Induktionsschritt: $(a_1, a_2, \ldots, a_i, o_{i+1}, o_{i+2}, \ldots, o_{t'})$ ist optimale Lösung
- also: $(a_1, a_2, \ldots, a_t)$ ist opt. Lsg.

Klausuraufgabe WS16/17

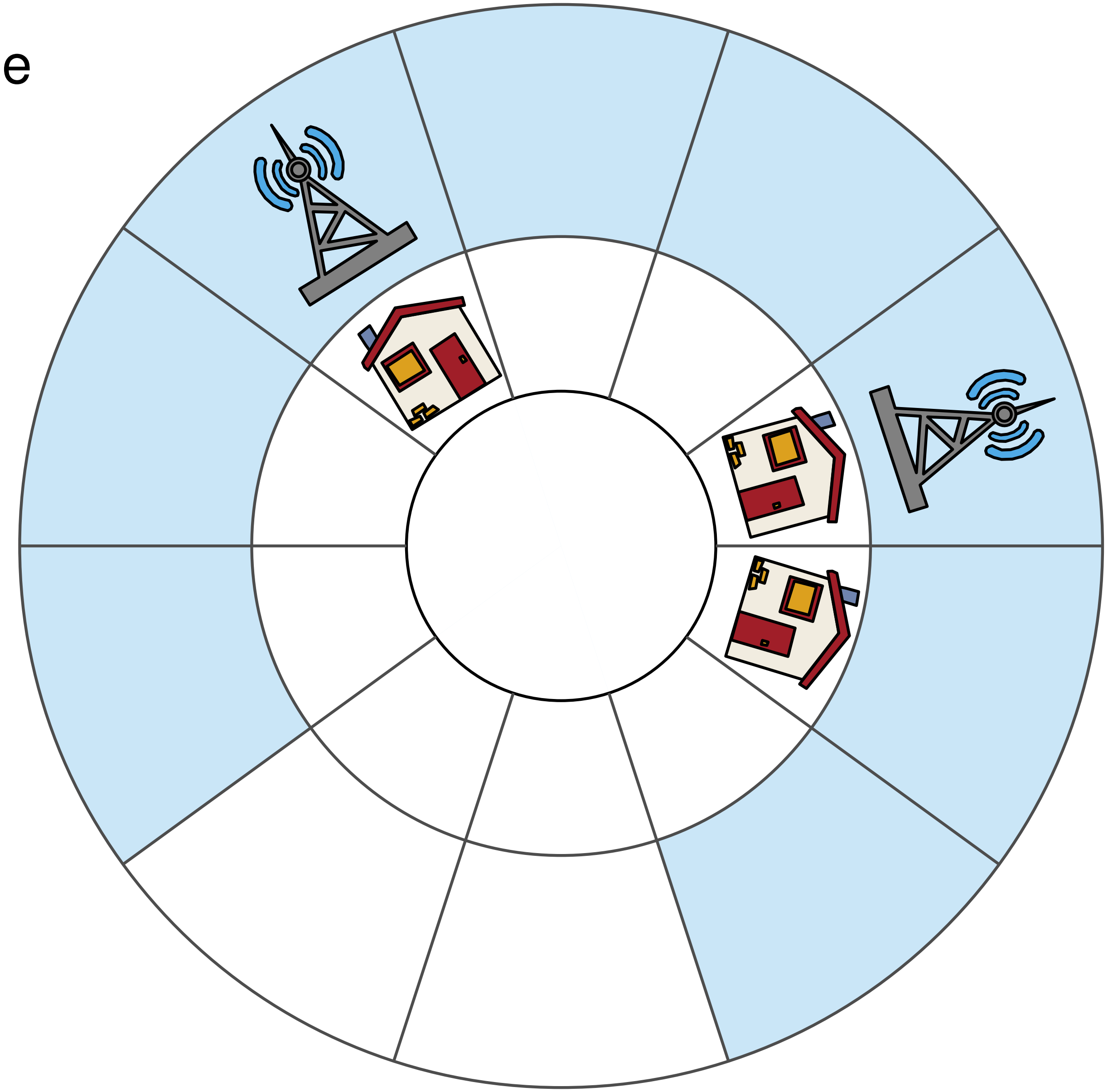
Jetzt: kreisförmige Straße



Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

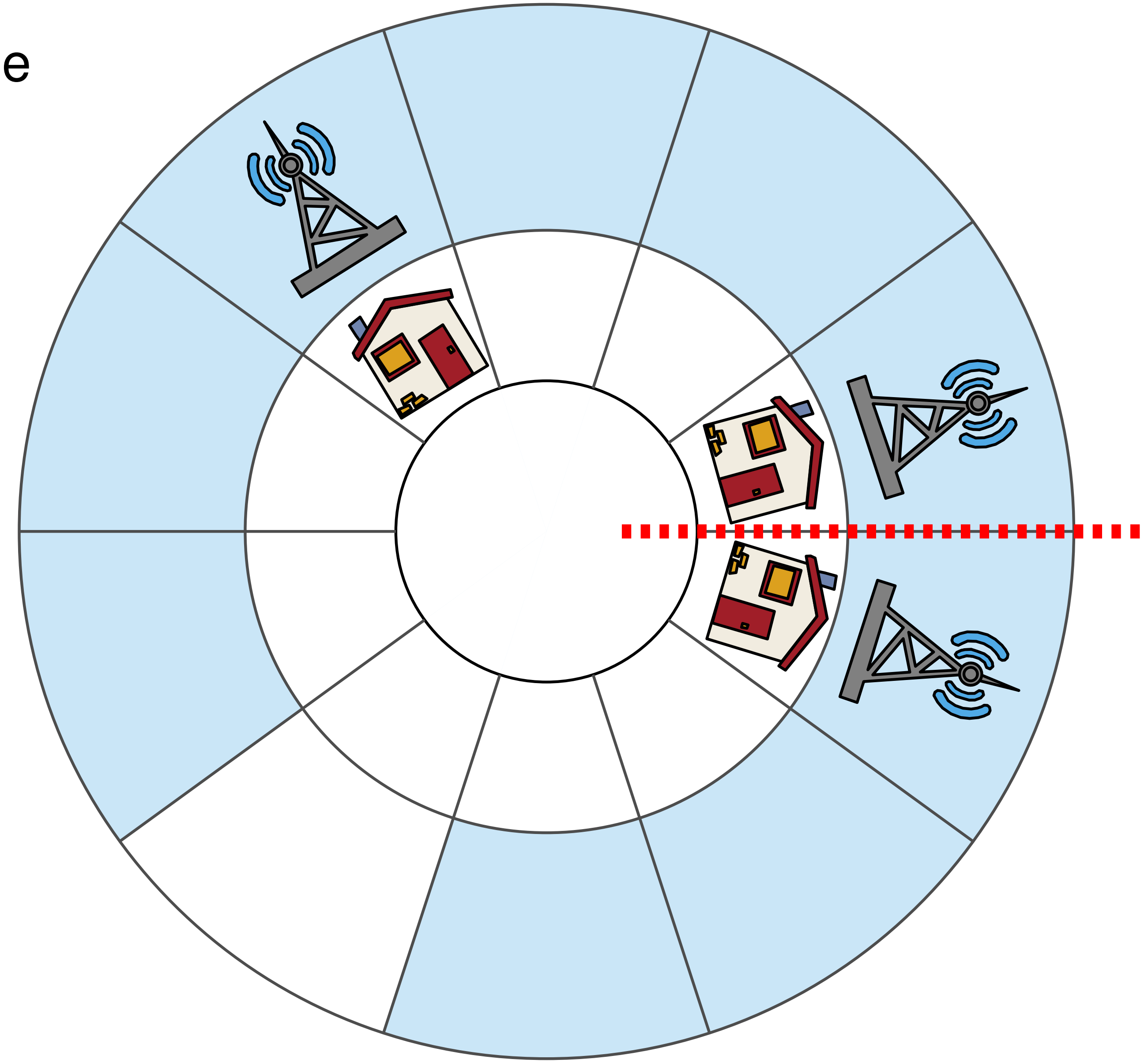
Algorithmus $\mathcal{A}'$:
schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an



Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

Algorithmus $\mathcal{A}'$:
schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an

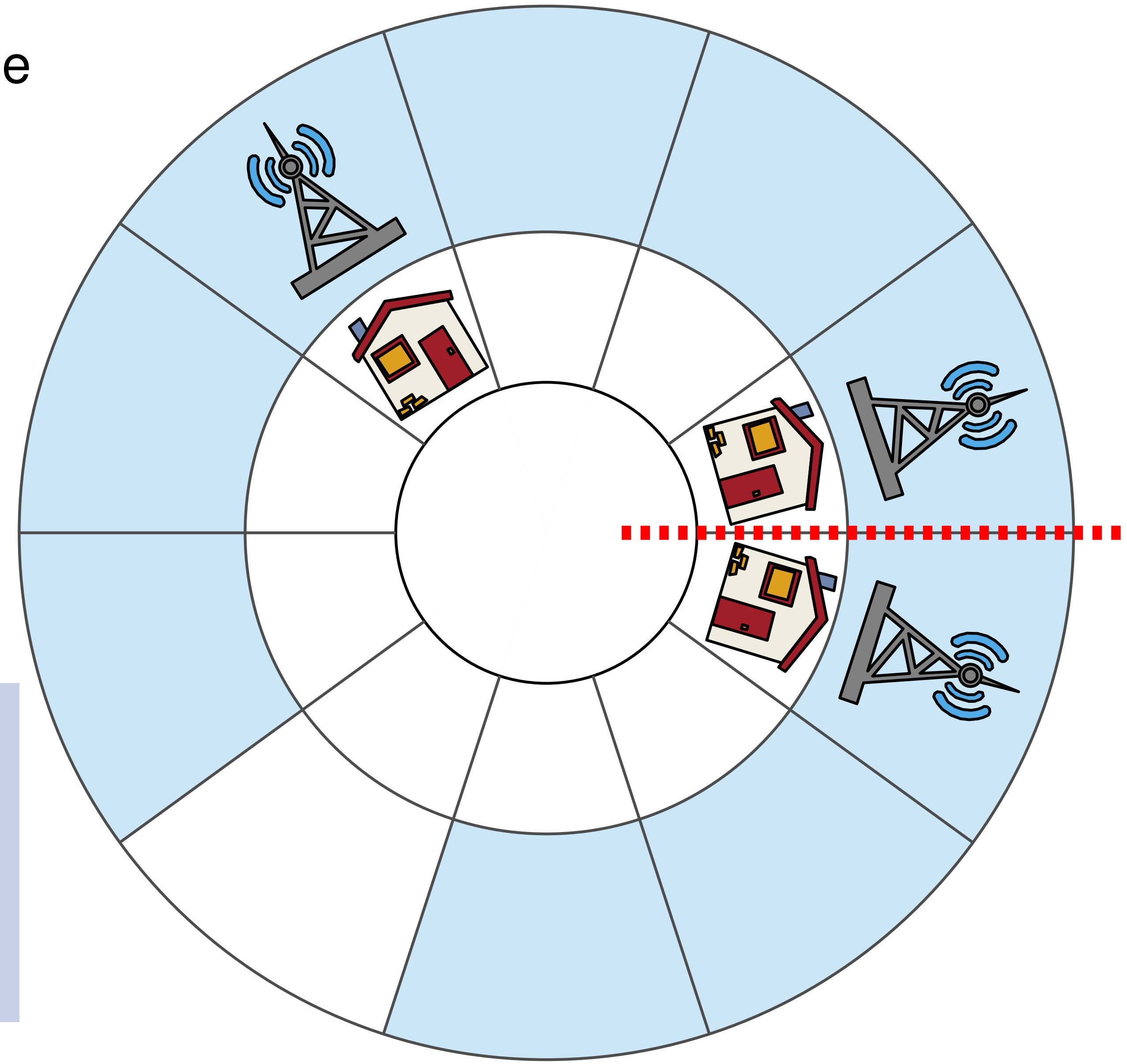


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

Algorithmus $\mathcal{A}'$: schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an



Zeigen Sie: $\mathcal{A}'$ ist ein Approximationsalgorithmus mit konstanter Gütegarantie 1

Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

Algorithmus $\mathcal{A}'$:
schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an

Zeigen Sie: $\mathcal{A}'$ ist ein Approximationsalgorithmus mit konstanter Gütegarantie 1

- gehe von einer optimalen kreisförmigen Lösung aus
- schneide diese auf

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

Algorithmus $\mathcal{A}'$:
schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an

- gehe von einer optimalen kreisförmigen Lösung aus
- schneide diese auf
- alle Häuser in Zellen $[k+1, k+2, \ldots, n-k]$ sind abgedeckt

Zeigen Sie: $\mathcal{A}'$ ist ein Approximationsalgorithmus mit konstanter Gütegarantie 1

Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

Algorithmus $\mathcal{A}'$:
schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an

- gehe von einer optimalen kreisförmigen Lösung aus
- schneide diese auf
- alle Häuser in Zellen $[k+1, k+2, \dots, n-k]$ sind abgedeckt
- nicht abgedeckte Häuser in $[1, 2, \dots, k]$ können mit einem zusätzlichen Sender abgedeckt werden

Zeigen Sie: $\mathcal{A}'$ ist ein Approximationsalgorithmus mit konstanter Gütegarantie 1

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

Algorithmus $\mathcal{A}'$:
schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an

Zeigen Sie: $\mathcal{A}'$ ist ein Approximationsalgorithmus mit konstanter Gütegarantie 1

- gehe von einer optimalen kreisförmigen Lösung aus
- schneide diese auf
- alle Häuser in Zellen $[k+1, k+2, \ldots, n-k]$ sind abgedeckt
- nicht abgedeckte Häuser in $[1, 2, \ldots, k]$ können mit einem zusätzlichen Sender abgedeckt werden
- analog für $[n-k+1, n-k+2, \ldots, n]$

Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

Algorithmus $\mathcal{A}'$: schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an

Zeigen Sie: $\mathcal{A}'$ ist ein Approximationsalgorithmus mit konstanter Gütegarantie 1

- gehe von einer optimalen kreisförmigen Lösung aus
- schneide diese auf
- alle Häuser in Zellen $[k+1, k+2, \ldots, n-k]$ sind abgedeckt
- nicht abgedeckte Häuser in $[1, 2, \ldots, k]$ können mit einem zusätzlichen Sender abgedeckt werden
- analog für $[n-k+1, n-k+2, \ldots, n]$
- nicht in beiden Randbereichen liegen nicht abgedeckte Häuser, denn diese wären in der ringförmigen Lösung nicht abgedeckt

Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

Algorithmus $\mathcal{A}'$: schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an

Zeigen Sie: $\mathcal{A}'$ ist ein Approximationsalgorithmus mit konstanter Gütegarantie 1

- gehe von einer optimalen kreisförmigen Lösung aus
- schneide diese auf
- alle Häuser in Zellen $[k+1, k+2, \dots, n-k]$ sind abgedeckt
- nicht abgedeckte Häuser in $[1, 2, \dots, k]$ können mit einem zusätzlichen Sender abgedeckt werden
- analog für $[n-k+1, n-k+2, \dots, n]$
- nicht in beiden Randbereichen liegen nicht abgedeckte Häuser, denn diese wären in der ringförmigen Lösung nicht abgedeckt
- ein zusätzlicher Sender reicht für Abdeckung von aufgeschnittener Straße

Klausuraufgabe WS16/17

Jetzt: kreisförmige Straße

Algorithmus $\mathcal{A}'$:
schneide Straße zwischen zwei beliebigen Zellen auf und wende Algorithmus $\mathcal{A}$ an

Zeigen Sie: $\mathcal{A}'$ ist ein Approximationsalgorithmus mit konstanter Gütegarantie 1

- gehe von einer optimalen kreisförmigen Lösung aus
- schneide diese auf
- alle Häuser in Zellen $[k+1, k+2, \dots, n-k]$ sind abgedeckt
- nicht abgedeckte Häuser in $[1, 2, \dots, k]$ können mit einem zusätzlichen Sender abgedeckt werden
- analog für $[n-k+1, n-k+2, \dots, n]$
- nicht in beiden Randbereichen liegen nicht abgedeckte Häuser, denn diese wären in der ringförmigen Lösung nicht abgedeckt
- ein zusätzlicher Sender reicht für Abdeckung von aufgeschnittener Straße
- $\mathcal{A}$ optimal $\Rightarrow$ $\mathcal{A}'$ ist konstante 1-Approx.

Ganzzahlige Programmierung

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

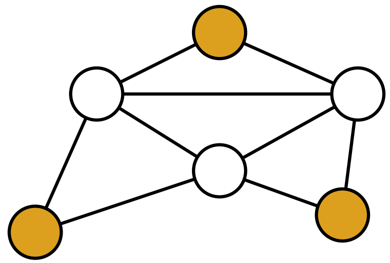
Problem GANZZAHLIGEPROGRAMMIERUNG:

$$\left.\begin{array}{ll}\underset{x}{\text{Minimiere}} & c^T x\end{array}\right\} \text{Zielfunktion}$$
$$\left.\begin{array}{ll}\text{unter} & Ax \leq b,\end{array}\right\} \text{Einschränkungen}$$
$$\left.\begin{array}{l} x \geq 0, \\ x \in \mathbb{Z}, \end{array}\right\} \text{Schranken}$$

c, b sind Vektoren, A ist Matrix

GANZZAHLIGEPROGRAMMIERUNG ist NP-schwer.

Problem UNABHÄNGIGE MENGE:
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine unabhängige Knotenmenge $V' \subseteq V$, so dass $|V'| \geq k$ gilt?
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.



Aufgabe

Problem UNABHÄNGIGE MENGE:
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Gesucht: Möglichst große unabhängige Menge $V' \subseteq V$.
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem UNABHÄNGIGE MENGE:
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Gesucht: Möglichst große unabhängige Menge $V' \subseteq V$.
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.

Variablen: Für jeden Knoten $u \in V$ eine Variable x_u

Idee: $x_u = 1$ genau dann wenn x_u gehört zu gesuchten unabhängigen Menge.

Nebenbedingungen:

Für alle $\{u, v\} \in E$: $\quad x_u + x_v \leq 1$

Für alle $u \in V$: $\quad x_u \in \{0, 1\}$

Zielfunktion: $\sum_{u \in V} x_u$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem MAX2SAT:
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Gesucht: Wahrheitsbelegung, sodass möglichst viele Klauseln erfüllt werden.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem MAX2SAT:
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Gesucht: Wahrheitsbelegung, sodass möglichst viele Klauseln erfüllt werden.

Variablen: Für jede Variable v führe die Variablen x_v und $\overline{x}_v$ ein.

Für jede Klausel c führe die Variable x_c ein.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem MAX2SAT:
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Gesucht: Wahrheitsbelegung, sodass möglichst viele Klauseln erfüllt werden.

Variablen: Für jede Variable v führe die Variablen x_v und $\overline{x}_v$ ein.

Für jede Klausel c führe die Variable x_c ein.

Nebenbedingungen:

Für alle Variablen v: $x_v + \overline{x_v} = 1$ und $x_v \in \{0, 1\}$

Für jede Klausel c: $x_c \in \{0, 1\}$

$x_c \leq x_u + x_v$ falls $c = u \vee v$

Um c zu erfüllen ($x_c = 1$) muss entweder u oder v wahr sein ($x_u = 1$ oder $x_v = 1$, also $x_u + x_v \geq 1$).

Aufgabe

Problem MAX2SAT:
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Gesucht: Wahrheitsbelegung, sodass möglichst viele Klauseln erfüllt werden.

Variablen: Für jede Variable v führe die Variablen x_v und $\overline{x}_v$ ein.
Für jede Klausel c führe die Variable x_c ein.

Nebenbedingungen:

Für alle Variablen v: $x_v + \overline{x_v} = 1$ und $x_v \in \{0, 1\}$

Für jede Klausel c: $x_c \in \{0, 1\}$

$x_c \leq x_u + x_v$ falls $c = u \vee v$
$x_c \leq \overline{x_u} + x_v$ falls $c = \overline{u} \vee v$
$x_c \leq x_u + \overline{x_v}$ falls $c = u \vee \overline{v}$
$x_c \leq \overline{x_u} + \overline{x_v}$ falls $c = \overline{u} \vee \overline{v}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Problem MAX2SAT:
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Gesucht: Wahrheitsbelegung, sodass möglichst viele Klauseln erfüllt werden.

Variablen: Für jede Variable v führe die Variablen x_v und $\overline{x}_v$ ein.

Für jede Klausel c führe die Variable x_c ein.

Nebenbedingungen:

Für alle Variablen v: $x_v + \overline{x_v} = 1$ und $x_v \in \{0, 1\}$

Für jede Klausel c: $x_c \in \{0, 1\}$

$$
\begin{aligned}
x_c &\leq x_u + x_v && \text{falls } c = u \vee v \\
x_c &\leq \overline{x_u} + x_v && \text{falls } c = \overline{u} \vee v \\
x_c &\leq x_u + \overline{x_v} && \text{falls } c = u \vee \overline{v} \\
x_c &\leq \overline{x_u} + \overline{x_v} && \text{falls } c = \overline{u} \vee \overline{v}
\end{aligned}
$$

Zielfunktion: $\sum_{\text{Klausel } c} x_c$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik