



Technische
Universität
Braunschweig



Algorithmen und Datenstrukturen 2 – Übung #4

Greedy_k, Vertex Cover

Matthias Konitzny

15.06.2022

Heute

- Greedy_k
 - Beispiel
 - Worst-Case Instanz
- Vertex Cover
 - Definition
 - Approximation



Algorithmus GREEDY_k

1. Teste jede Teilmenge \bar{S} mit $|\bar{S}| \leq k$
2. Fülle \bar{S} mit Greedy₀ auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

→ $O(n^k)$ zu testende Teilmengen

→ $O(n)$ Zeit pro Teilmenge

→ $O(1)$

→ Insgesamt $O(n^{k+1})$

Warum nicht $O(n \log n)$?

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$$Z = 30, k = 2$$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$$Z = 30, k = 2$$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$$Z = 30, k = 2$$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$$Z = 30, k = 2$$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$$Z = 30, k = 2$$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$$Z = 30, k = 2$$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4	$Z = 30, k = 2$
$z_i = p_i$	13	11	10	8	

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}
{1,4}	21	9	21	24	{1,2}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}
{1,4}	21	9	21	24	{1,2}
{2,3}	21	9	29	29	{2,3,4}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}
{1,4}	21	9	21	24	{1,2}
{2,3}	21	9	29	29	{2,3,4}
{2,4}	19	11	29	29	{2,3,4}

Greedy_k – Beispiel

i	1	2	3	4
$z_i = p_i$	13	11	10	8

$Z = 30, k = 2$

\bar{S} : Fixierte Menge

$\sum_{i \in \bar{S}} z_i$: Gewicht der fixierten Menge

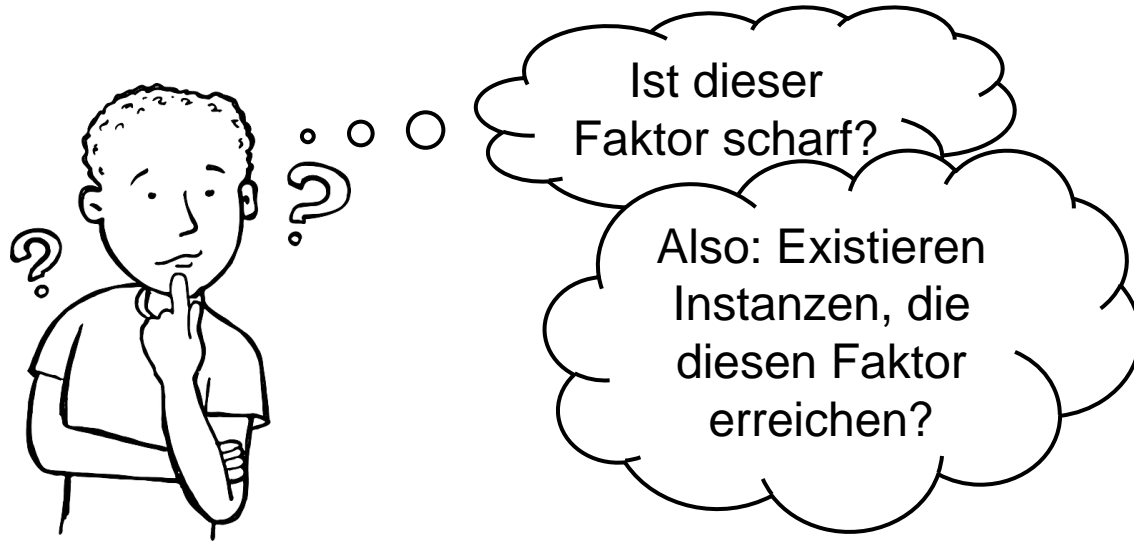
$Z - \sum_{i \in \bar{S}} z_i$: Restkapazität

$G + \sum_{i \in \bar{S}} p_i$: Wert der fixierten Menge + auffüllen

G_k, S : Bisher bester Lösungswert und Menge

\bar{S}	$\sum_{i \in \bar{S}} z_i$	$Z - \sum_{i \in \bar{S}} z_i$	$G + \sum_{i \in \bar{S}} p_i$	G_k	S
\emptyset	0	30	24	24	{1,2}
{1}	13	17	24	24	{1,2}
{2}	11	19	24	24	{1,2}
{3}	10	20	23	24	{1,2}
{4}	8	22	21	24	{1,2}
{1,2}	24	6	24	24	{1,2}
{1,3}	23	7	23	24	{1,2}
{1,4}	21	9	21	24	{1,2}
{2,3}	21	9	29	29	{2,3,4}
{2,4}	19	11	29	29	{2,3,4}
{3,4}	18	12	29	29	{2,3,4}

Satz: Greedy_k ist eine $\left(1 - \frac{1}{k+1}\right)$ -Approximation



Greedy_k

Satz: Der Approximationsfaktor $\left(1 - \frac{1}{k+1}\right)$ für Greedy_k ist scharf.

Beweis

Für ein $N \in \mathbb{N}$ wählen wir $Z = N(k + 1)$, sowie

$p_1 = \dots = p_{k+1} = z_1 = \dots = z_{k+1} = N$ und $p_{k+2} = 2, z_{k+2} = 1$.

Lösung von Greedy_k

1. Falls $k + 2 \in \bar{S}$: Es kommen k Objekte mit $p_i = z_i = N$ hinzu. \Rightarrow Wert $kN + 2$
2. Falls $k + 2 \notin \bar{S}$:
 1. Es sind $\leq k$ Objekte mit $p_i = z_i = N$ fixiert.
 2. Greedy₀ packt Objekt $k + 2$ hinzu und füllt mit anderen auf. \Rightarrow Wert $kN + 2$

Greedy_k liefert Wert $kN + 2$

Greedy_k

Satz: Der Approximationsfaktor $\left(1 - \frac{1}{k+1}\right)$ für Greedy_k ist scharf.

Beweis

Für ein $N \in \mathbb{N}$ wählen wir $Z = N(k + 1)$, sowie
 $p_1 = \dots = p_{k+1} = z_1 = \dots = z_{k+1} = N$ und $p_{k+2} = 2, z_{k+2} = 1$.

Greedy_k liefert Wert $kN + 2$

Lösung von Opt

Nimm Objekte 1 bis $k + 1$ auf. \Rightarrow Wert $N(k + 1)$

Optimum besitzt Wert $N(k + 1)$

Greedy_k

Satz: Der Approximationsfaktor $\left(1 - \frac{1}{k+1}\right)$ für Greedy_k ist scharf.

Beweis

Für ein $N \in \mathbb{N}$ wählen wir $Z = N(k + 1)$, sowie

$p_1 = \dots = p_{k+1} = z_1 = \dots = z_{k+1} = N$ und $p_{k+2} = 2, z_{k+2} = 1$.

Greedy_k liefert Wert $kN + 2$

Optimum besitzt Wert $N(k + 1)$

Damit ist:

$$\frac{ALG}{OPT} = \frac{kN + 2}{N(k + 1)} = \frac{kN}{N(k + 1)} + \frac{2}{N(k + 1)} = 1 - \frac{1}{k + 1} + \underbrace{\frac{2}{N(k + 1)}}_{\rightarrow 0 \text{ für } N \rightarrow \infty}$$

Fragen?

Vertex Cover

Vertex Cover

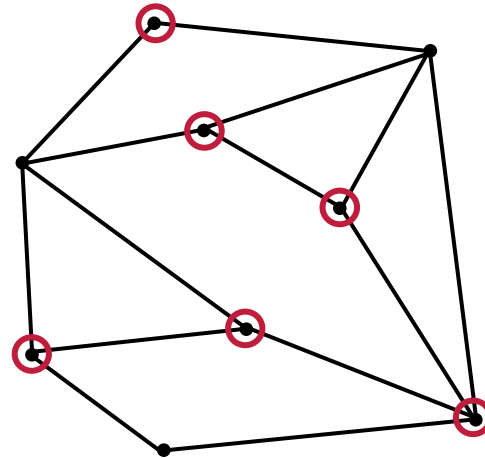
Gegeben

- Graph $G = (V, E)$
- Zahl $k \in \mathbb{N}$

Frage

- Existiert eine Menge $VC \subseteq V$ mit $|VC| \leq k$, sodass für jede Kante $\{u, v\} \in E$ gilt:
 $u \in VC$ oder $v \in VC$?

Als Optimierungsproblem: Such die kleinste Zahl k .



Vertex Cover

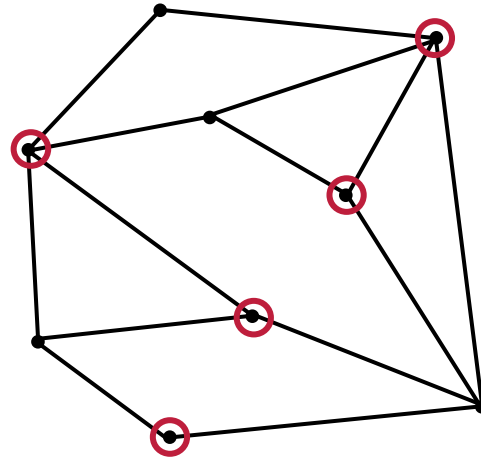
Gegeben

- Graph $G = (V, E)$
- Zahl $k \in \mathbb{N}$

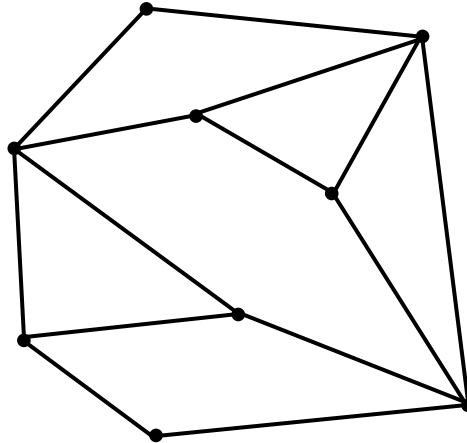
Frage

- Existiert eine Menge $VC \subseteq V$ mit $|VC| \leq k$, sodass für jede Kante $\{u, v\} \in E$ gilt:
 $u \in VC$ oder $v \in VC$?

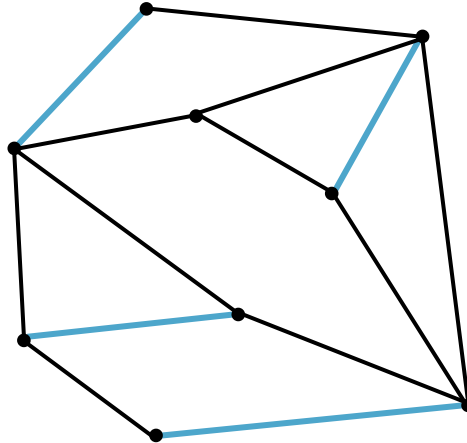
Als Optimierungsproblem: Such die kleinste Zahl k .



Schranken



Schranken

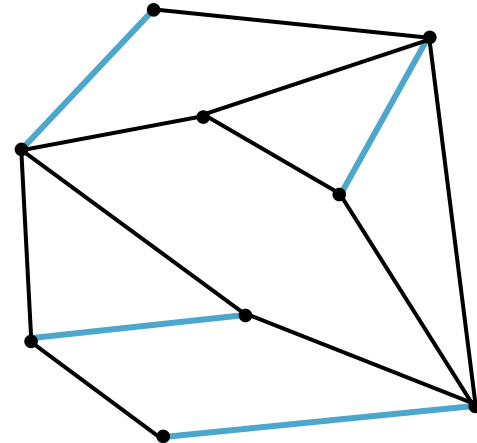


Schranken

Beobachtungen

Für jede blaue Kante benötigen wir mind. einen adjazenten Knoten in VC !

Nehmen wir beide Knoten in VC auf, erhalten wir ein Vertex Cover



Matching

Gegeben

Graph $G = (V, E)$

Gesucht

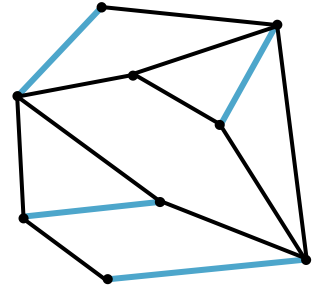
Eine Menge von Kanten, die sich paarweise keinen Knoten teilen.

Ein Matching M ist *inklusionsmaximal*, wenn es keine Kante $e \notin M$ gibt, sodass $M \cup \{e\}$ ein Matching ist.

Ein Matching M ist *kardinalitätsmaximal*, wenn es kein Matching M' gibt, sodass $|M| < |M'|$ gilt.

Engl.: maximal

Engl.: maximum



Vertex Cover und Matchings

Satz: Sei VC_{opt} ein kleinstes Vertex Cover und M ein inklusionsmaximales Matching. Dann gilt:

$$|M| \leq VC_{opt} \leq 2|M|$$

Für jede Kante $\{u, v\} \in M$ muss u oder v in VC_{opt} enthalten sein.

$$\Rightarrow |M| \leq VC_{opt}$$

Betrachte nun $VC' = \bigcup_{\{u,v\} \in M} \{u, v\}$.

Annahme: VC' ist kein Vertex Cover

Dann existiert eine Kante $\{u', v'\}$ mit $u', v' \notin VC'$.

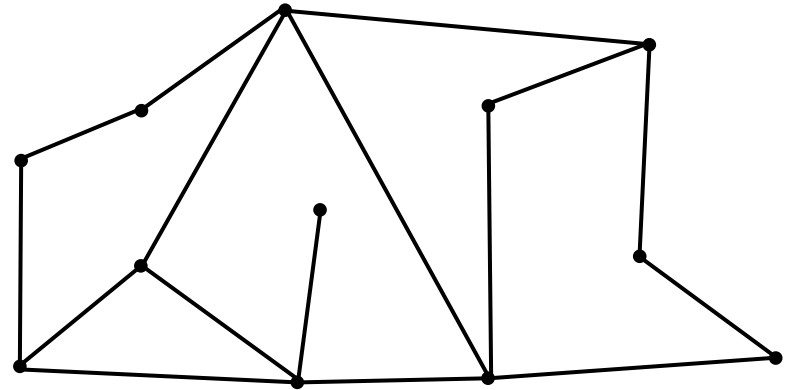
Damit kann M mit $\{u', v'\}$ erweitert werden. Widerspruch, dass M inklusionsmaximal ist.

$$\Rightarrow 2|M| \geq |VC'| \geq |VC_{opt}|$$

Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

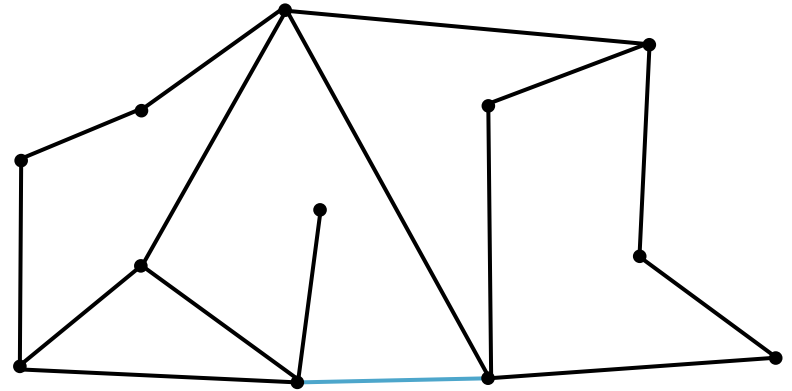
```
 $VC := \emptyset$   
for each  $\{u, v\} \in E$  do  
  if  $u \notin VC$  und  $v \notin VC$  then  
     $VC := VC \cup \{u, v\}$   
return  $VC$ 
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

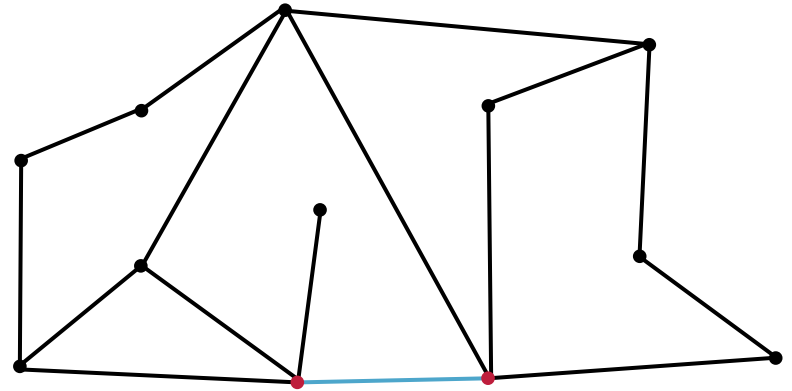
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

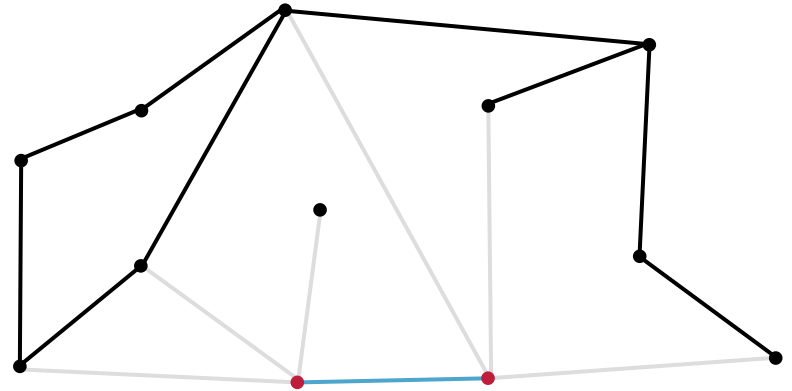
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

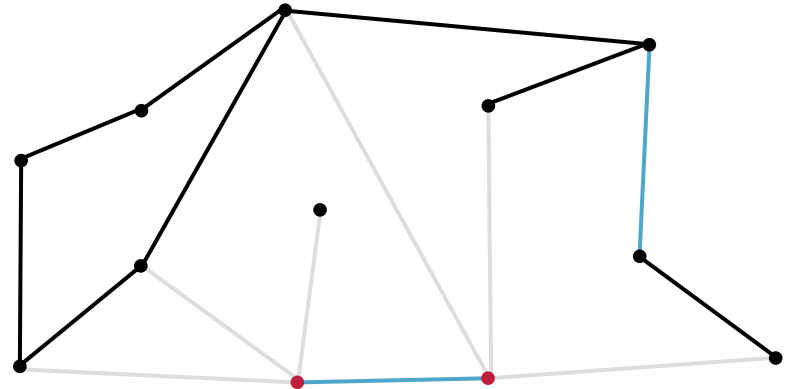
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

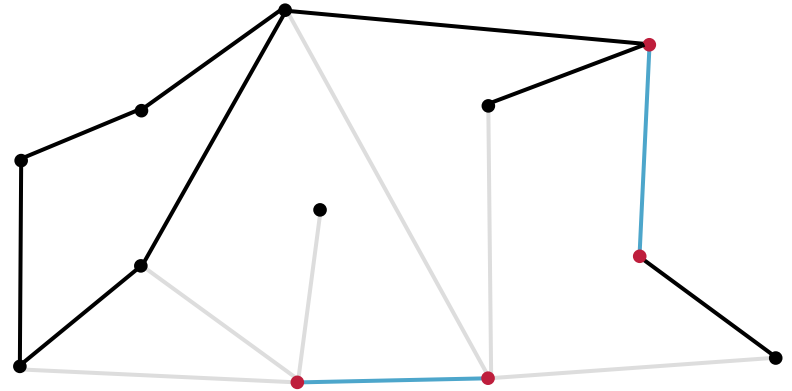
```
VC := ∅  
for each {u, v} ∈ E do  
    if u ∉ VC und v ∉ VC then  
        VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

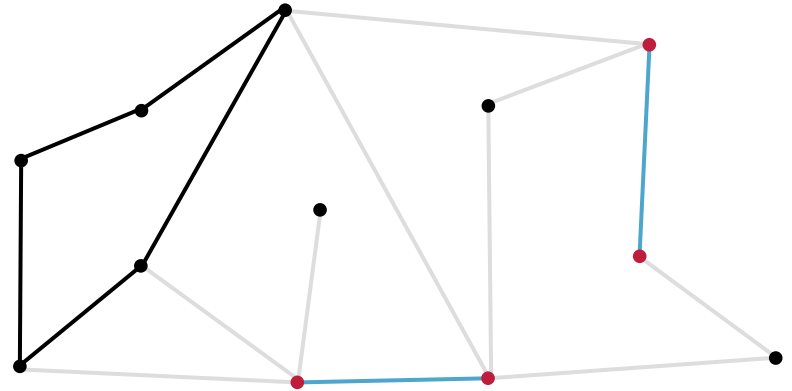
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

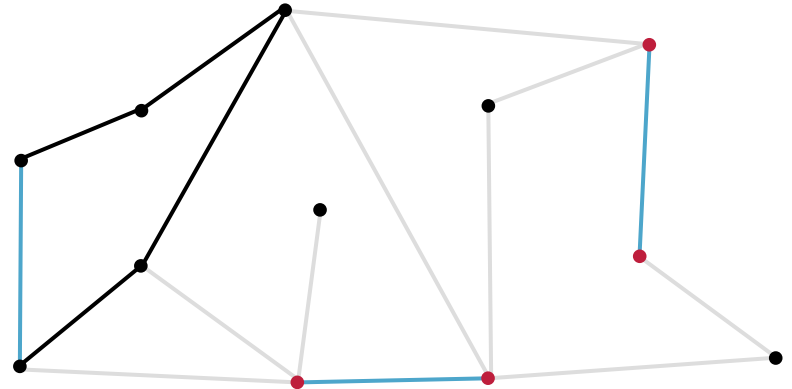
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

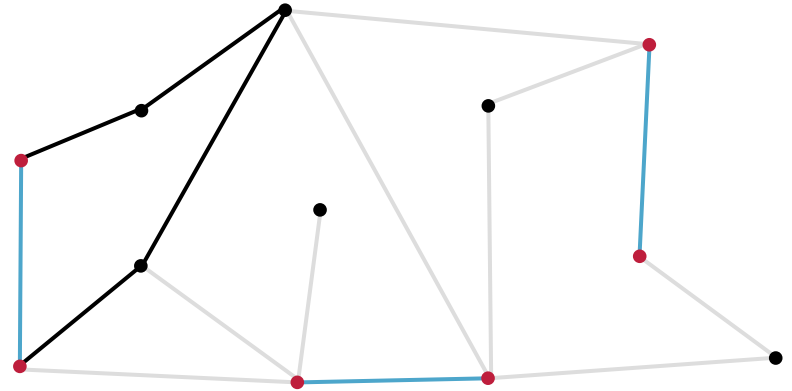
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

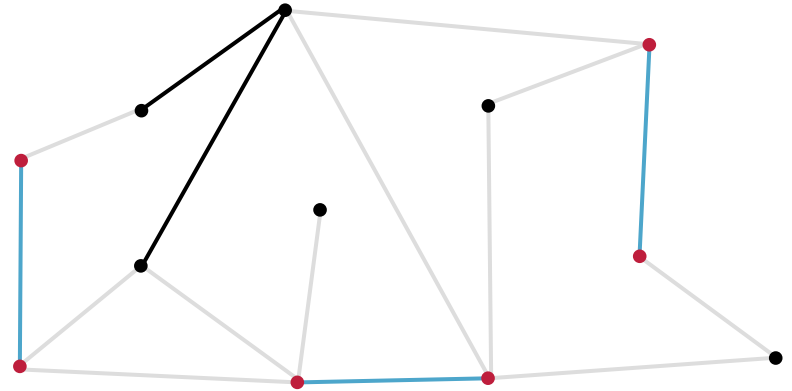
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

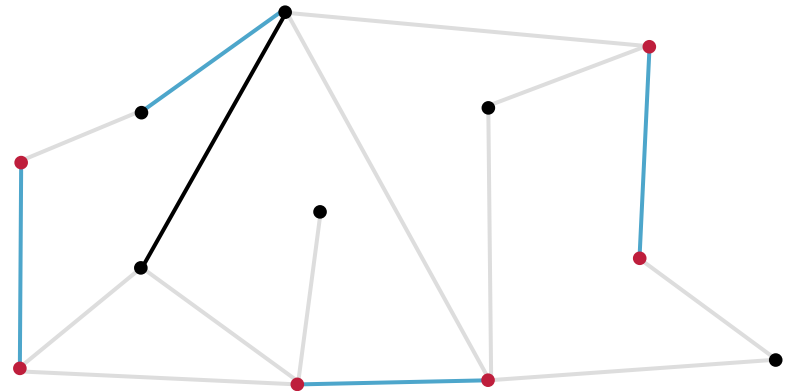
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

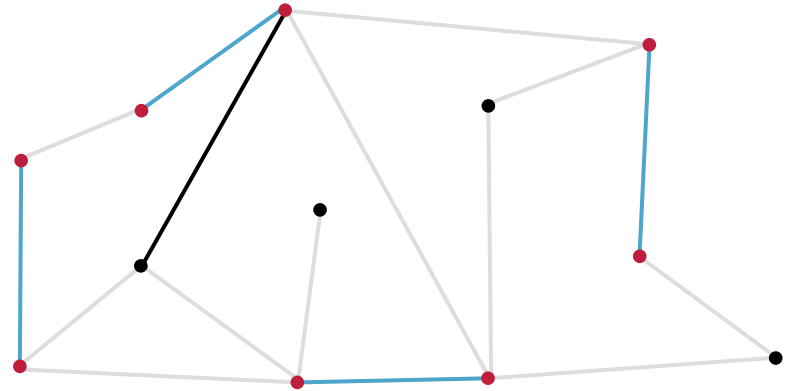
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

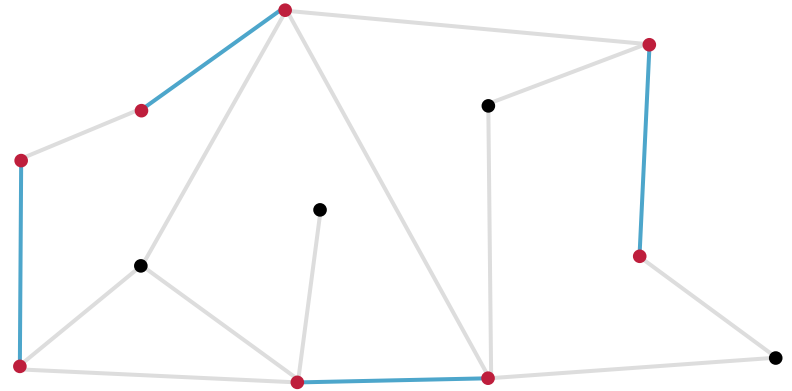
```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```



Vertex Cover und Matchings

Korollar: Es existiert eine 2-Approximation für die Optimierungsvariante von Vertex Cover.

```
VC := ∅  
for each {u, v} ∈ E do  
  if u ∉ VC und v ∉ VC then  
    VC := VC ∪ {u, v}  
return VC
```

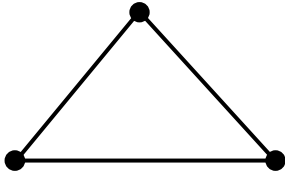


Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Welche?

Vollständige
Graphen

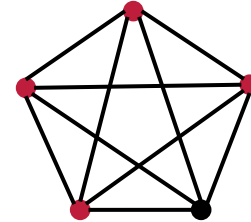
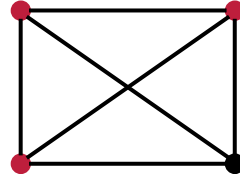
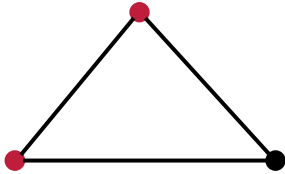


Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Welche?

Vollständige
Graphen



$$|VC| = n - 1$$

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Welche?

Vollständige
Graphen

$$|VC| = n - 1$$

Pfad
Graphen



Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Welche?

Vollständige
Graphen

$$|VC| = n - 1$$

Pfad
Graphen



$$|VC| = \left\lfloor \frac{n}{2} \right\rfloor$$

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Welche?

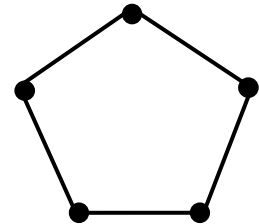
Vollständige
Graphen

$$|VC| = n - 1$$

Pfad
Graphen

$$|VC| = \left\lfloor \frac{n}{2} \right\rfloor$$

Kreis
Graphen



Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Welche?

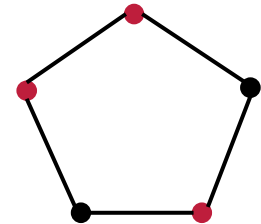
Vollständige
Graphen

$$|VC| = n - 1$$

Pfad
Graphen

$$|VC| = \left\lfloor \frac{n}{2} \right\rfloor$$

Kreis
Graphen



$$|VC| = \left\lfloor \frac{n}{2} \right\rfloor$$

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Welche?

Vollständige
Graphen

$$|VC| = n - 1$$

Pfad
Graphen

$$|VC| = \left\lfloor \frac{n}{2} \right\rfloor$$

Kreis
Graphen

$$|VC| = \left\lceil \frac{n}{2} \right\rceil$$

Baum
Graphen

$O(n)$ Algorithmus

Intervall
Graphen

$O(n \log n)$ Algorithmus

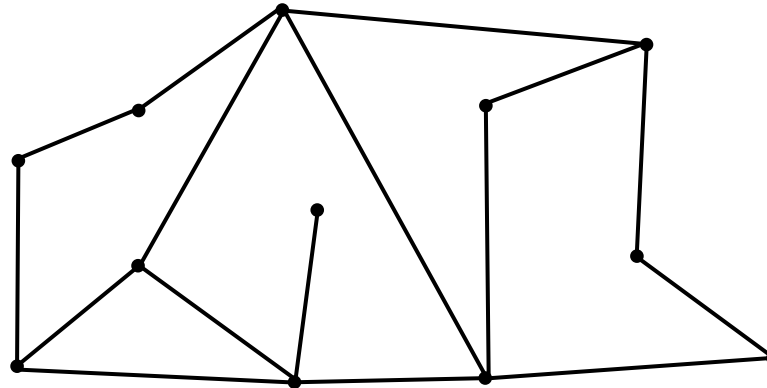
Schranken Vertex Cover

Vollständige
Graphen

Pfad
Graphen

Kreis
Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$$8 \geq |VC| \geq 2 + 1 + 3 = 6$$

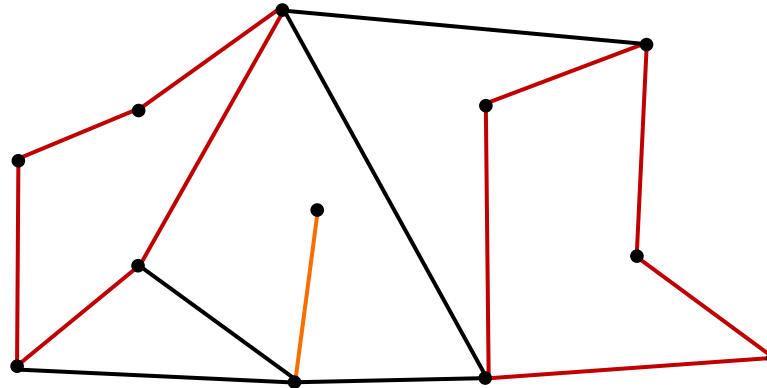
Wie groß ist ein kleinstes Vertex Cover? 6, 7 oder 8?

Schranken Vertex Cover

Vollständige
Graphen

Pfad
Graphen

Kreis
Graphen



$$8 \geq |VC| \geq 1 + 6 = 7$$

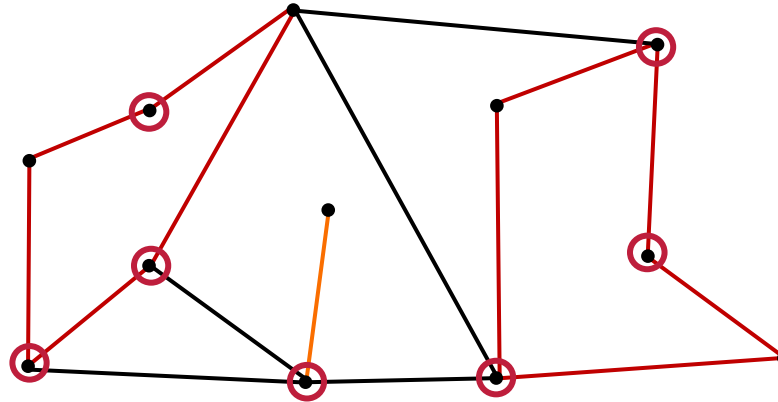
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige
Graphen

Pfad
Graphen

Kreis
Graphen



$$7 \geq |VC| \geq 1 + 6 = 7$$

Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder ~~8~~?

Fragen?