Technische Universität Braunschweig

Algorithmen und Datenstrukturen 2 – Übung #4

Approximation: Greedy$_k$, Vertex Cover

Ramin Kosfeld und Chek-Manh Loi

12.06.2024

Heute

Heute

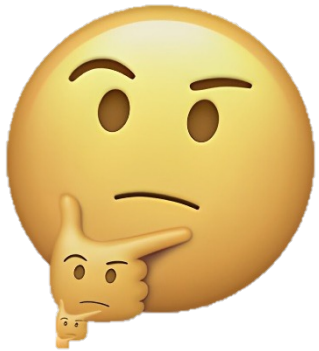
- Greedy_k
  - Beispiel
  - Worst-Case-Instanz

Heute

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

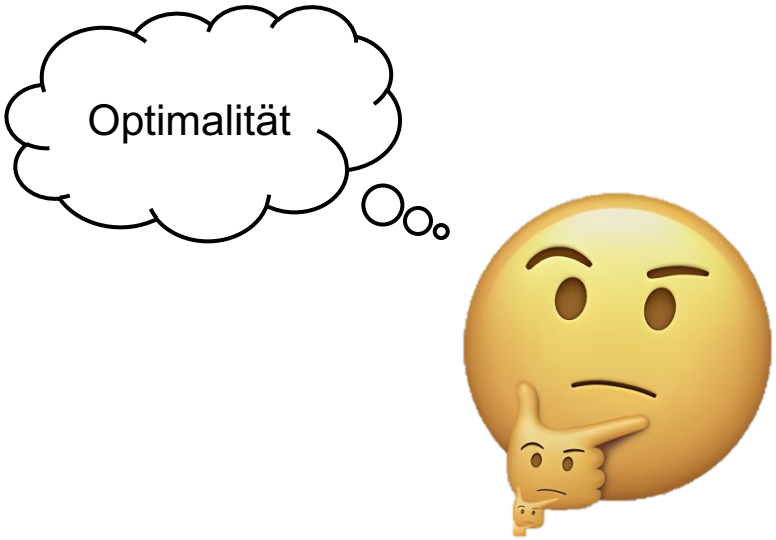
Heute

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



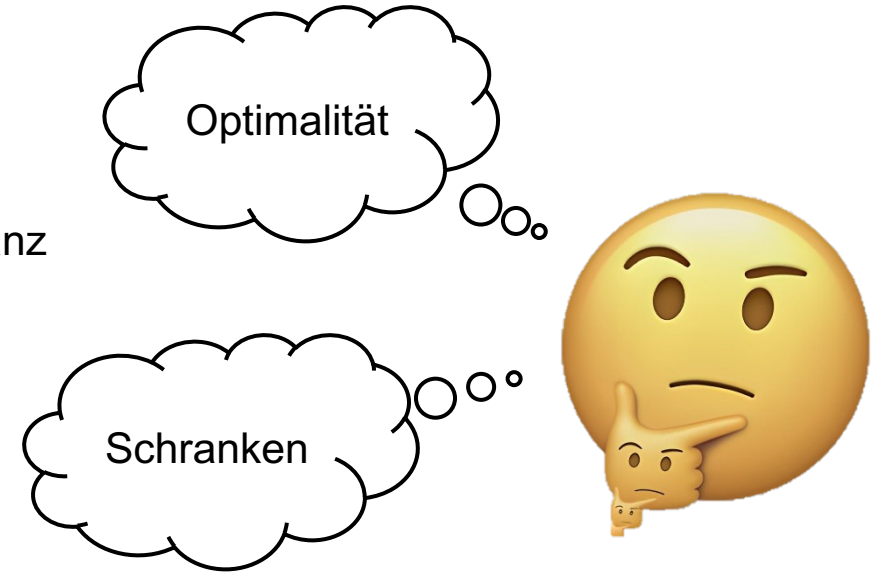
Heute

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



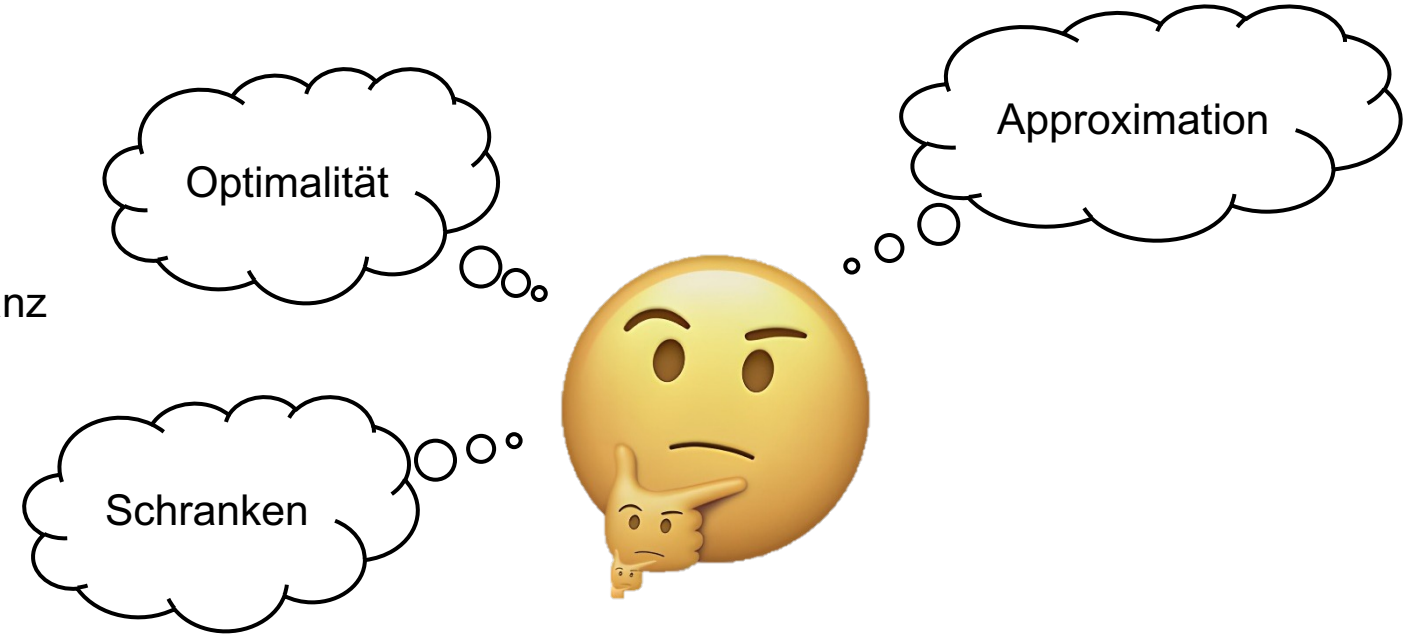
Heute

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



Heute

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



Approximationsalgorithmen

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Idee (Approximationsalgorithmen): Verlange keine Optimalität!

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Idee (Approximationsalgorithmen): Verlange keine Optimalität!

- Wir wollen aber nicht ‘irgendwas’ machen (z.B. irgendeine Greedy-Heuristik)

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Idee (Approximationsalgorithmen): Verlange keine Optimalität!

- Wir wollen aber nicht ‘irgendwas’ machen (z.B. irgendeine Greedy-Heuristik)
- Wir wollen auch im schlimmsten Fall noch ‘relativ gut’ sein

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Idee (Approximationsalgorithmen): Verlange keine Optimalität!

- Wir wollen aber nicht ‘irgendwas’ machen (z.B. irgendeine Greedy-Heuristik)
- Wir wollen auch im schlimmsten Fall noch ‘relativ gut’ sein

Was heißt ‘relativ gut’?

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Idee (Approximationsalgorithmen): Verlange keine Optimalität!

- Wir wollen aber nicht ‘irgendwas’ machen (z.B. irgendeine Greedy-Heuristik)
- Wir wollen auch im schlimmsten Fall noch ‘relativ gut’ sein

Was heißt ‘relativ gut’?

- Konstanter Approximationsfaktor c (nicht von der Eingabe abhängig)

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Idee (Approximationsalgorithmen): Verlange keine Optimalität!

- Wir wollen aber nicht ‘irgendwas’ machen (z.B. irgendeine Greedy-Heuristik)
- Wir wollen auch im schlimmsten Fall noch ‘relativ gut’ sein

Was heißt ‘relativ gut’?

- Konstanter Approximationsfaktor c (nicht von der Eingabe abhängig)
- Selbst auf der schlimmsten Instanz nicht weiter vom Optimum entfernt als Faktor c

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Idee (Approximationsalgorithmen): Verlange keine Optimalität!

- Wir wollen aber nicht ‘irgendwas’ machen (z.B. irgendeine Greedy-Heuristik)
- Wir wollen auch im schlimmsten Fall noch ‘relativ gut’ sein

Was heißt ‘relativ gut’?

- Konstanter Approximationsfaktor c (nicht von der Eingabe abhängig)
- Selbst auf der schlimmsten Instanz nicht weiter vom Optimum entfernt als Faktor c
- D.h. für Maximierungsprobleme $ALG(I) \geq c \cdot OPT(I)$ für alle Instanzen I

Approximationsalgorithmen

Wir haben ein Problem, das wir nicht immer effizient optimal lösen können

Idee (Approximationsalgorithmen): Verlange keine Optimalität!

- Wir wollen aber nicht ‘irgendwas’ machen (z.B. irgendeine Greedy-Heuristik)
- Wir wollen auch im schlimmsten Fall noch ‘relativ gut’ sein

Was heißt ‘relativ gut’?

- Konstanter Approximationsfaktor c (nicht von der Eingabe abhängig)
- Selbst auf der schlimmsten Instanz nicht weiter vom Optimum entfernt als Faktor c
- D.h. für Maximierungsprobleme $ALG(I) \geq c \cdot OPT(I)$ für alle Instanzen I
- Polynomielle Laufzeit ($O(n^k)$ für ein nicht von der Eingabe abhängiges k)

Entwurf eines Approximationsalgorithmus

Entwurf eines Approximationsalgorithmus

Was müssen wir bei Entwurf/Beweis eines Approximationsalgorithmus bedenken?

Entwurf eines Approximationsalgorithmus

Was müssen wir bei Entwurf/Beweis eines Approximationsalgorithmus bedenken?

- **Korrektheit**
  - Wie sonst auch beim Algorithmenentwurf
  - Grob gesagt: Wir finden immer korrekte Lösungen
  - Ist oft relativ einfach zu argumentieren

Entwurf eines Approximationsalgorithmus

Was müssen wir bei Entwurf/Beweis eines Approximationsalgorithmus bedenken?

- **Korrektheit**
  - Wie sonst auch beim Algorithmenentwurf
  - Grob gesagt: Wir finden immer korrekte Lösungen
  - Ist oft relativ einfach zu argumentieren
- **Polynomielle Laufzeit** $(O(n^k))$
  - Ist oft auch nicht zu schwer

Entwurf eines Approximationsalgorithmus

Was müssen wir bei Entwurf/Beweis eines Approximationsalgorithmus bedenken?

- **Korrektheit**
  - Wie sonst auch beim Algorithmenentwurf
  - Grob gesagt: Wir finden immer korrekte Lösungen
  - Ist oft relativ einfach zu argumentieren
- **Polynomielle Laufzeit** $(O(n^k))$
  - Ist oft auch nicht zu schwer
- **Approximationsfaktor**
  - Das ist in der Regel der schwierige Teil

Entwurf eines Approximationsalgorithmus

Was müssen wir bei Entwurf/Beweis eines Approximationsalgorithmus bedenken?

- **Korrektheit**
  - Wie sonst auch beim Algorithmenentwurf
  - Grob gesagt: Wir finden immer korrekte Lösungen
  - Ist oft relativ einfach zu argumentieren
- **Polynomielle Laufzeit** $(O(n^k))$
  - Ist oft auch nicht zu schwer
- **Approximationsfaktor**
  - Das ist in der Regel der schwierige Teil
  - Braucht Beweis, dass der Algorithmus *auf jeder Instanz* den Faktor einhält
  - Dabei benutzt man oft Abschätzungen und arbeitet mit Ungleichungen

Entwurf eines Approximationsalgorithmus

Was müssen wir bei Entwurf/Beweis eines Approximationsalgorithmus bedenken?

- **Korrektheit**
  - Wie sonst auch beim Algorithmenentwurf
  - Grob gesagt: Wir finden immer korrekte Lösungen
  - Ist oft relativ einfach zu argumentieren
- **Polynomielle Laufzeit** $(O(n^k))$
  - Ist oft auch nicht zu schwer
- **Approximationsfaktor**
  - Das ist in der Regel der schwierige Teil
  - Braucht Beweis, dass der Algorithmus *auf jeder Instanz* den Faktor einhält
  - Dabei benutzt man oft Abschätzungen und arbeitet mit Ungleichungen
  - Oft noch schwieriger, den exakten Faktor eines Algorithmus zu bestimmen

Greedy_k

Greedy$_k$

Algorithmus GREEDY$_K$

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf
3. Aktualisiere Lösung bei Bedarf

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

$\rightarrow O(n^k)$ zu testende Teilmengen

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

→ $O(n^k)$ zu testende Teilmengen

→ $O(n \log n)$ Zeit pro Teilmenge

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

→ $O(n^k)$ zu testende Teilmengen

→ $O(n)$ Zeit pro Teilmenge

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

→ $O(n^k)$ zu testende Teilmengen

→ $O(n)$ Zeit pro Teilmenge

→ $O(1)$

Greedy$_k$

Algorithmus GREEDY$_K$

1. Teste jede Teilmenge $\bar{S}$ mit $|\bar{S}| \leq k$
2. Fülle $\bar{S}$ mit Greedy$_0$ auf
3. Aktualisiere Lösung bei Bedarf

Laufzeit

→ $O(n^k)$ zu testende Teilmengen

→ $O(n)$ Zeit pro Teilmenge

→ $O(1)$

→ Insgesamt $O(n^{k+1})$

Greedy_k – Beispiel

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

$Z = 30, k = 2$

Greedy_k – Beispiel

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

$Z = 30, k = 2$

$\bar{S}$: Fixierte Menge

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

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

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

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
$z_i = p_i$	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}

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}

Greedy_k

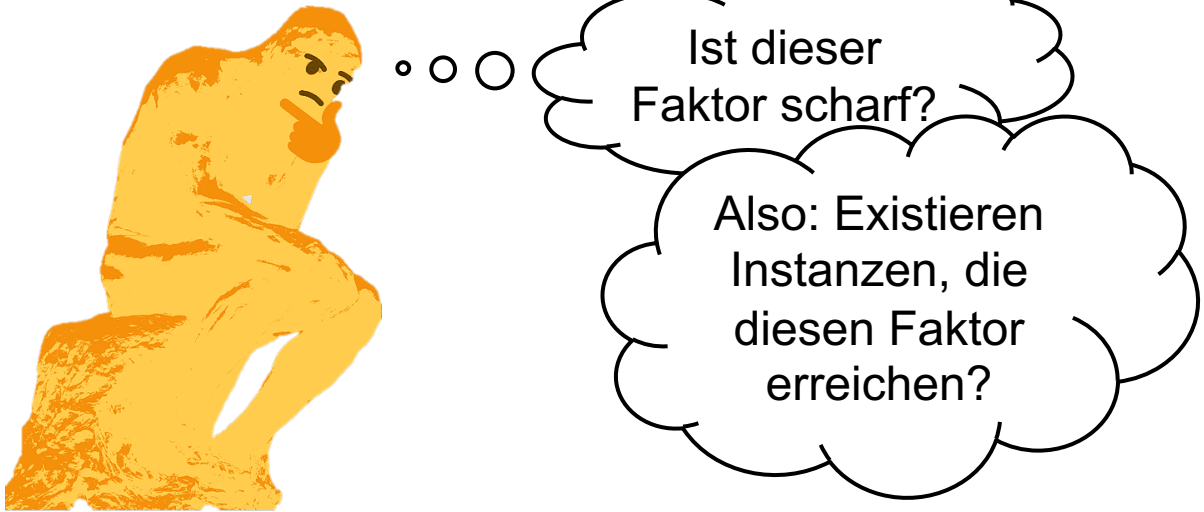
Satz: Greedy_k ist eine $\left(1 - \frac{1}{k+1}\right)$-Approximation

Greedy_k

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

Greedy_k

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



Greedy_k

Greedy_k

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

Greedy_k

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

$$\frac{k+1-1}{k+1} = \frac{k}{k+1}$$

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 $p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$, sowie $Z = N \cdot (k+1)$ und $p_{k+2} = 2,\ z_{k+2} = 1$.

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 $p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$, sowie $Z = N \cdot (k+1)$ und $p_{k+2} = 2,\ z_{k+2} = 1$.

Lösung von Greedy$_k$

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 $p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$, sowie $Z = N \cdot (k+1)$ 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. ⇒ 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 $p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$, sowie $Z = N \cdot (k+1)$ 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. ⇒ Wert $kN + 2$
2. Falls $k+2 \notin \bar{S}$:

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 $p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$, sowie $Z = N \cdot (k+1)$ 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. ⇒ Wert $kN + 2$
2. Falls $k+2 \notin \bar{S}$:
    1. Es sind $\leq k$ Objekte mit $p_i = z_i = N$ fixiert.

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 $p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$, sowie $Z = N \cdot (k+1)$ 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. ⇒ Wert $kN + 2$
2. Falls $k+2 \notin \bar{S}$:
    1. Es sind $\leq k$ Objekte mit $p_i = z_i = N$ fixiert.
    2. Greedy$_0$ packt Objekt $k+2$ hinzu und füllt mit anderen auf. ⇒ 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 $p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$, sowie $Z = N \cdot (k+1)$ 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. ⇒ Wert $kN + 2$
2. Falls $k+2 \notin \bar{S}$:
   1. Es sind $\leq k$ Objekte mit $p_i = z_i = N$ fixiert.
   2. Greedy$_0$ packt Objekt $k+2$ hinzu und füllt mit anderen auf. ⇒ 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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1.$

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1.$

Greedy$_k$ liefert Wert $kN + 2$

Optimale Lösung

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

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1.$

Greedy$_k$ liefert Wert $kN + 2$

Optimale Lösung

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

Optimum hat Wert $N \cdot (k+1)$

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1$.

Greedy$_k$ liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1$.

Greedy$_k$ liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

Damit ist:

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1.$

Greedy$_k$ liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

Damit ist:

$$\frac{ALG}{OPT} =$$

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1$.

Greedy$_k$ liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

Damit ist:

$$\frac{ALG}{OPT} = \frac{kN+2}{N \cdot (k+1)} =$$

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1$.

Greedy$_k$ liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

Damit ist:

$$\frac{ALG}{OPT} = \frac{kN+2}{N\cdot(k+1)} = \frac{kN}{N\cdot(k+1)} + \frac{2}{N\cdot(k+1)} =$$

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1$.

Greedy$_k$ liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

Damit ist:

$$\frac{ALG}{OPT} = \frac{kN+2}{N\cdot(k+1)} = \frac{kN}{N\cdot(k+1)} + \frac{2}{N\cdot(k+1)} = 1 - \frac{1}{k+1} + \frac{2}{N\cdot(k+1)}$$

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1$.

Greedy_k liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

Damit ist:

$$\frac{k}{k+1} = \frac{k+1-1}{k+1}$$

$$\frac{ALG}{OPT} = \frac{kN+2}{N\cdot(k+1)} = \frac{kN}{N\cdot(k+1)} + \frac{2}{N\cdot(k+1)} = 1 - \frac{1}{k+1} + \frac{2}{N\cdot(k+1)}$$

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 $Z = N \cdot (k+1)$, sowie
$p_1 = \cdots = p_{k+1} = z_1 = \cdots = z_{k+1} = N$ und $p_{k+2} = 2,\ z_{k+2} = 1$.

Greedy_k liefert Wert $kN + 2$

Optimum hat Wert $N \cdot (k+1)$

Damit ist:

$$\frac{k}{k+1} = \frac{k+1-1}{k+1}$$

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

Fragen?

Vertex Cover

Vertex Cover

& Matchings 

Vertex Cover

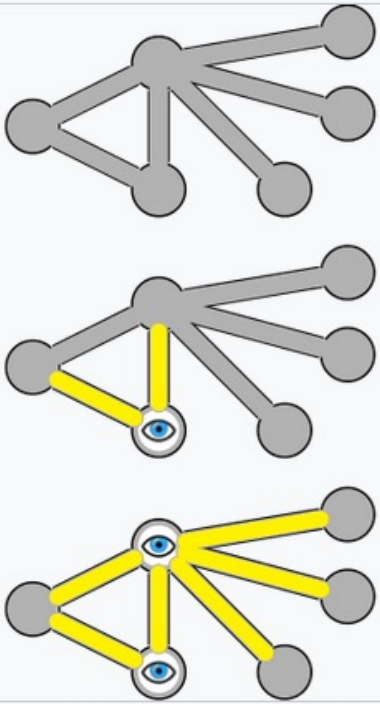
set of vertices that includes at least one

a classical optimization problem. It is
Moreover, it is hard to approximate – it
conjecture is true. On the other hand, it
NP-hard optimization problem that has
em, was one of Karp's 21 NP-complete
onal complexity theory. Furthermore, the
n parameterized complexity theory.

linear program whose dual linear

x cover in hypergraphs.

set of V such that
ore every edge has at least



Example graph that has a vertex cover comprising 2 vertices (bottom), but none with fewer.

Vertex Cover

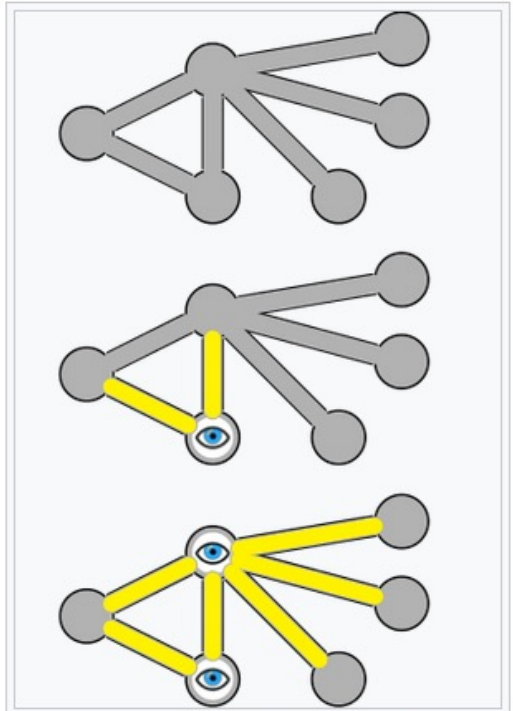
set of vertices that includes at least one

a classical optimization problem. It is
Moreover, it is hard to approximate – it
conjecture is true, the other hand, it
NP-hard optimiz lem that has
em, was one o IP-complete
nal complexi rmore, the
n paramete ory.

linear pr

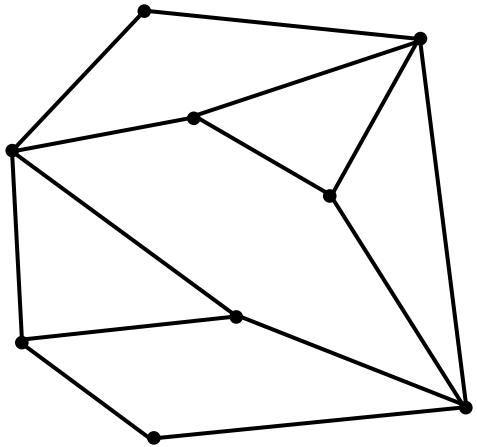
x cover in hypergraphs.

set of V such that
ore every edge has at least



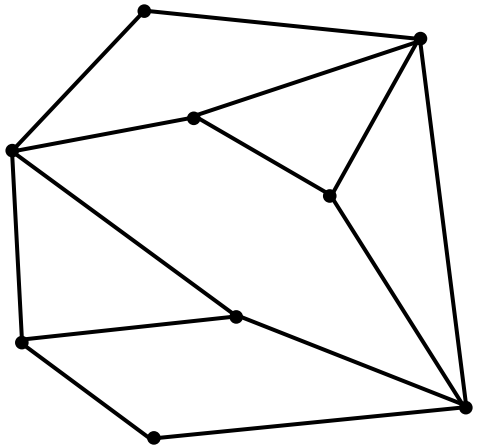
Example graph that has a vertex cover comprising 2 vertices (bottom), but none with fewer.

Vertex Cover



Vertex Cover

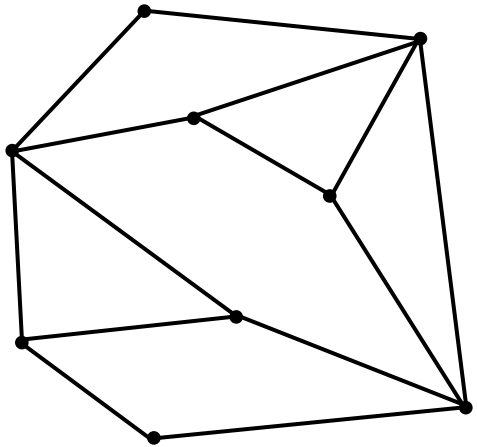
Gegeben



Vertex Cover

Gegeben

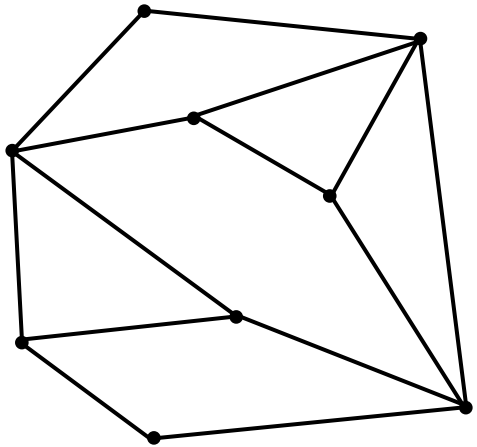
- Graph $G = (V, E)$



Vertex Cover

Gegeben

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



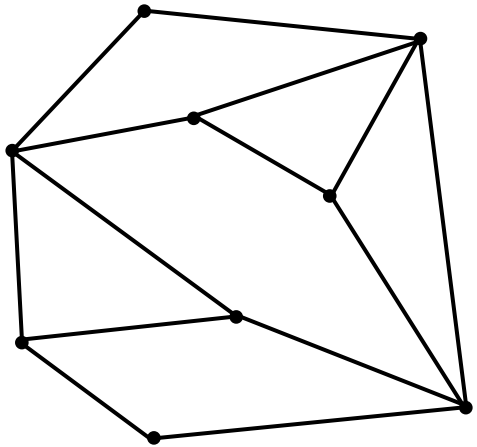
Vertex Cover

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 $v \in VC$?



Vertex Cover

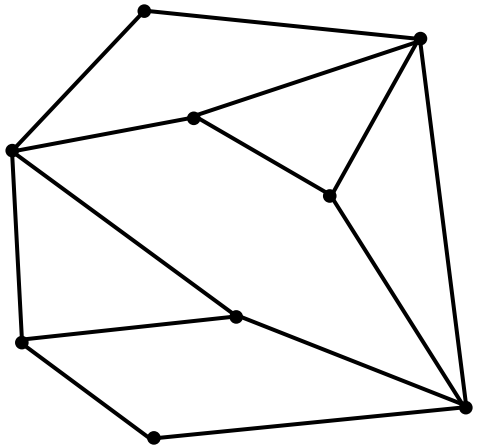
Gegeben

- 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: Suche die kleinste Zahl k.



Vertex Cover

Gegeben

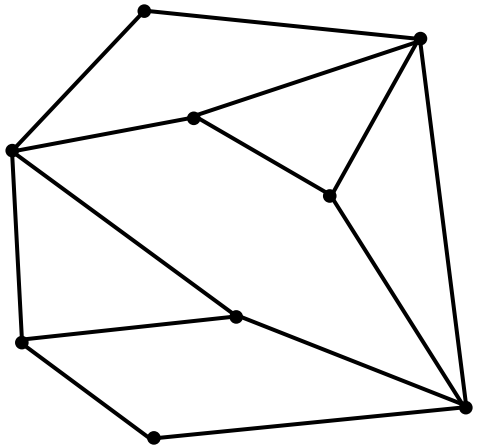
- Graph $G = (V, E)$
- Zahl $k \in \mathbb{N}$

Das ist unser Vertex-Cover

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: Suche die kleinste Zahl k.



Vertex Cover

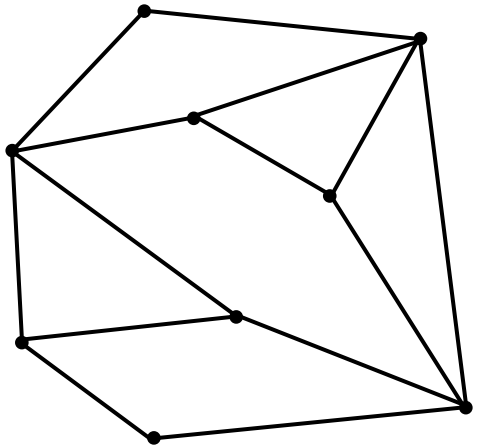
Gegeben

- 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 $v \in VC$?

Das ist unser Vertex-Cover



Als Optimierungsproblem: Suche die kleinste Zahl k.

(Kleinste) Menge an Knoten, die jede Kante abdeckt!

Vertex Cover

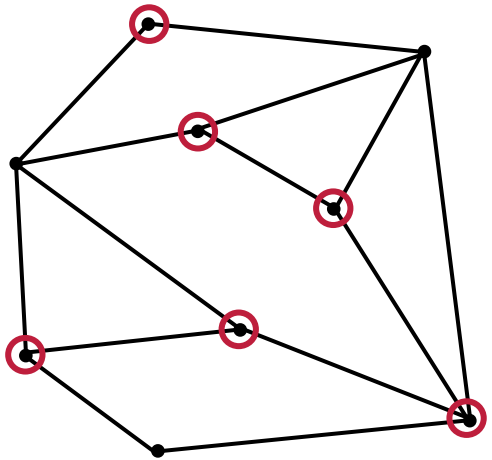
Gegeben

- 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 $v \in VC$?

Das ist unser Vertex-Cover



Als Optimierungsproblem: Suche die kleinste Zahl k.

(Kleinste) Menge an Knoten, die jede Kante abdeckt!

Vertex Cover

Gegeben

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

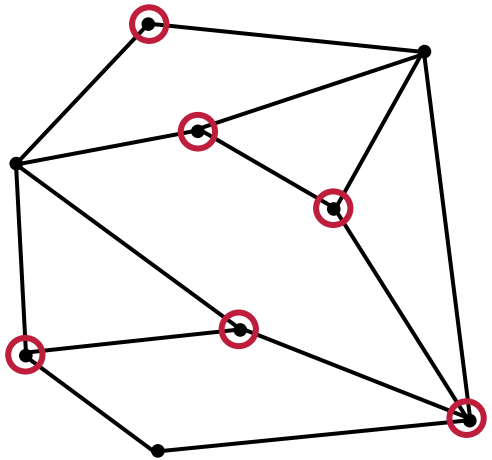
Frage

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

Das ist unser Vertex-Cover

Als Optimierungsproblem: Suche die kleinste Zahl k.

(Kleinste) Menge an Knoten, die jede Kante abdeckt!



$|VC| = 6$

Vertex Cover

Geht das noch besser?

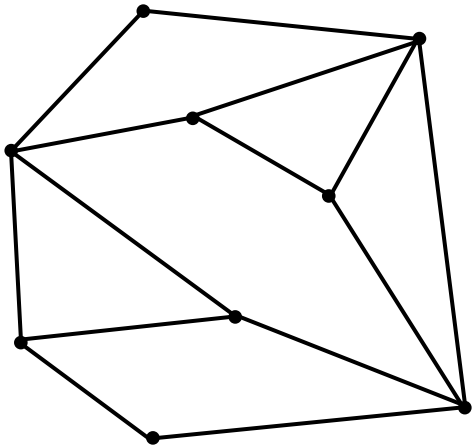
Gegeben

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

Das ist unser Vertex-Cover

Frage

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



$|VC| = 6$

Als Optimierungsproblem: Suche die kleinste Zahl k.

(Kleinste) Menge an Knoten, die jede Kante abdeckt!

Vertex Cover

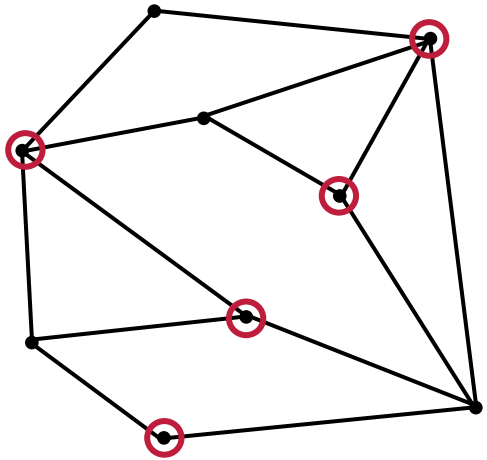
Gegeben

- 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 $v \in VC$?

Das ist unser Vertex-Cover



Als Optimierungsproblem: Suche die kleinste Zahl k.

(Kleinste) Menge an Knoten, die jede Kante abdeckt!

Vertex Cover

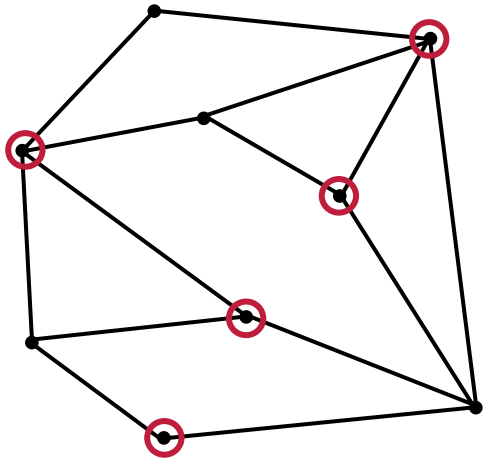
Gegeben

- 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 $v \in VC$?

Das ist unser Vertex-Cover

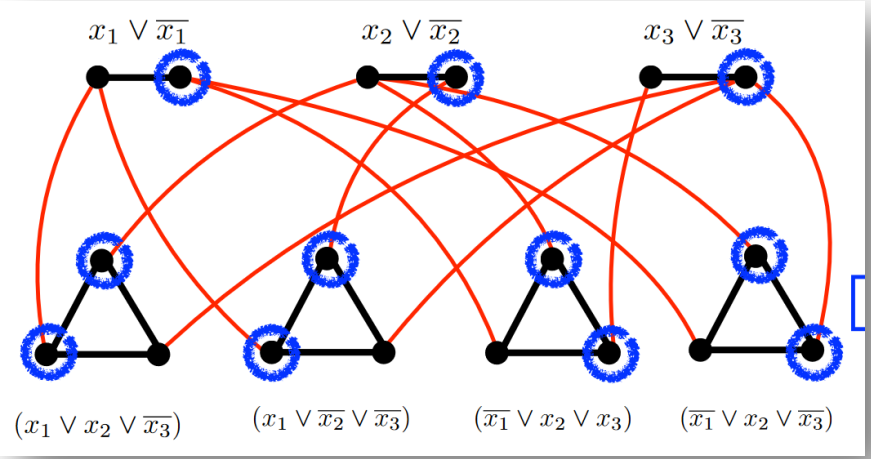


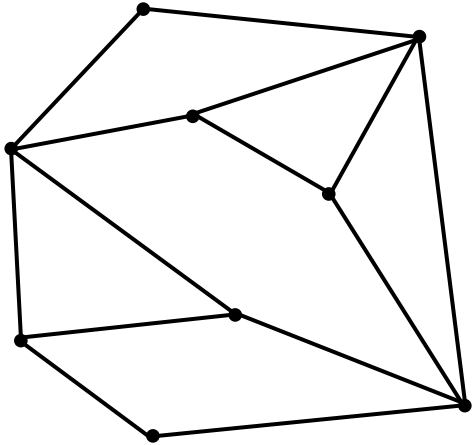
$|VC| = 5$

Als Optimierungsproblem: Suche die kleinste Zahl k.

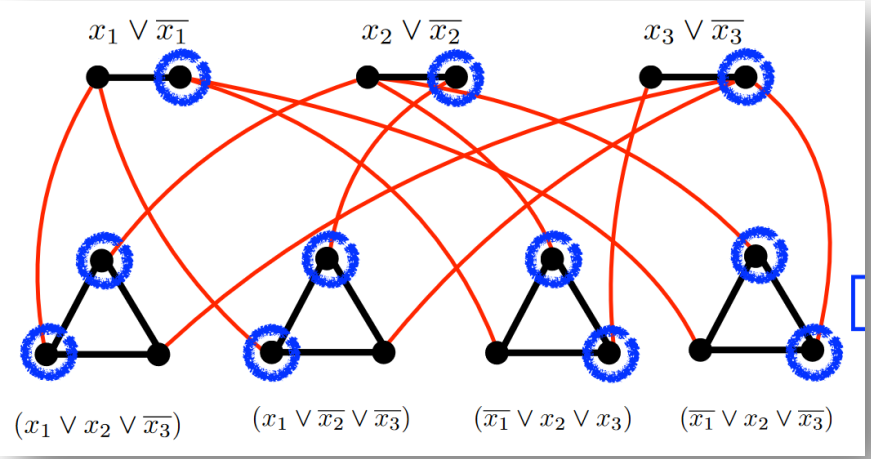
(Kleinste) Menge an Knoten, die jede Kante abdeckt!

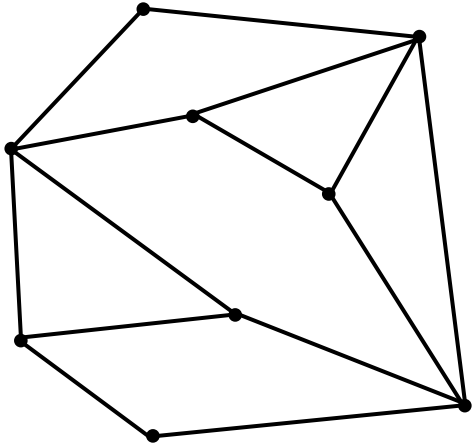
Vertex Cover





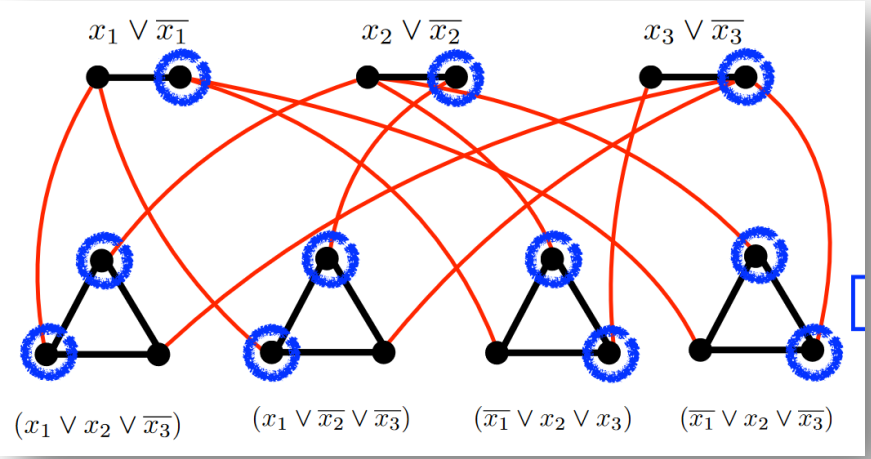
Vertex Cover

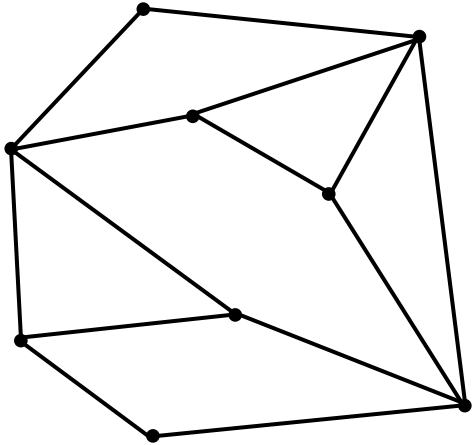




Vorlesung: Ein minimales Vertex-Cover zu finden ist NP-schwer!

Vertex Cover

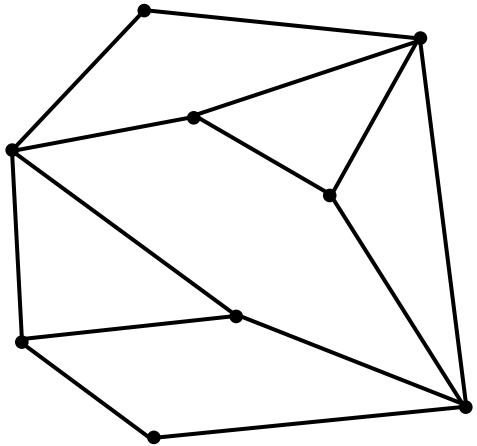




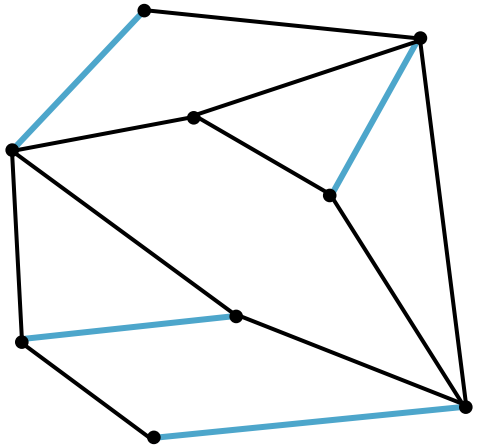
Vorlesung: Ein minimales Vertex-Cover zu finden ist NP-schwer!

→ Schranken? (Und damit dann Approximationen?)

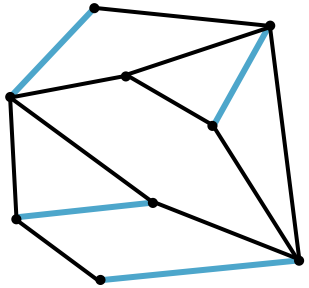
Schranken



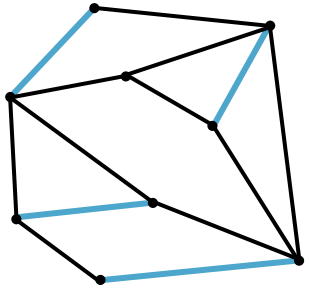
Schranken



Matching

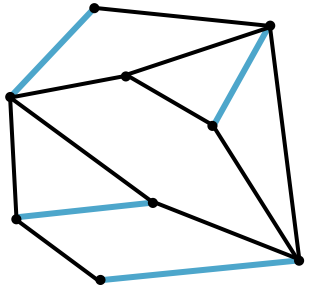


Matching



Matching

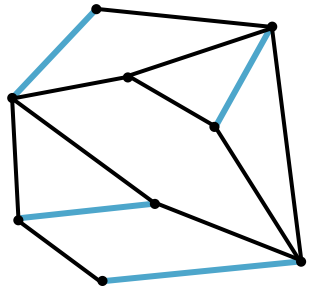
Gegeben



Matching

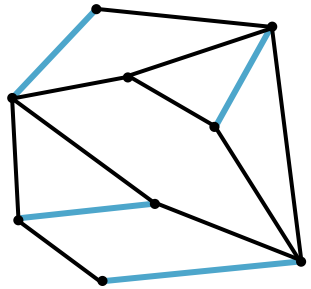
Gegeben

Graph $G = (V, E)$



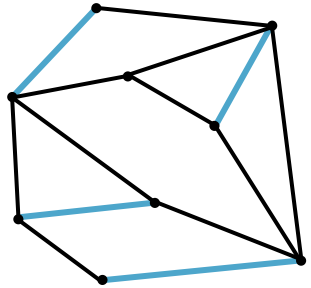
Matching

Gegeben	Gesucht
Graph $G = (V, E)$	



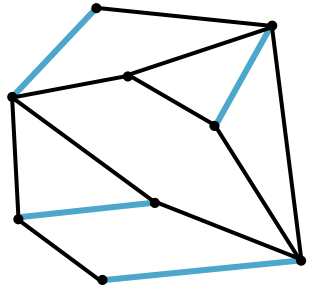
Matching

Gegeben	**Gesucht**
Graph $G = (V, E)$	Eine Menge von Kanten $M \subseteq E$, die sich paarweise keinen Knoten teilen.



Matching

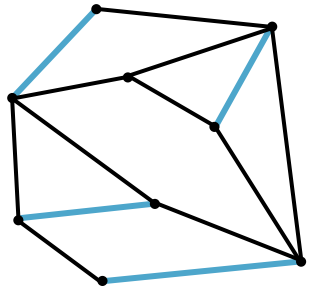
Gegeben	**Gesucht**
Graph $G = (V, E)$	Eine Menge von Kanten $M \subseteq E$, die sich paarweise keinen Knoten teilen.



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

Matching

Gegeben	**Gesucht**
Graph $G = (V, E)$	Eine Menge von Kanten $M \subseteq E$, die sich paarweise keinen Knoten teilen.

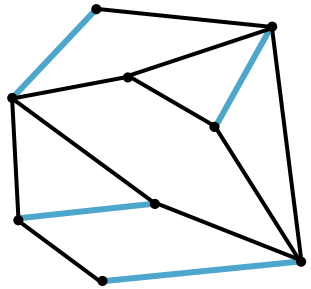


Ein Matching M ist *inklusionsmaximal*, wenn es keine Kante $e \notin M$ gibt, sodass $M \cup \{e\}$ ein Matching ist.

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

Matching

Gegeben	**Gesucht**
Graph $G = (V, E)$	Eine Menge von Kanten $M \subseteq E$, die sich paarweise keinen Knoten teilen.



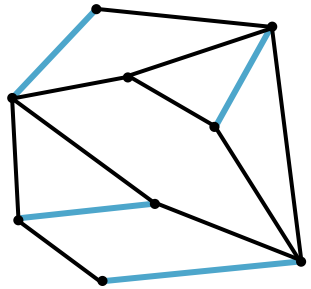
Ein Matching M ist *inklusionsmaximal*, wenn es keine Kante $e \notin M$ gibt, sodass $M \cup \{e\}$ ein Matching ist.

Engl.:
maximal matching

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

Matching

Gegeben	**Gesucht**
Graph $G = (V, E)$	Eine Menge von Kanten $M \subseteq E$, 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.

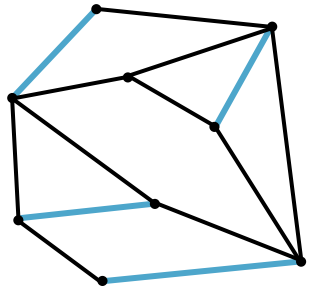
Engl.:
maximal matching

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

Engl.:
maximum matching

Matching

Gegeben	**Gesucht**
Graph $G = (V, E)$	Eine Menge von Kanten $M \subseteq E$, 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.

Engl.:
maximal matching

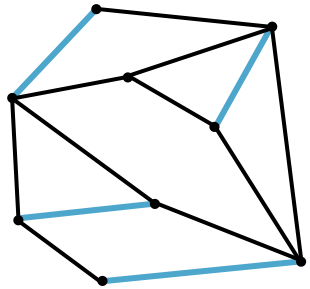
Das findet man easy!

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

Engl.:
maximum matching

Matching

Gegeben	**Gesucht**
Graph $G = (V, E)$	Eine Menge von Kanten $M \subseteq E$, 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.

Engl.: maximal matching

Das findet man easy!

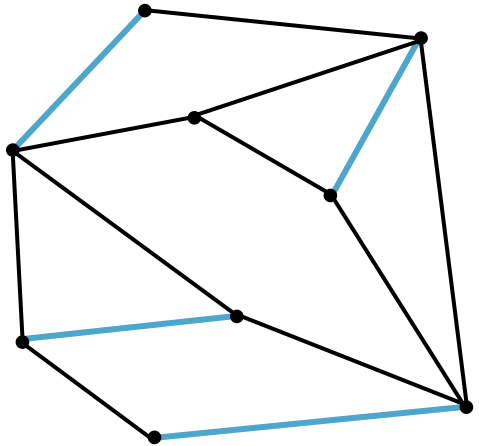
Ein Matching M ist *kardinalitätsmaximal*, wenn es kein Matching M' gibt, sodass $|M| < |M'|$ gilt.

Engl.: maximum matching

Hui …*

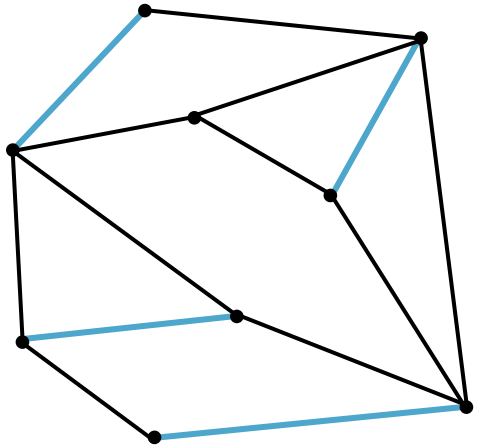
*Mehr dazu: Netzwerkalgorithmen.
Nicht NP-schwer. Aber auch nicht ganz trivial…

Matching



Matching

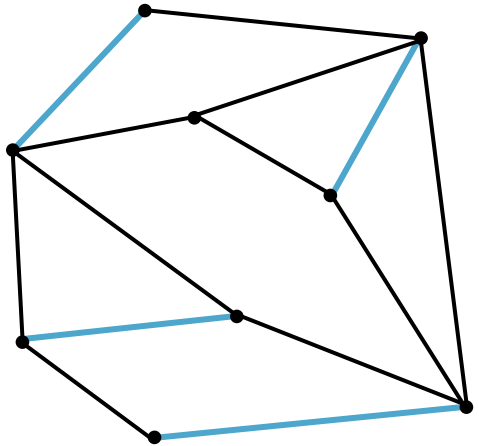
Beobachtungen



Matching

Beobachtungen

Die blauen Kanten bilden ein Matching!

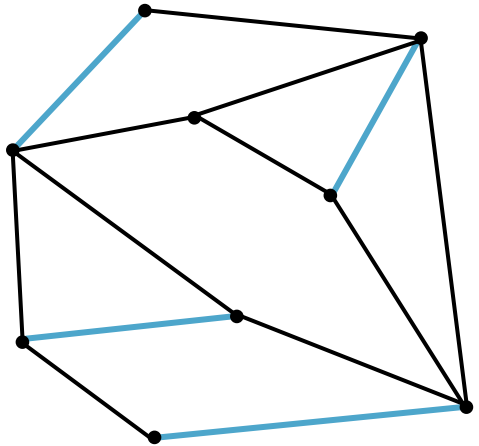


Matching

Beobachtungen

Die blauen Kanten bilden ein Matching!

… ein *kardinalitätsmaximales* Matching?

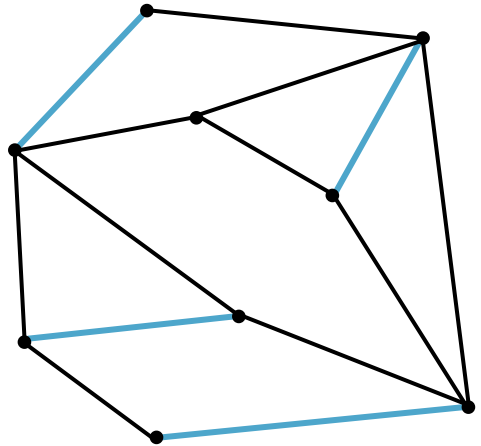


Matching

Beobachtungen

Die blauen Kanten bilden ein Matching!

… ein *kardinalitätsmaximales* Matching?
Hmm, das sehen wir nicht so einfach.



Matching

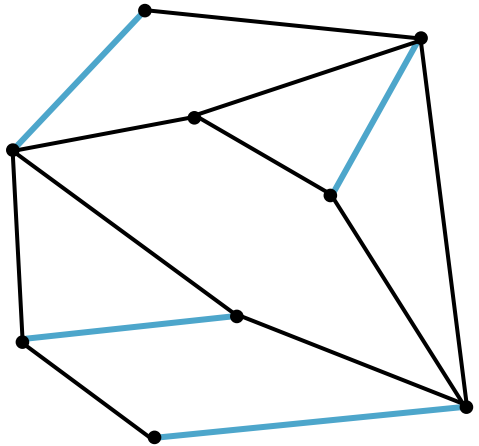
Beobachtungen

Die blauen Kanten bilden ein Matching!

… ein *kardinalitätsmaximales* Matching?
Hmm, das sehen wir nicht so einfach.

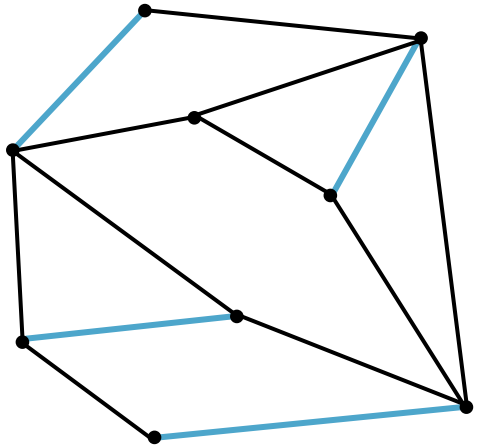
… zumindest aber ein
inklusionsmaximales Matching!

(Man kann keine Kante mehr hinzufügen!)



Matching

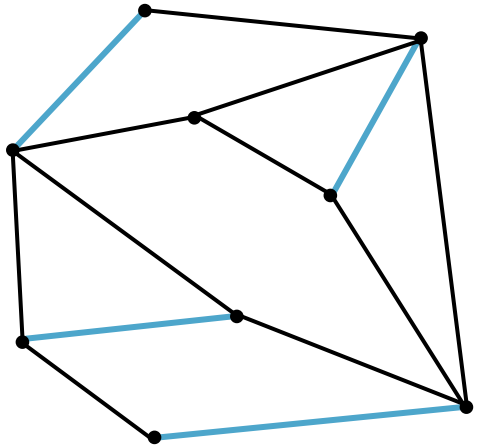
Inklusionsmaximales Matching



Matching

Eigentlich suchen wir doch ein minimales Vertex Cover!

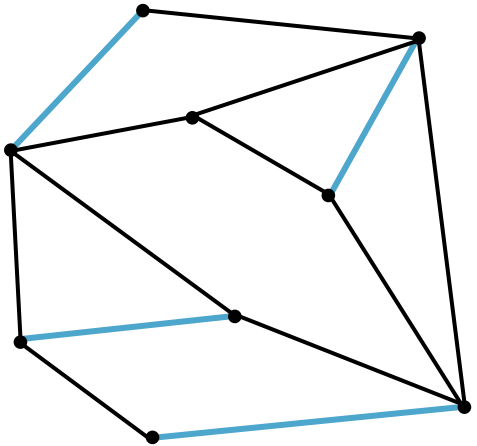
Inklusionsmaximales Matching



Matching

Eigentlich suchen wir doch ein minimales Vertex Cover!
Also: Eine minimale Knotenmenge VC, die jede Kante im Graph abdeckt

Inklusionsmaximales Matching

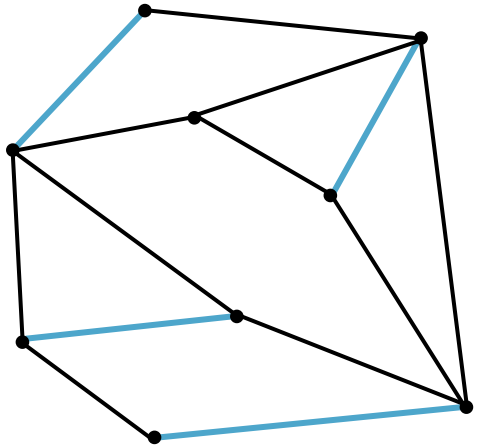


Matching

Eigentlich suchen wir doch ein minimales Vertex Cover!
Also: Eine minimale Knotenmenge VC, die jede Kante im Graph abdeckt

Zwei zentrale Beobachtungen

Inklusionsmaximales Matching



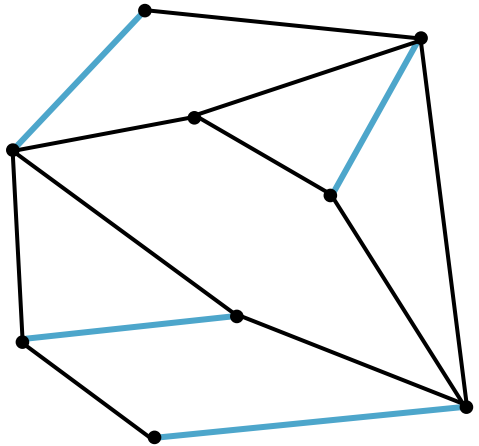
Matching

Eigentlich suchen wir doch ein minimales Vertex Cover!
Also: Eine minimale Knotenmenge VC, die jede Kante im Graph abdeckt

Zwei zentrale Beobachtungen

Für jede Kante im Matching benötigen wir mindestens einen Knoten in VC!

Inklusionsmaximales Matching



Matching

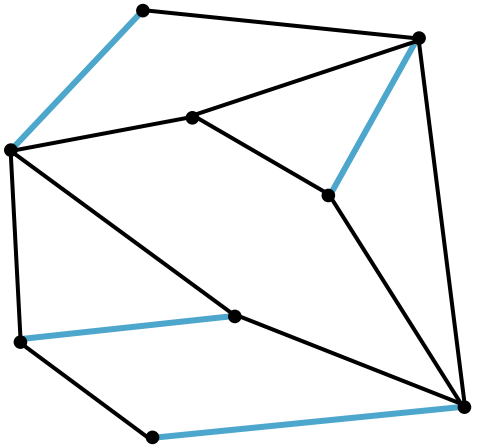
Eigentlich suchen wir doch ein minimales Vertex Cover!
Also: Eine minimale Knotenmenge VC, die jede Kante im Graph abdeckt

Zwei zentrale Beobachtungen

Für jede Kante im Matching benötigen wir mindestens einen Knoten in VC!

Fügen wir jeweils *beide* Knoten zu VC hinzu, erhalten wir ein Vertex Cover!
(bei inklusionsmaximalen Matchings)

Inklusionsmaximales Matching

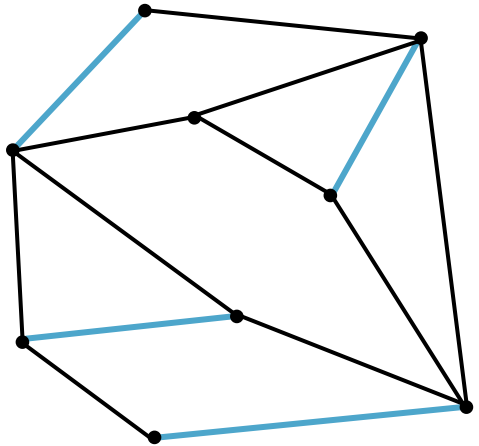


Matching

Für jede Kante im Matching benötigen wir mindestens einen Knoten in VC!

Fügen wir jeweils *beide* Knoten zu VC hinzu, erhalten wir ein Vertex Cover!
(bei inklusionsmaximalen Matchings)

Inklusionsmaximales Matching



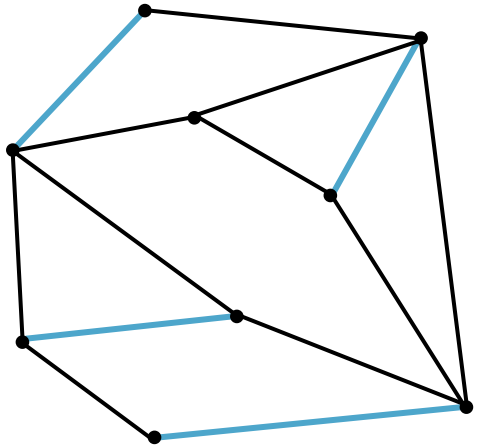
Matching

Für jede Kante im Matching benötigen wir mindestens einen Knoten in VC!

Fügen wir jeweils *beide* Knoten zu VC hinzu, erhalten wir ein Vertex Cover!
(bei inklusionsmaximalen Matchings)

Untere Schranke für Min. Vertex Cover!

Inklusionsmaximales Matching



Matching

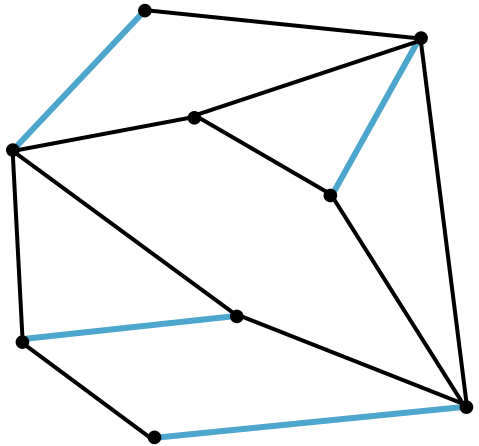
Für jede Kante im Matching benötigen wir mindestens einen Knoten in VC!

Fügen wir jeweils *beide* Knoten zu VC hinzu, erhalten wir ein Vertex Cover!
(bei inklusionsmaximalen Matchings)

Untere Schranke für Min. Vertex Cover!

Obere Schranke für Min. Vertex Cover!

Inklusionsmaximales Matching



Matching

Für jede Kante im Matching benötigen wir mindestens einen Knoten in VC!

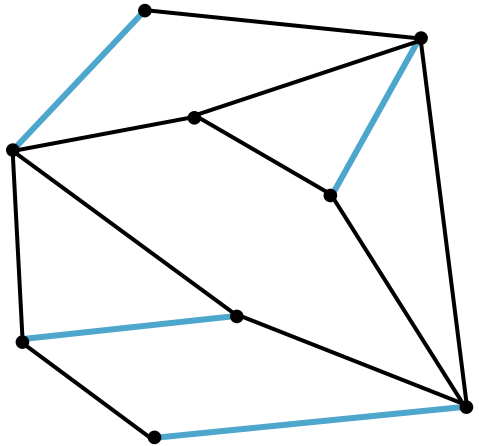
Untere Schranke für Min. Vertex Cover!

Fügen wir jeweils *beide* Knoten zu VC hinzu, erhalten wir ein Vertex Cover!
(bei inklusionsmaximalen Matchings)

Obere Schranke für Min. Vertex Cover!

Damit haben wir eine 2-Approximation für Vertex Cover gefunden!

Inklusionsmaximales Matching



Matching

Für jede Kante im Matching benötigen wir mindestens einen Knoten in VC!

Untere Schranke für Min. Vertex Cover!

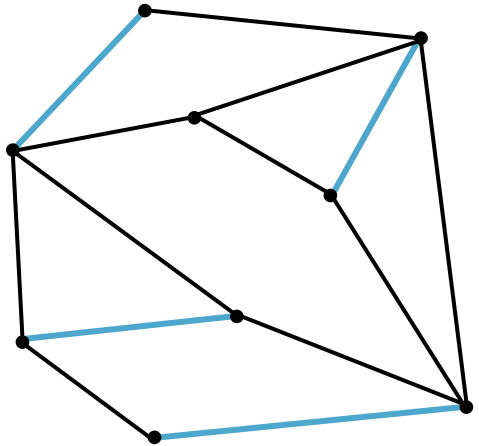
Fügen wir jeweils *beide* Knoten zu VC hinzu, erhalten wir ein Vertex Cover!
(bei inklusionsmaximalen Matchings)

Obere Schranke für Min. Vertex Cover!

Damit haben wir eine 2-Approximation für Vertex Cover gefunden!

… wir schreiben das mal sauber auf:

Inklusionsmaximales Matching



Vertex Cover und Matchings

Vertex Cover und Matchings

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

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

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.

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.

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

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.

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

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

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.

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

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

Annahme: VC' ist kein Vertex Cover

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.

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

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

Annahme: VC' ist kein Vertex Cover

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

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.

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

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.

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.

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

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.

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

Vertex Cover und Matchings

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

Vertex Cover und Matchings

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

Aus welchen Schritten besteht diese Approximation?

Vertex Cover und Matchings

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

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

Aus welchen Schritten besteht diese Approximation?

Vertex Cover und Matchings

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

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

Aus welchen Schritten besteht diese Approximation?

Vertex Cover und Matchings

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

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

Aus welchen Schritten besteht diese Approximation?

Vertex Cover und Matchings

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

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

Aus welchen Schritten besteht diese Approximation?

Vertex Cover und Matchings

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

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

Aus welchen Schritten besteht diese Approximation?

Wir haben ja gerade gesehen, wie man dieses Vertex Cover findet.

Vertex Cover und Matchings

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

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

Aus welchen Schritten besteht diese Approximation?

- Finde für den gegebenen Graph ein inklusionsmaximales Matching M

Wir haben ja gerade gesehen, wie man dieses Vertex Cover findet.

Vertex Cover und Matchings

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

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

Aus welchen Schritten besteht diese Approximation?

Wir haben ja gerade gesehen, wie man dieses Vertex Cover findet.

- Finde für den gegebenen Graph ein inklusionsmaximales Matching M
- Wähle beide Knoten in allen Kanten aus M als Rückgabe aus

Vertex Cover und Matchings

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

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

Aus welchen Schritten besteht diese Approximation?

Wir haben ja gerade gesehen, wie man dieses Vertex Cover findet.

- Finde für den gegebenen Graph ein inklusionsmaximales Matching M
- Wähle beide Knoten in allen Kanten aus M als Rückgabe aus

Beide Schritte zusammen:

Vertex Cover und Matchings

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

Beide Schritte zusammen:

Vertex Cover und Matchings

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

Beide Schritte zusammen:

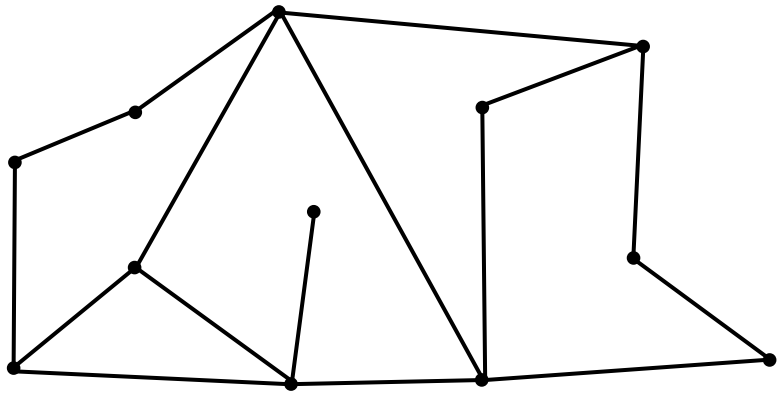
$VC \coloneqq \emptyset$
for each $\{u, v\} \in E$ **do**
  if $u \notin VC$ und $v \notin VC$ **then**
    $VC \coloneqq VC \cup \{u, v\}$
return VC

Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

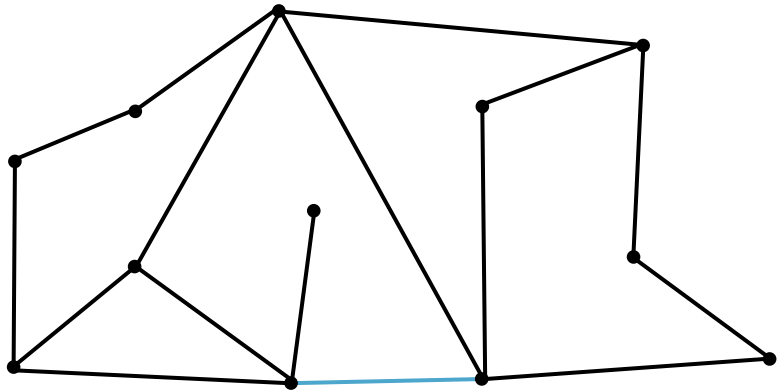


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

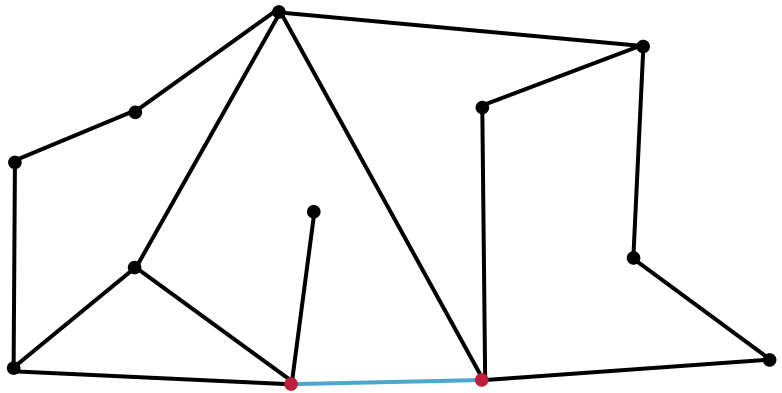


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

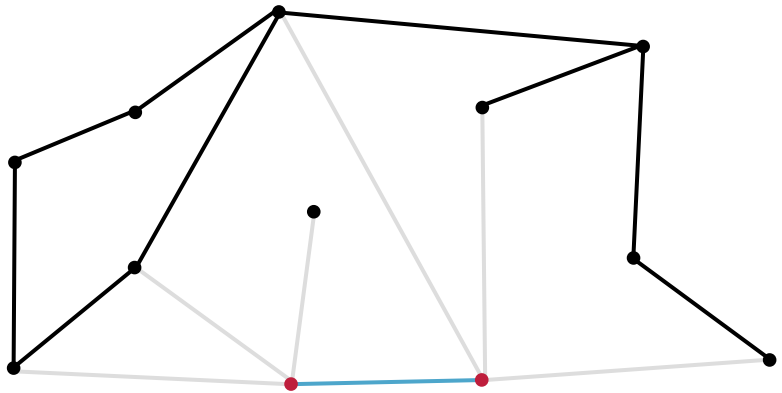


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

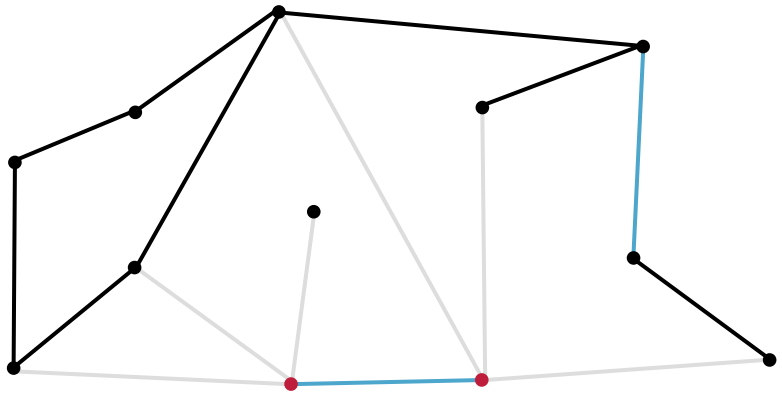


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

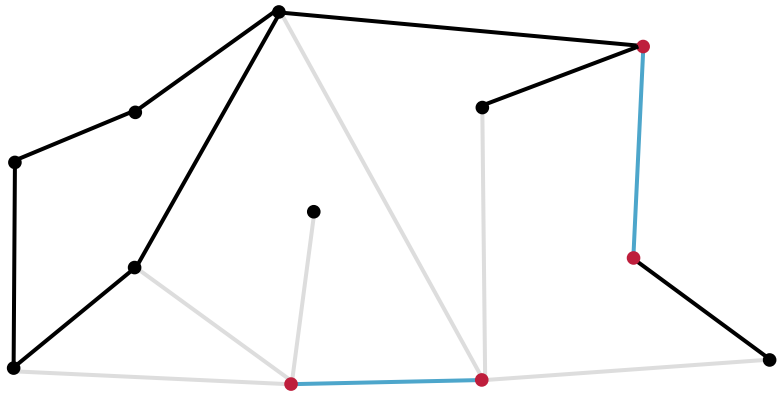


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

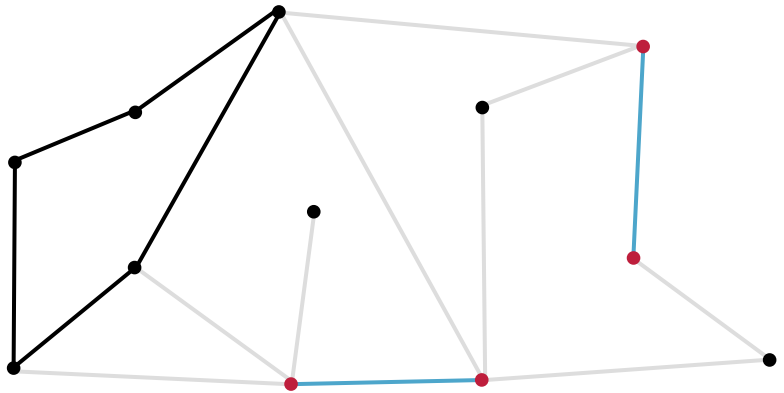


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

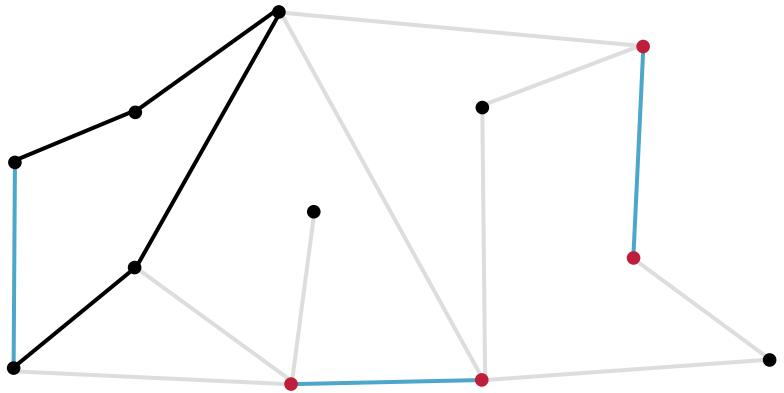


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

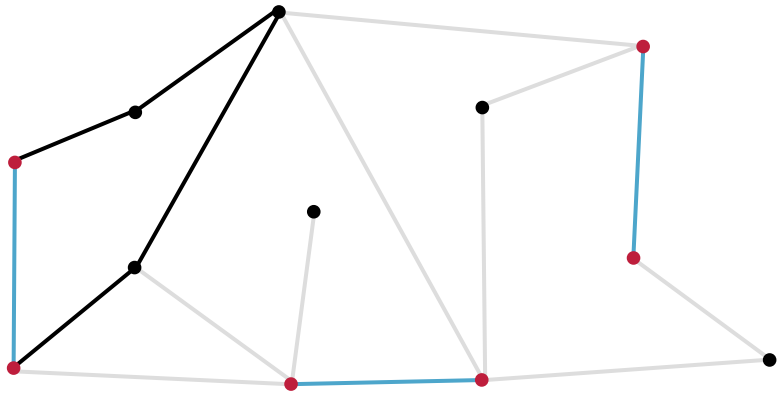


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

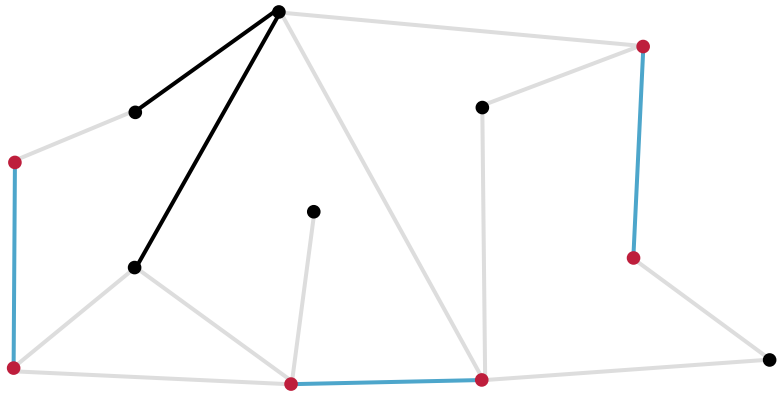


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

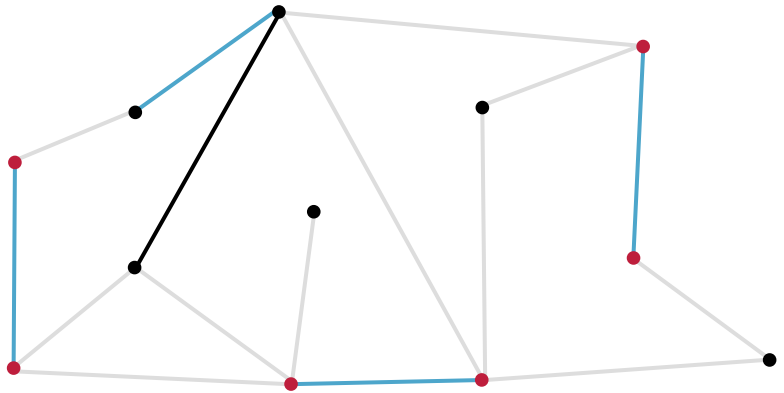


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

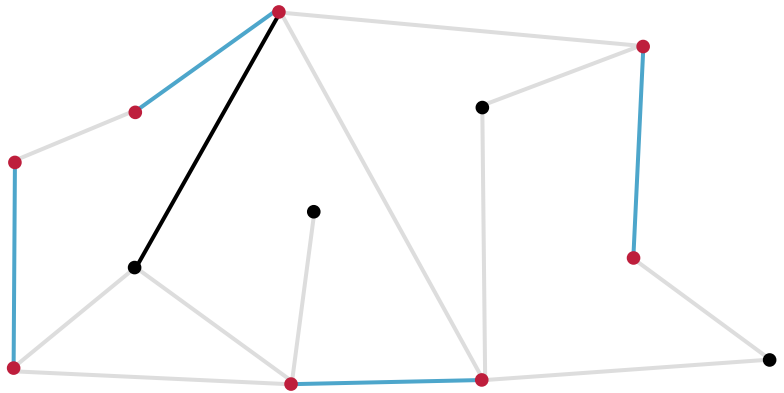


Vertex Cover und Matchings

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

Beide Schritte zusammen:

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

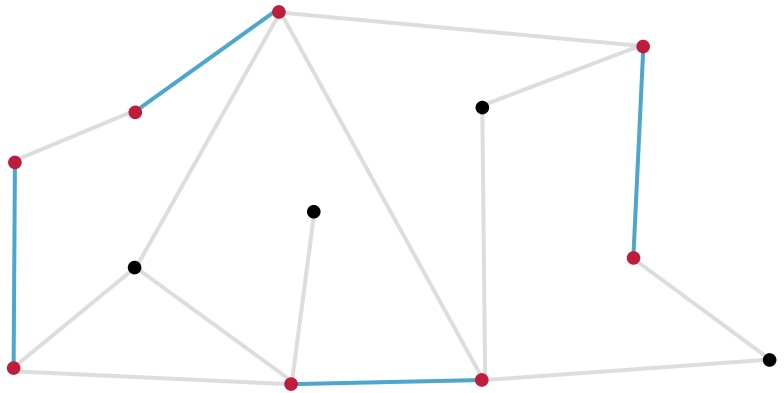


Vertex Cover und Matchings

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

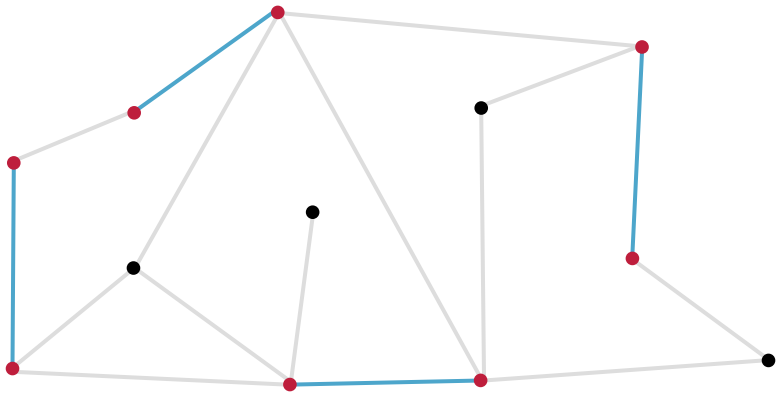
Beide Schritte zusammen:

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



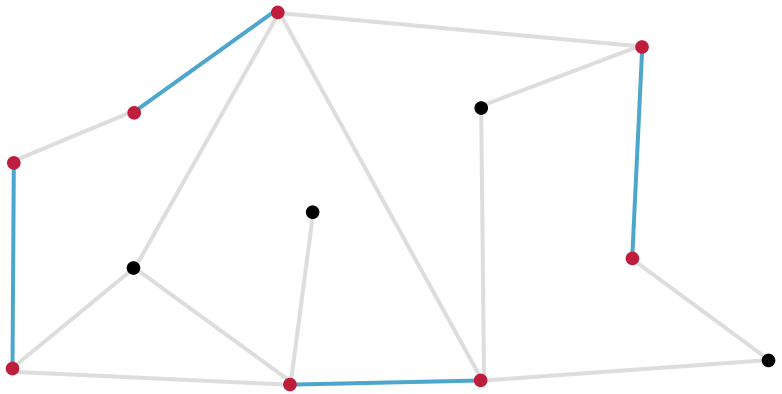
Fragen?

Vertex Cover und Matchings



Vertex Cover und Matchings

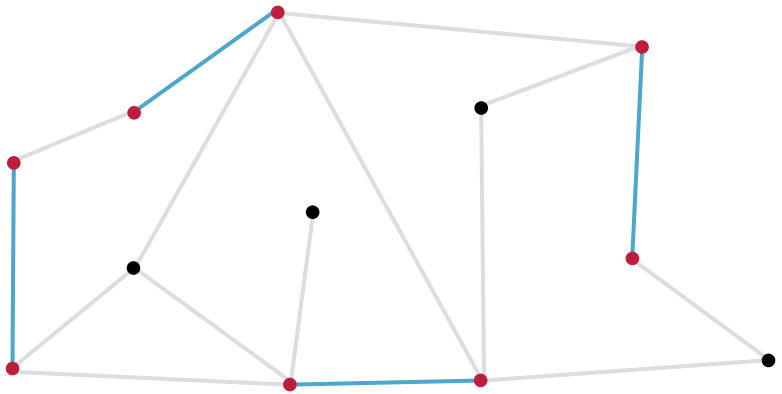
In diesem Graphen gibt es ein inklusionsmaximales Matching der Größe 4.



Vertex Cover und Matchings

In diesem Graphen gibt es ein inklusionsmaximales Matching der Größe 4.

Also gilt für ein minimales Vertex-Cover VC:

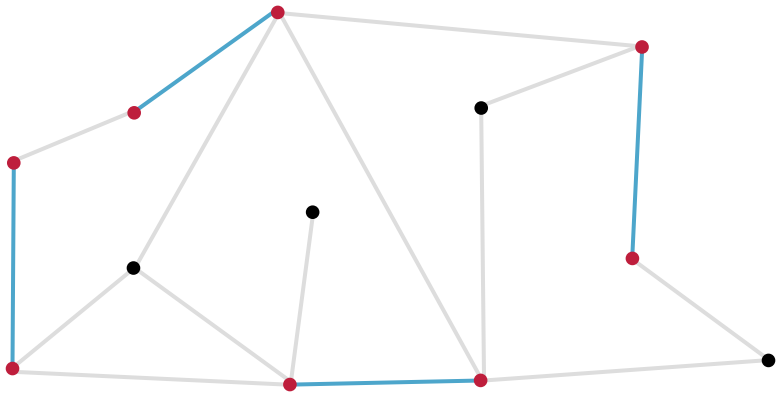


Vertex Cover und Matchings

In diesem Graphen gibt es ein inklusionsmaximales Matching der Größe 4.

Also gilt für ein minimales Vertex-Cover VC:

$$4 \leq |VC| \leq 8$$



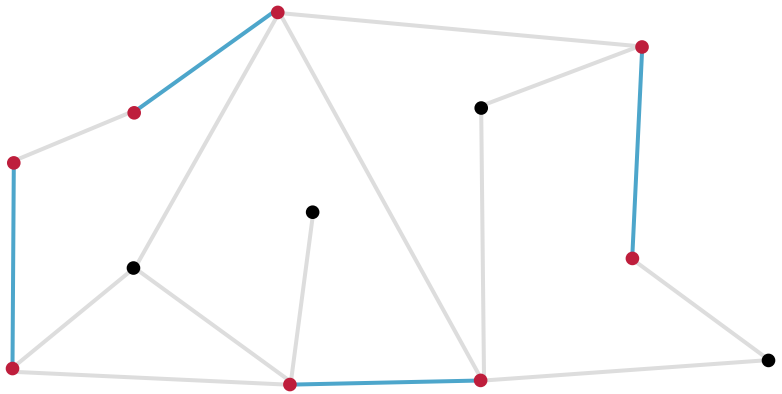
Vertex Cover und Matchings

In diesem Graphen gibt es ein inklusionsmaximales Matching der Größe 4.

Also gilt für ein minimales Vertex-Cover VC:

$$4 \leq |VC| \leq 8$$

Basierend hierauf:
Wie können wir andere (und vielleicht bessere) untere Schranken finden?



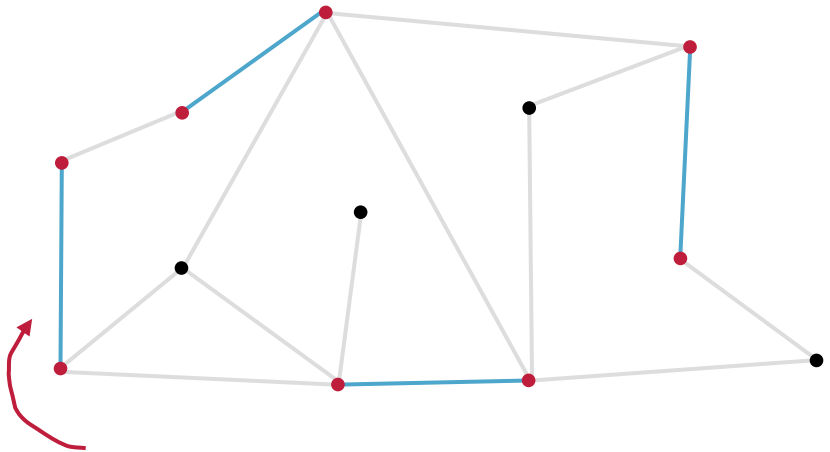
Vertex Cover und Matchings

In diesem Graphen gibt es ein inklusionsmaximales Matching der Größe 4.

Also gilt für ein minimales Vertex-Cover VC:

$$4 \leq |VC| \leq 8$$

Basierend hierauf:
Wie können wir andere (und vielleicht bessere) untere Schranken finden?



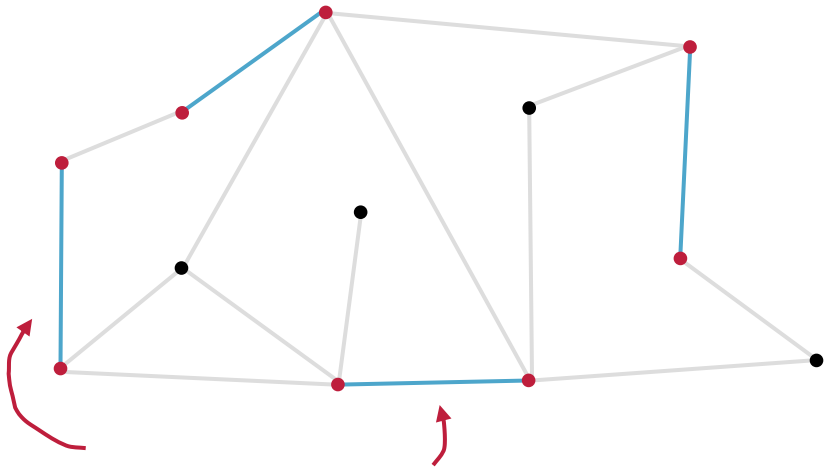
Vertex Cover und Matchings

In diesem Graphen gibt es ein inklusionsmaximales Matching der Größe 4.

Also gilt für ein minimales Vertex-Cover VC:

$$4 \leq |VC| \leq 8$$

Basierend hierauf:
Wie können wir andere (und vielleicht bessere) untere Schranken finden?



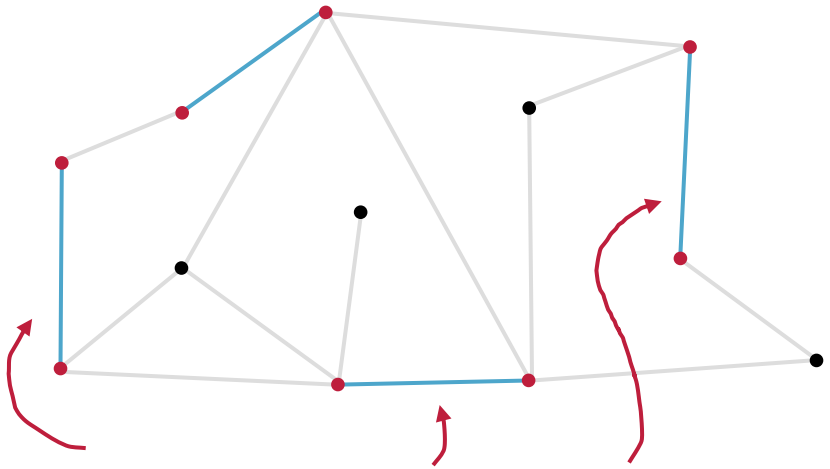
Vertex Cover und Matchings

In diesem Graphen gibt es ein inklusionsmaximales Matching der Größe 4.

Also gilt für ein minimales Vertex-Cover VC:

$$4 \leq |VC| \leq 8$$

Basierend hierauf:
Wie können wir andere (und vielleicht bessere) untere Schranken finden?



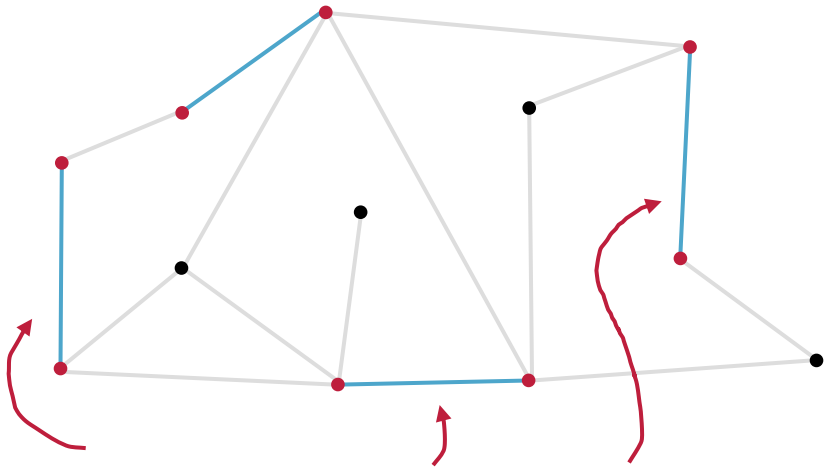
Vertex Cover und Matchings

In diesem Graphen gibt es ein inklusionsmaximales Matching der Größe 4.

Also gilt für ein minimales Vertex-Cover VC:

$$4 \leq |VC| \leq 8$$

Basierend hierauf:
Wie können wir andere (und vielleicht bessere) untere Schranken finden?



Vielleicht können wir den Graphen statt in einzelne unabhängige Kanten in größere Teilgraphen unterteilen und die jeweils lösen …

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

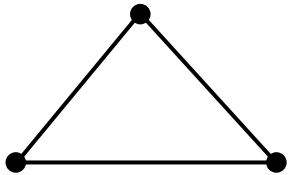
Habt ihr Ideen, wo das geht?

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

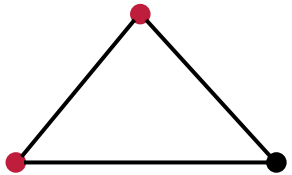


Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

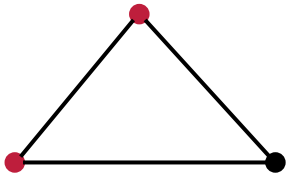


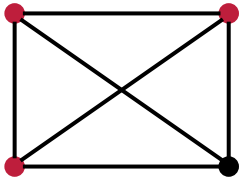
Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen



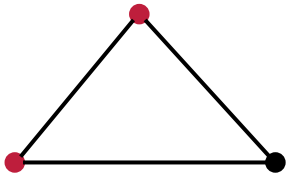


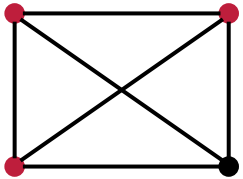
Optimale Vertex Cover

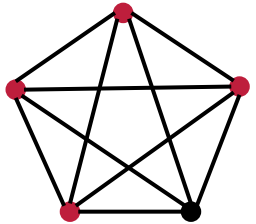
Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen





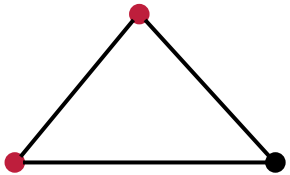


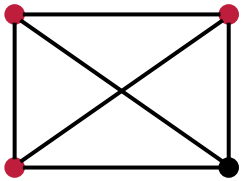
Optimale Vertex Cover

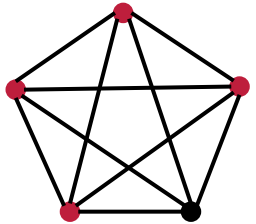
Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen







$$|VC| = n - 1$$

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$$|VC| = n - 1$$

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

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.

Habt ihr Ideen, wo das geht?

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.

Habt ihr Ideen, wo das geht?

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.

Habt ihr Ideen, wo das geht?

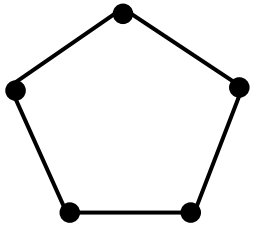
Vollständige Graphen

$$|VC| = n - 1$$

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.

Habt ihr Ideen, wo das geht?

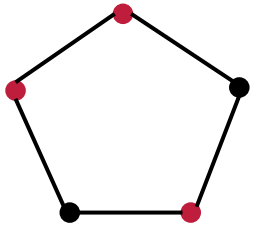
Vollständige Graphen

$$|VC| = n - 1$$

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.

Habt ihr Ideen, wo das geht?

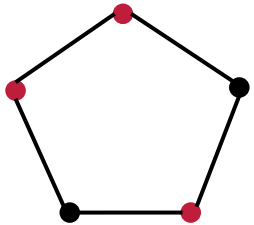
Vollständige Graphen

$$|VC| = n - 1$$

Pfad Graphen

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

Kreis Graphen



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

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

Vollständige Graphen

$|VC| = n - 1$

Pfad Graphen

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

Kreis Graphen

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

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

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

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

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

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

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

Optimale Vertex Cover

Für spezielle Graphen kann man kleinste Vertex Cover effizient bestimmen.

Habt ihr Ideen, wo das geht?

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

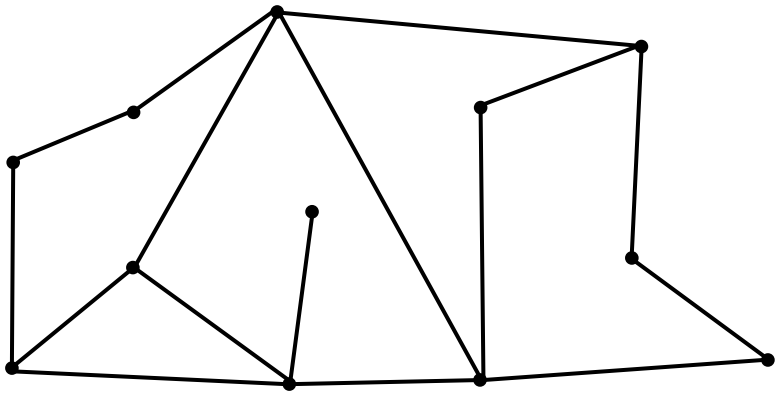
$O(n)$ Algorithmus

Intervall Graphen

$O(n \log n)$ Algorithmus

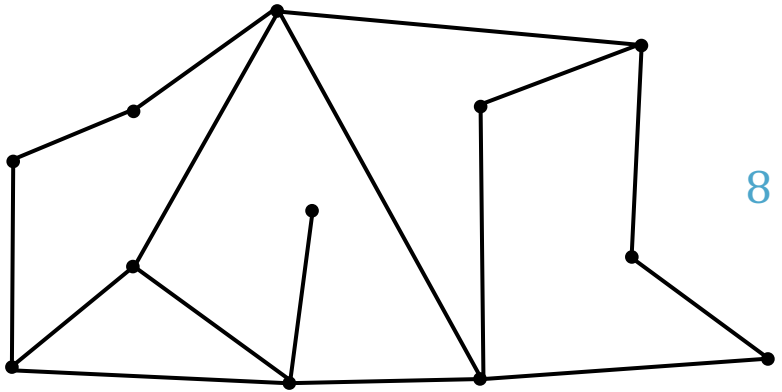
Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen



Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

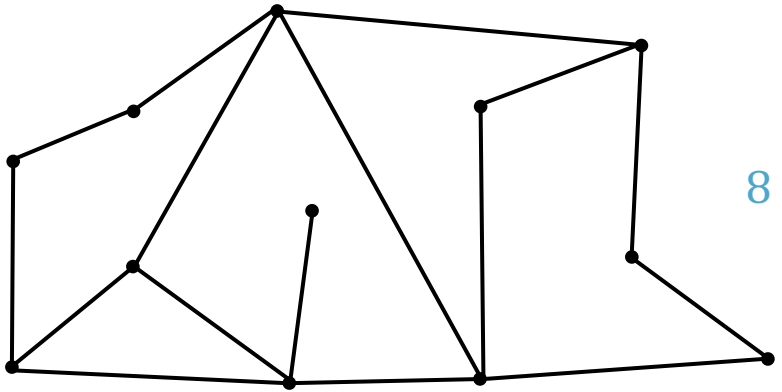


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

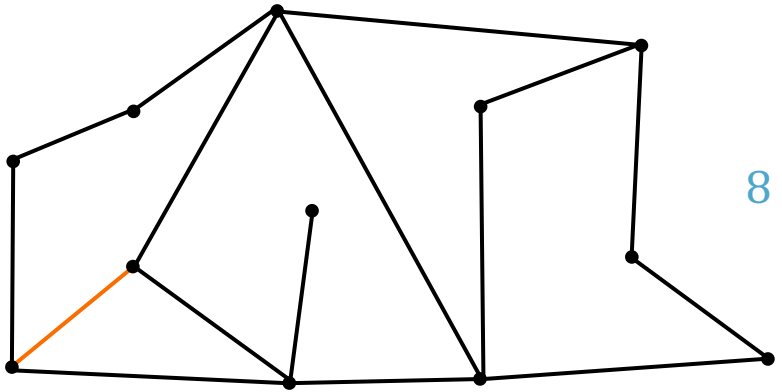


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

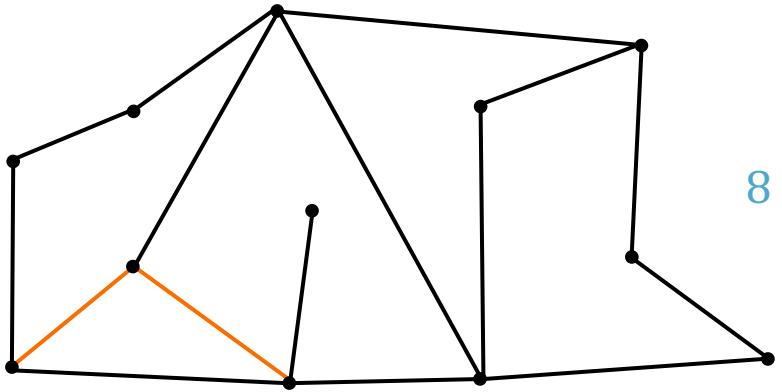


$$8 \geq |VC| \geq 4$$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

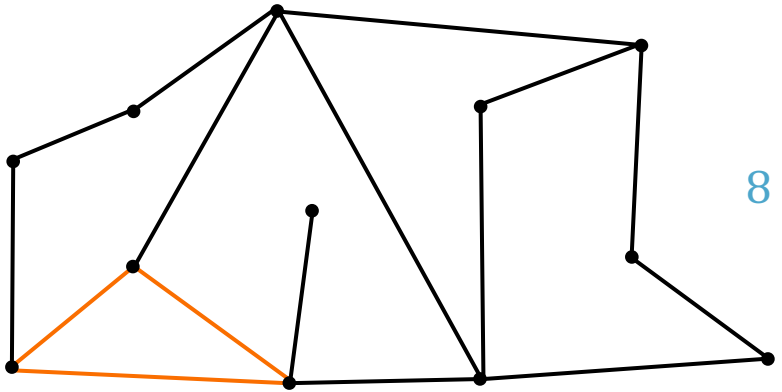


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

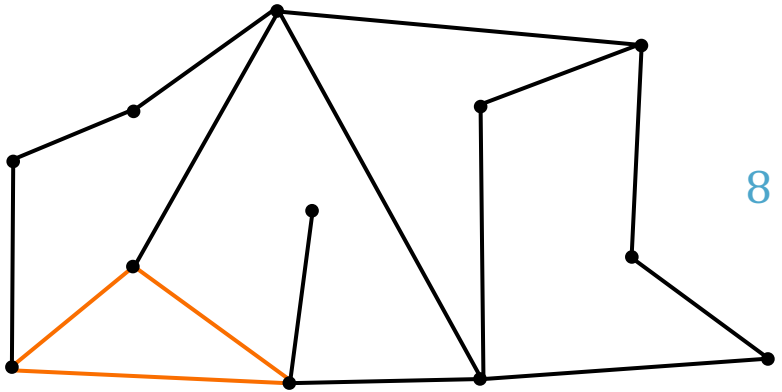
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

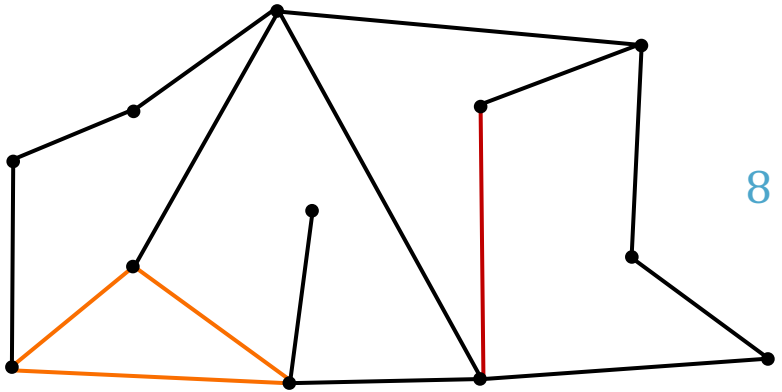
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

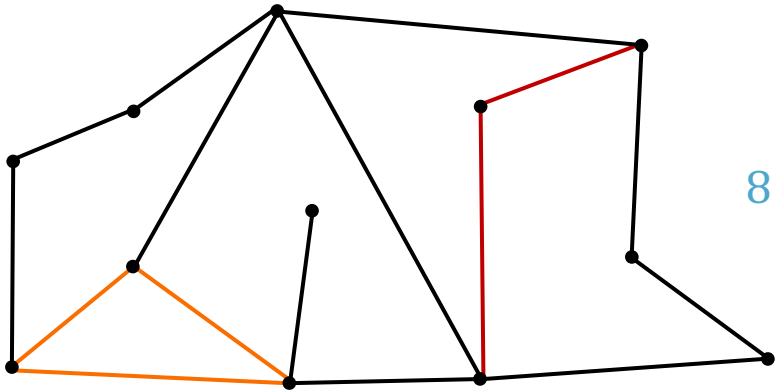
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

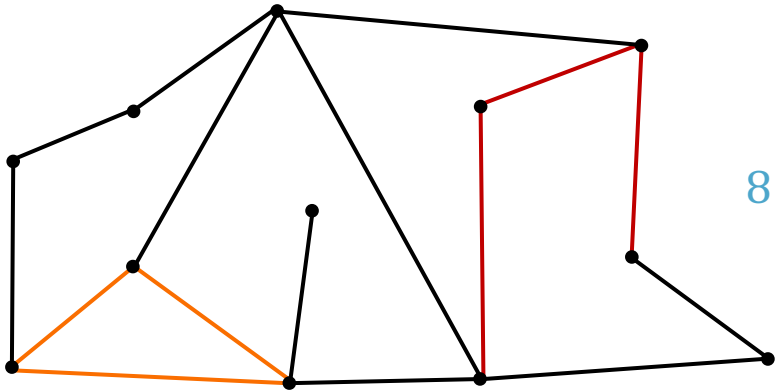
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

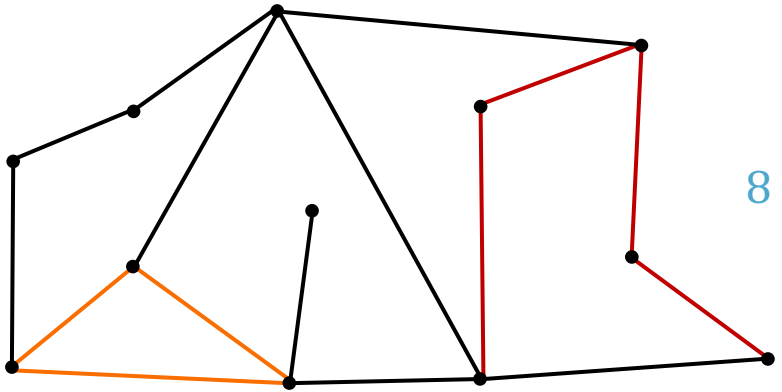
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

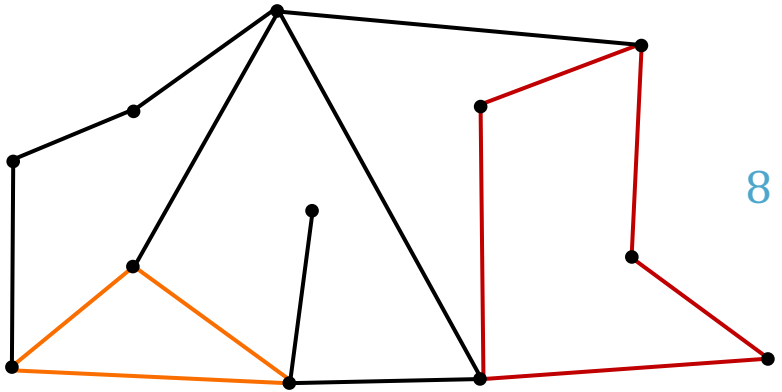
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

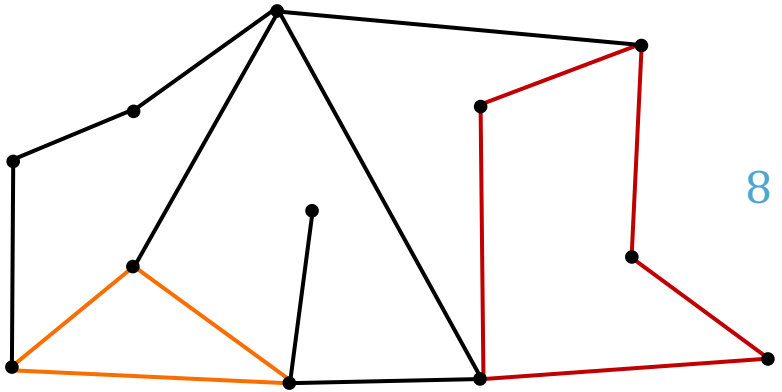
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

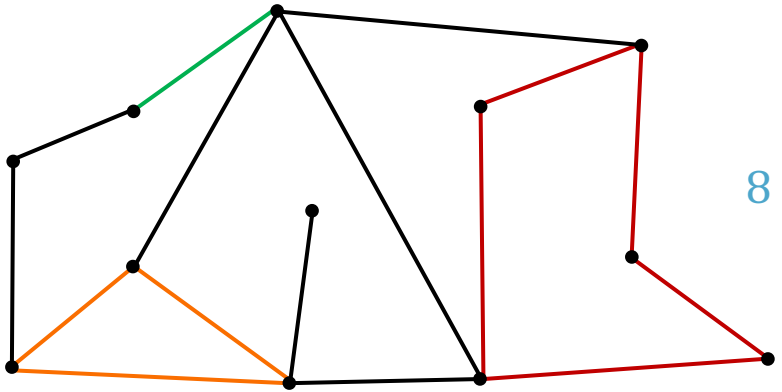
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

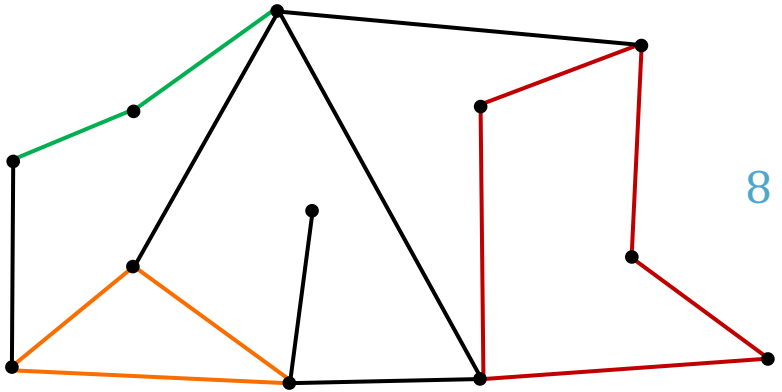
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

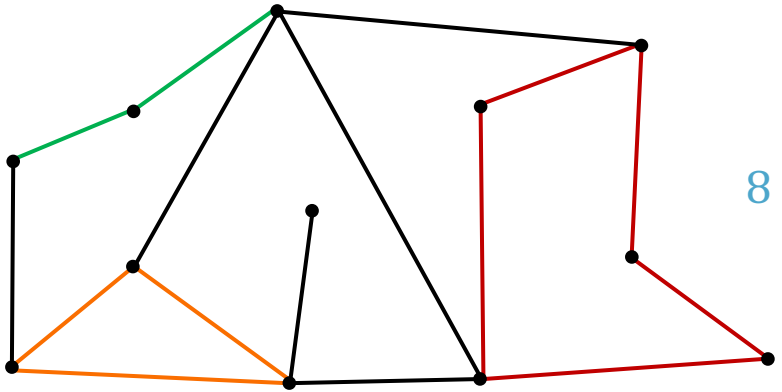
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

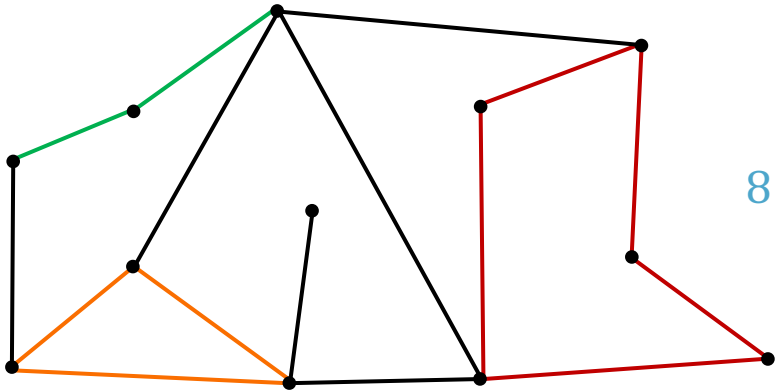
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

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

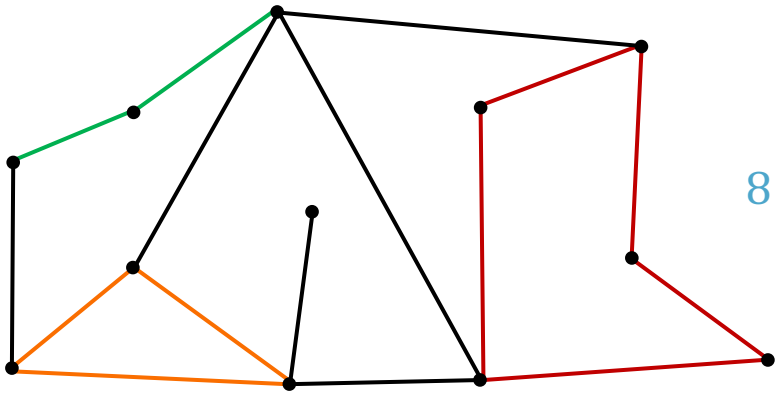
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



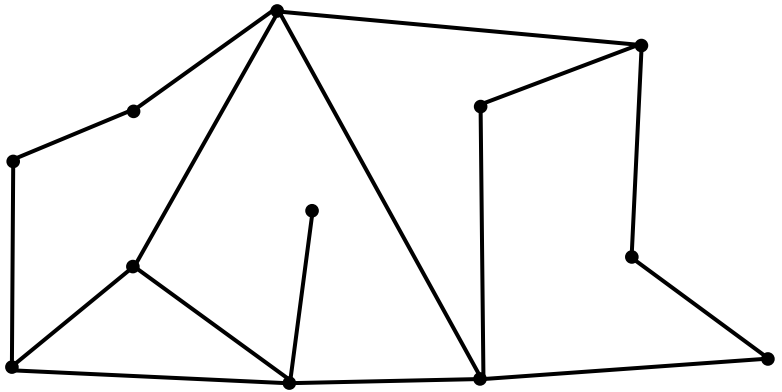
$8 \geq |VC| \geq 4$

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

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

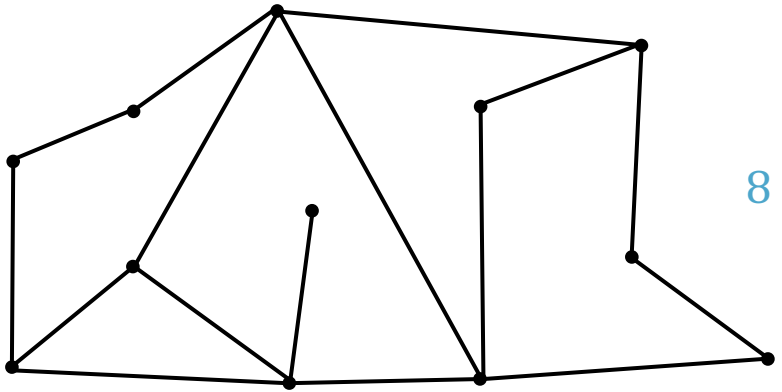
Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen



Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

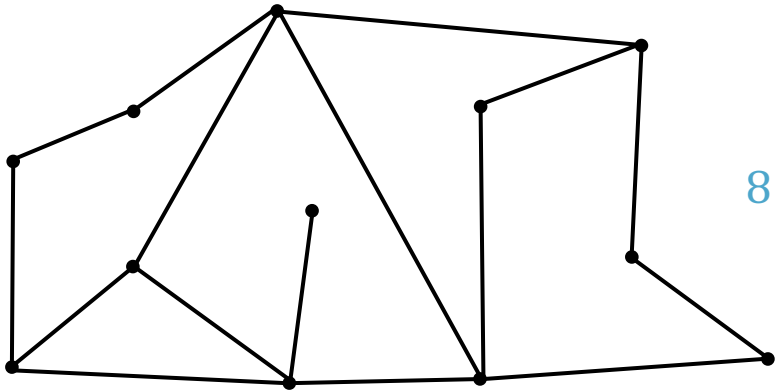


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

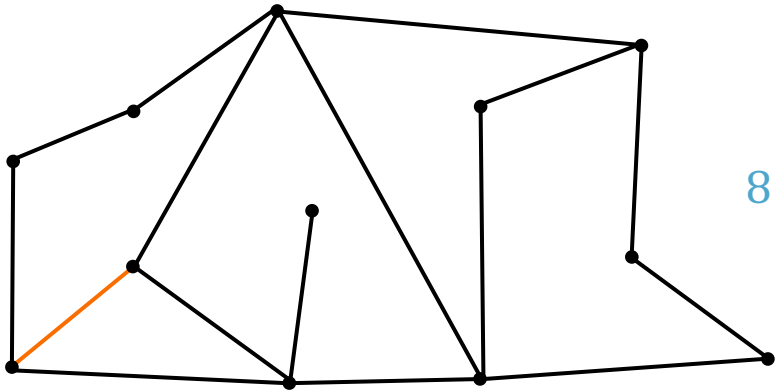


$$8 \geq |VC| \geq 4$$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

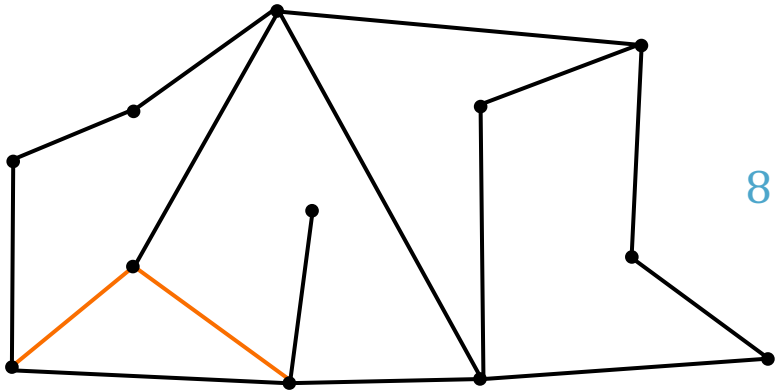


$$8 \geq |VC| \geq 4$$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

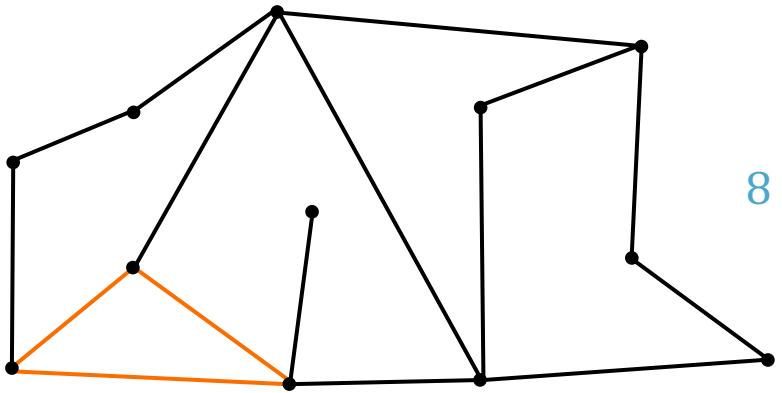


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

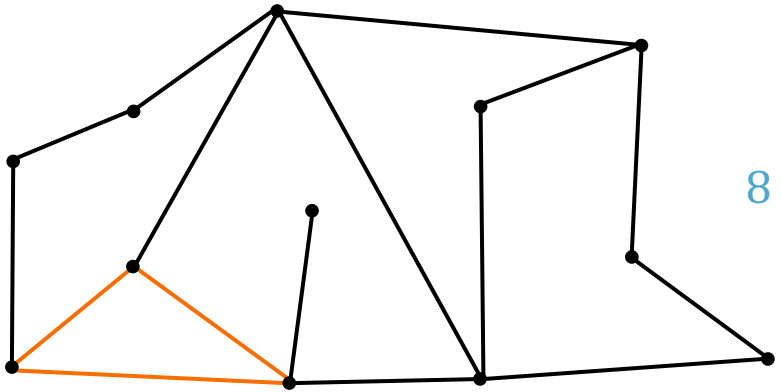
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

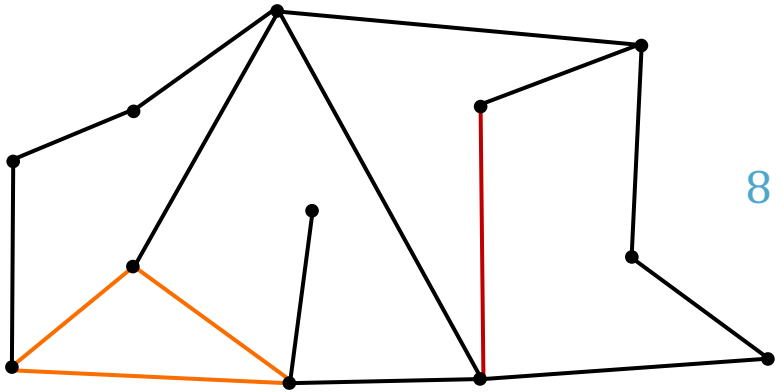
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

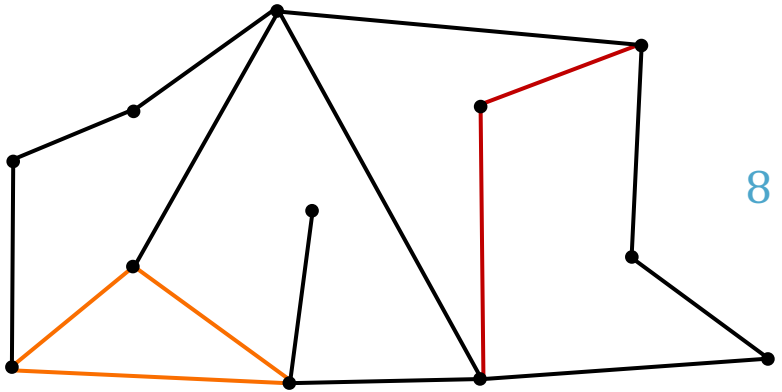
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

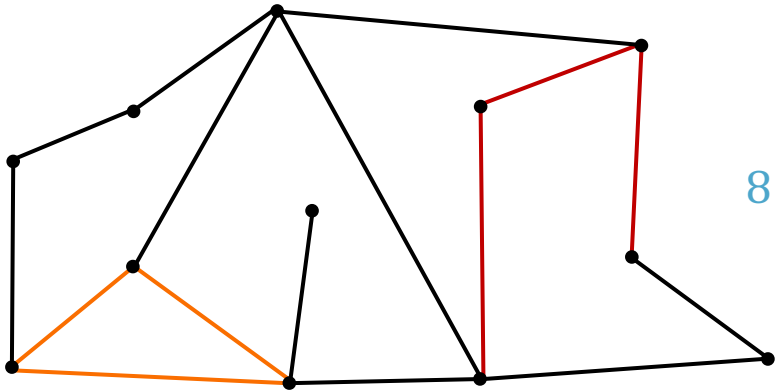
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

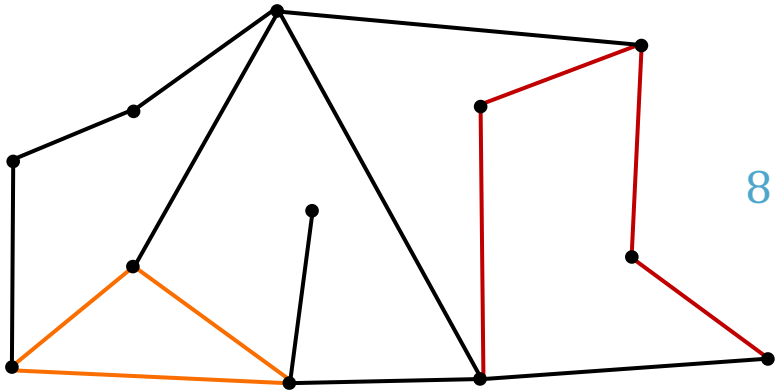
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

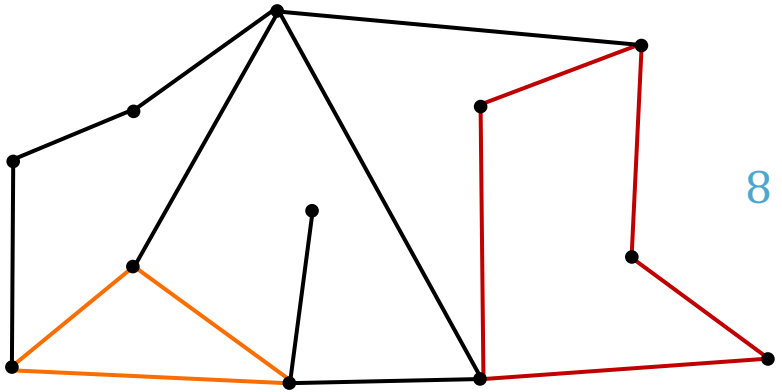
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

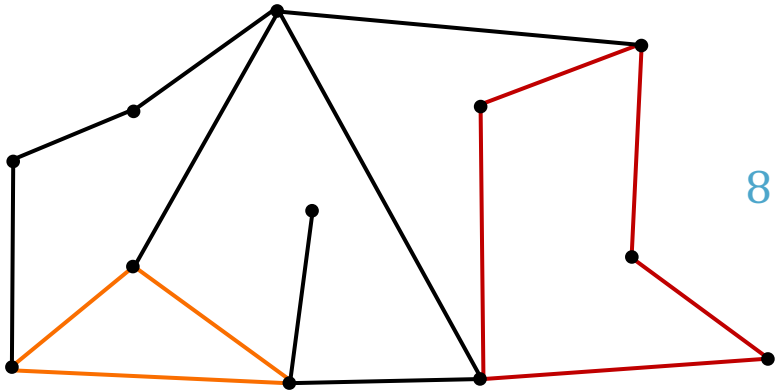
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

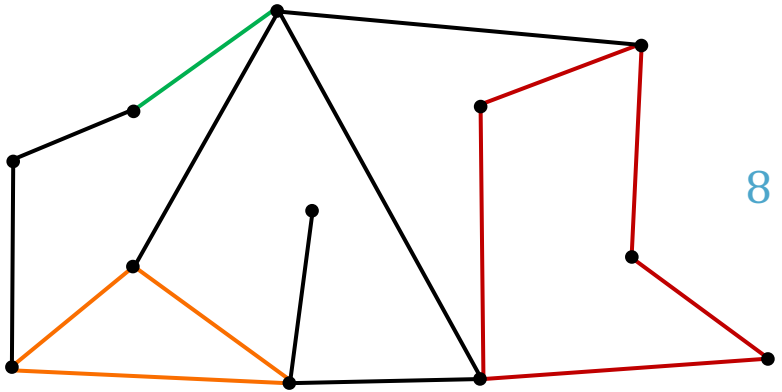
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

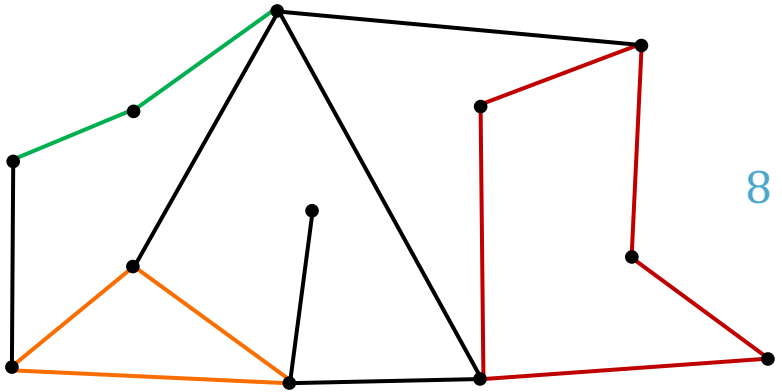
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

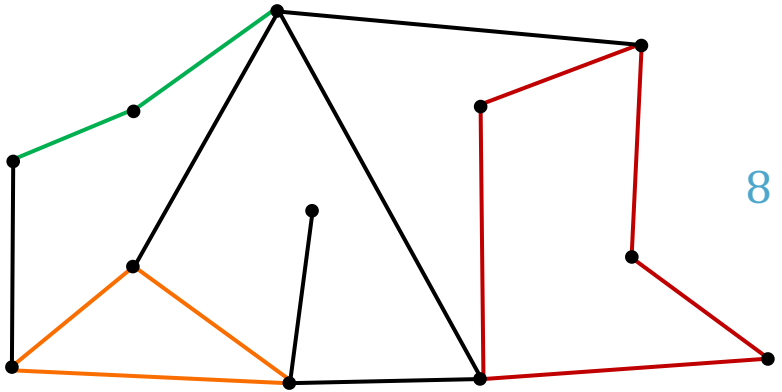
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

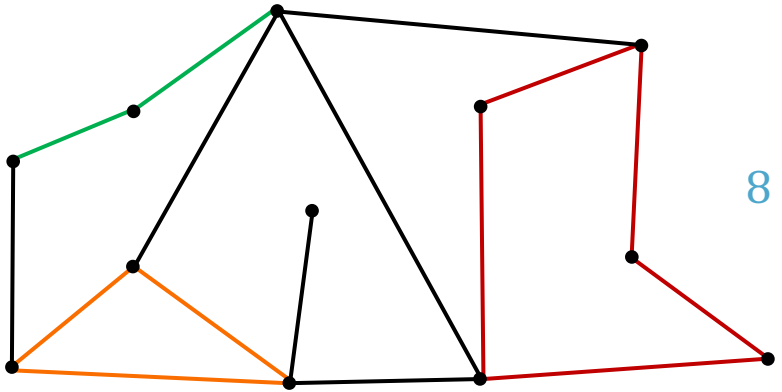
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

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

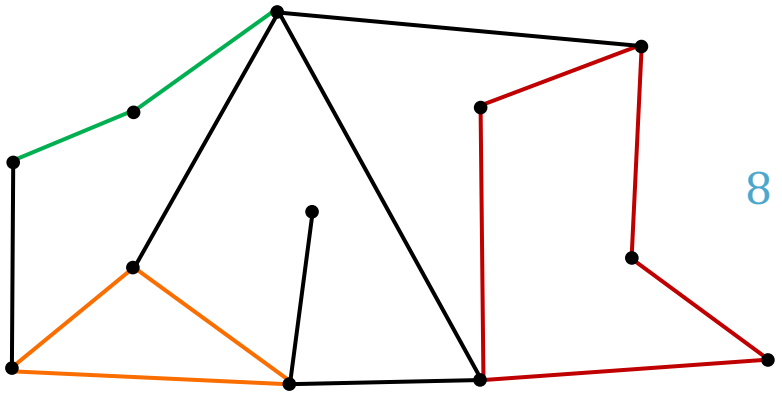
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



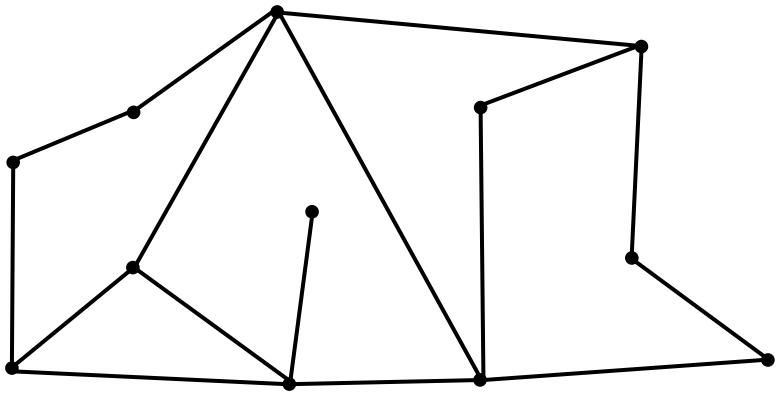
$8 \geq |VC| \geq 4$

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

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

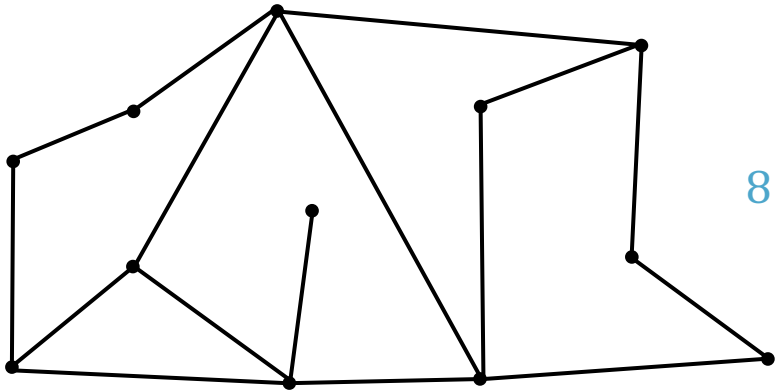
Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen



Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

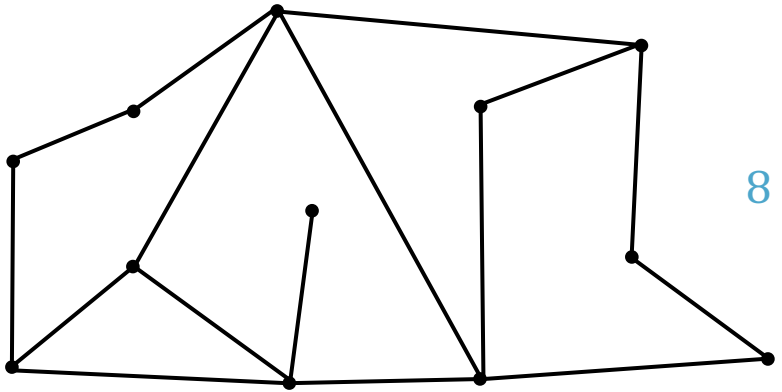


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

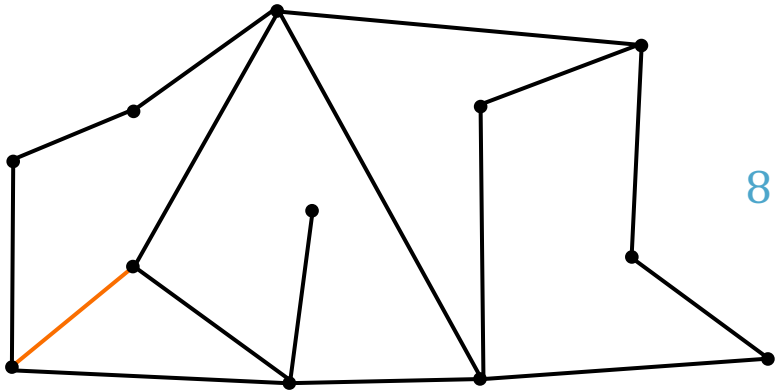


$$8 \geq |VC| \geq 4$$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

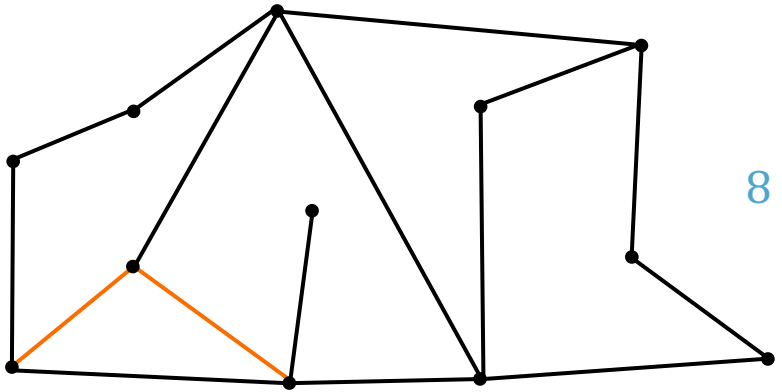


$$8 \geq |VC| \geq 4$$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

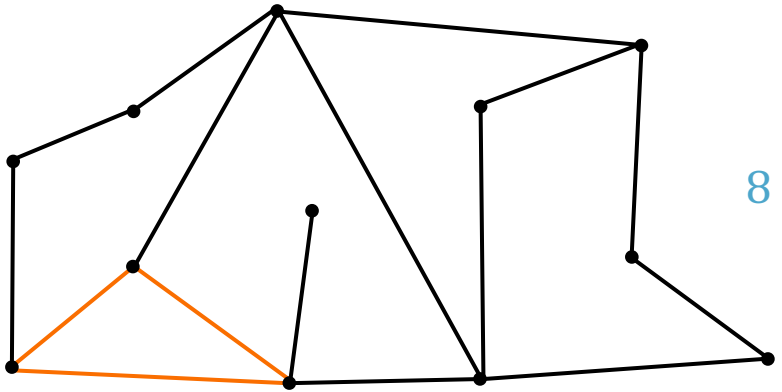


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

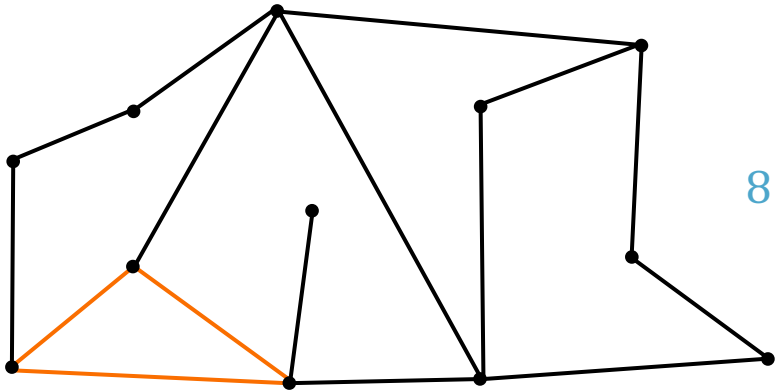
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

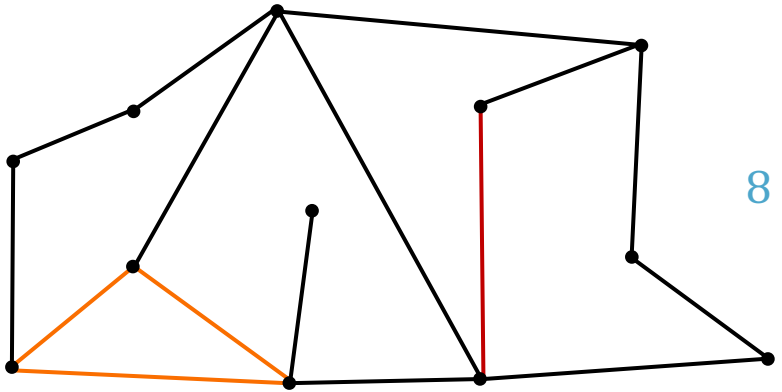
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

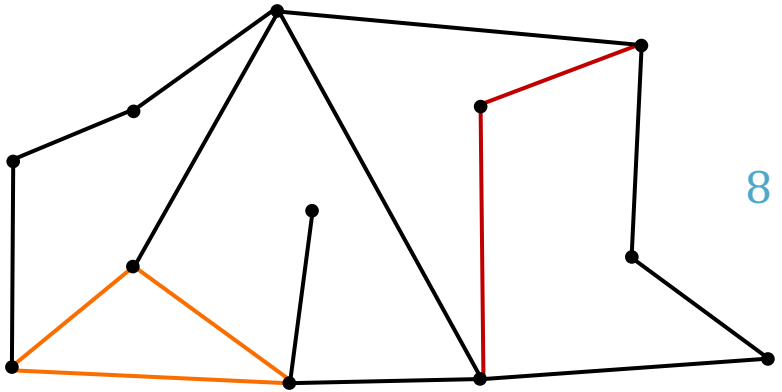
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

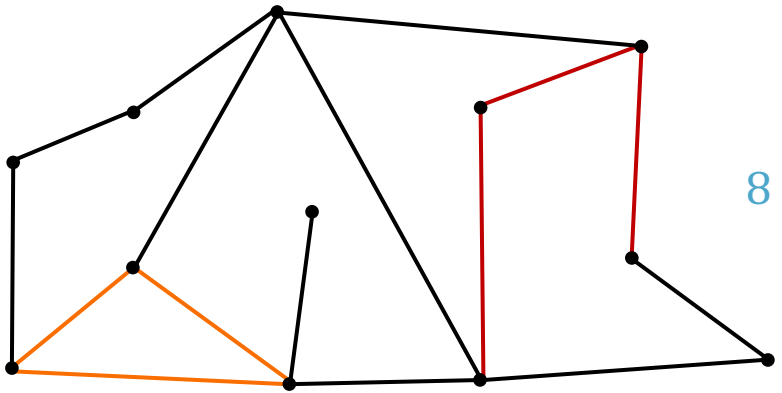
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

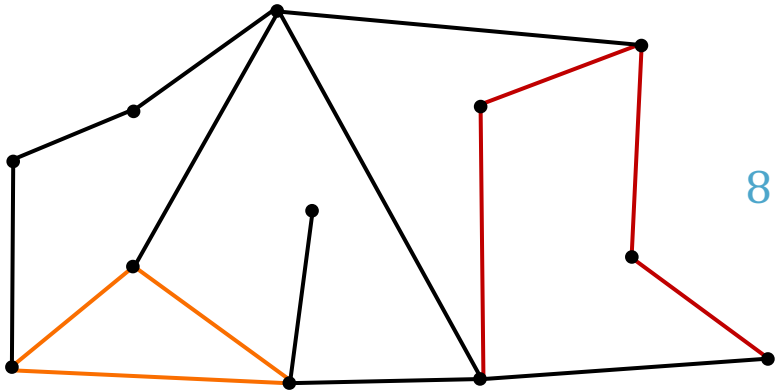
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

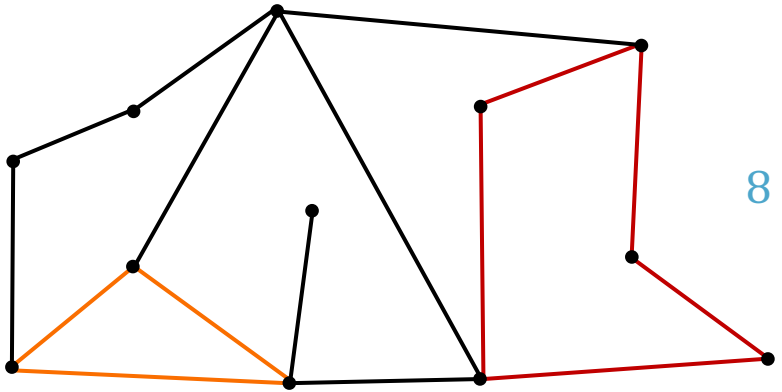
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

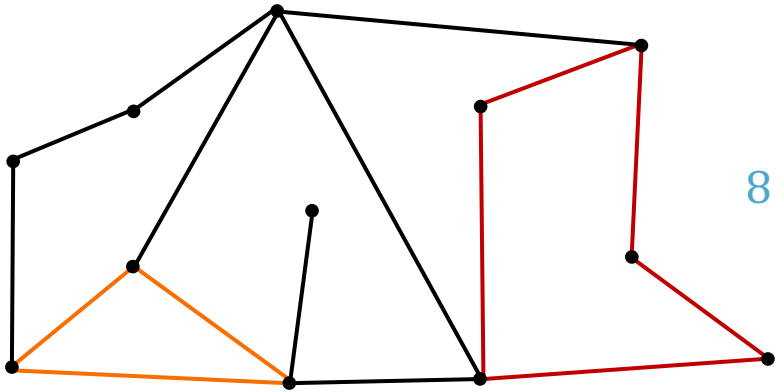
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

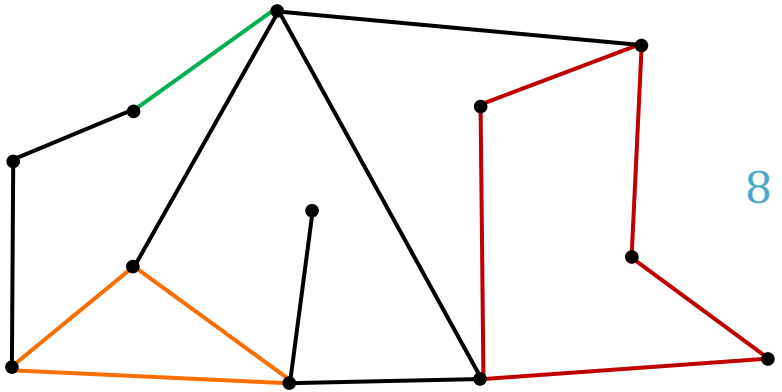
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

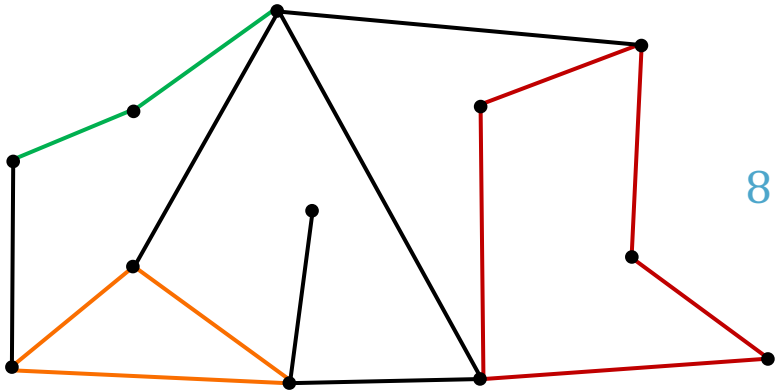
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

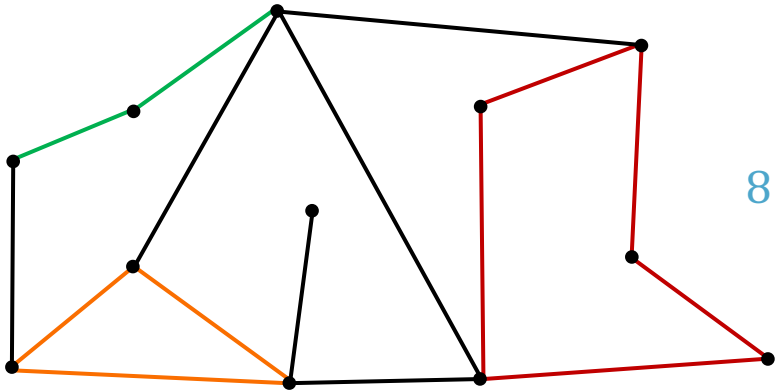
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

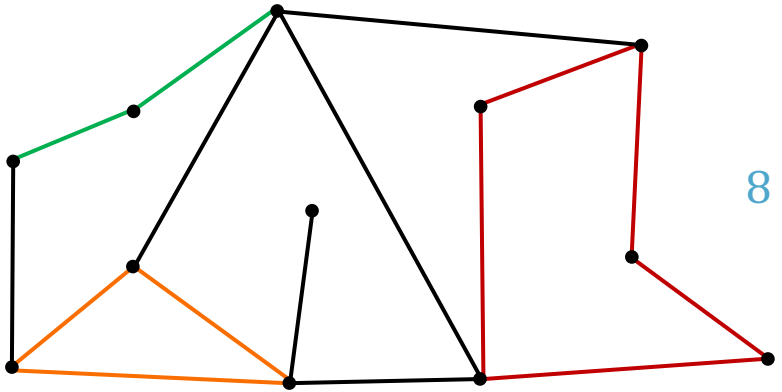
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

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

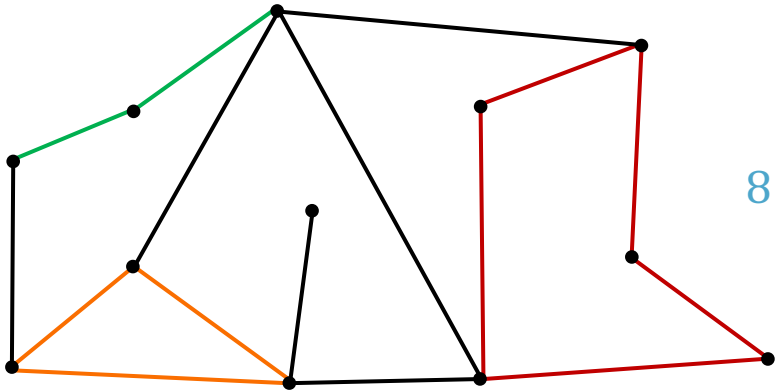
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



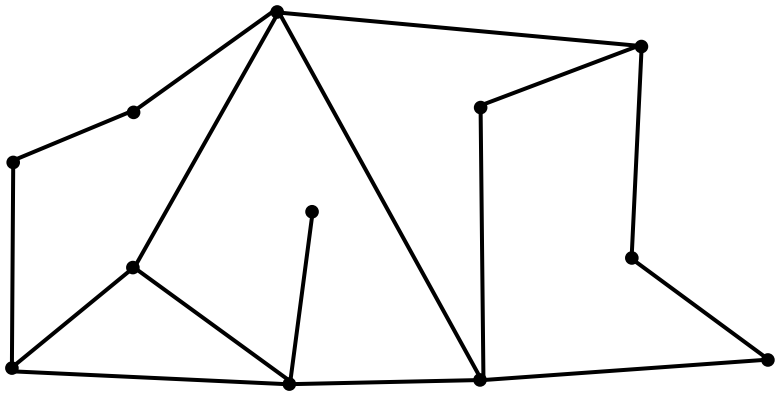
$8 \geq |VC| \geq 4$

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

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

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

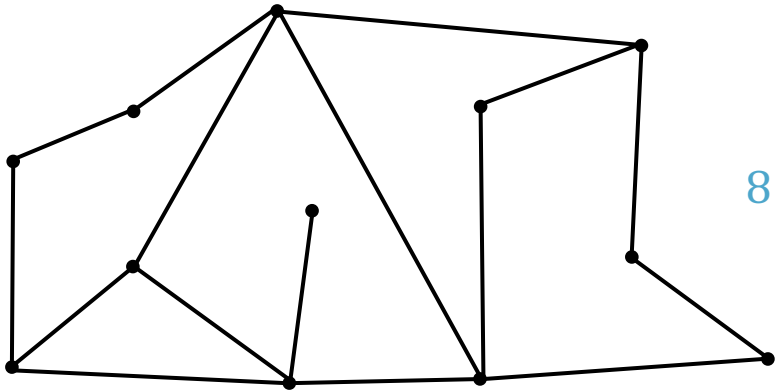


Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

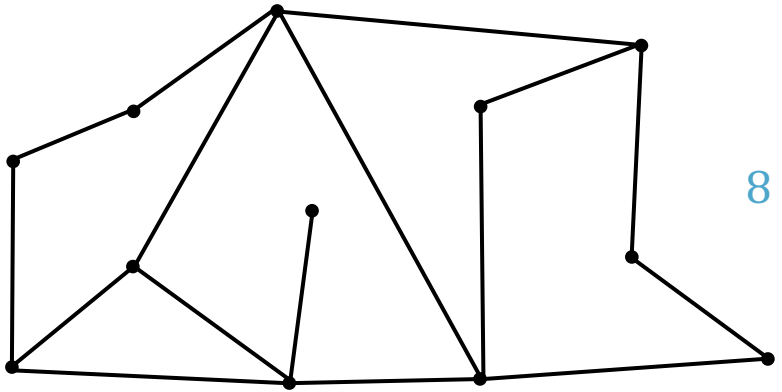


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

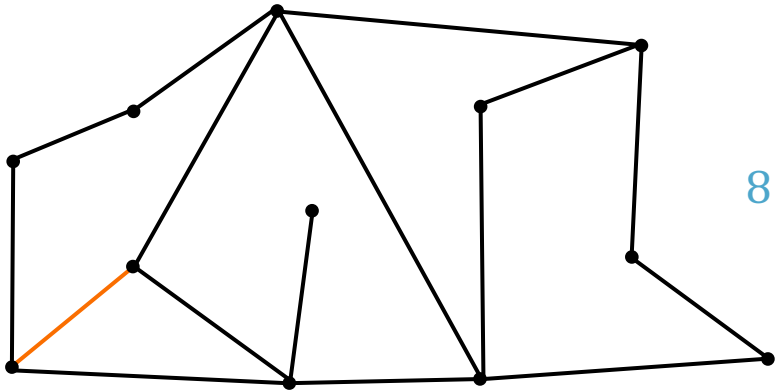


$$8 \geq |VC| \geq 4$$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

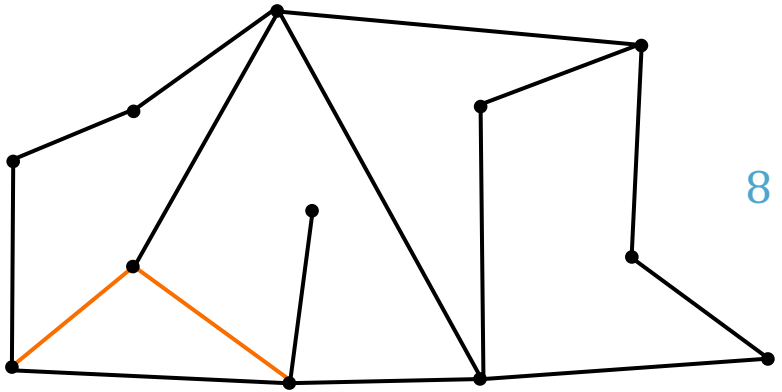


$$8 \geq |VC| \geq 4$$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

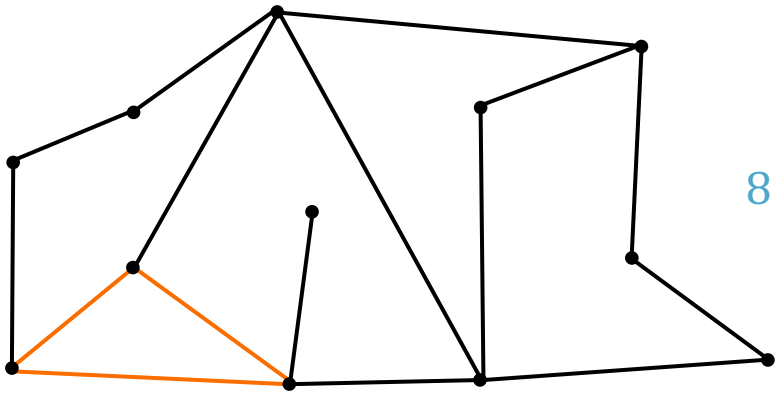


$$8 \geq |VC| \geq 4$$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$$8 \geq |VC| \geq 4$$

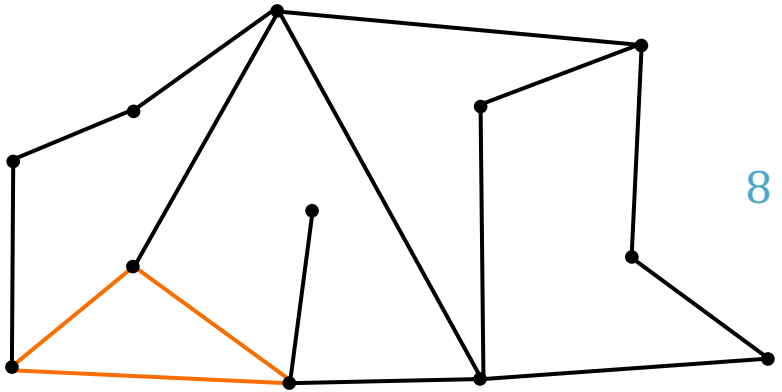
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

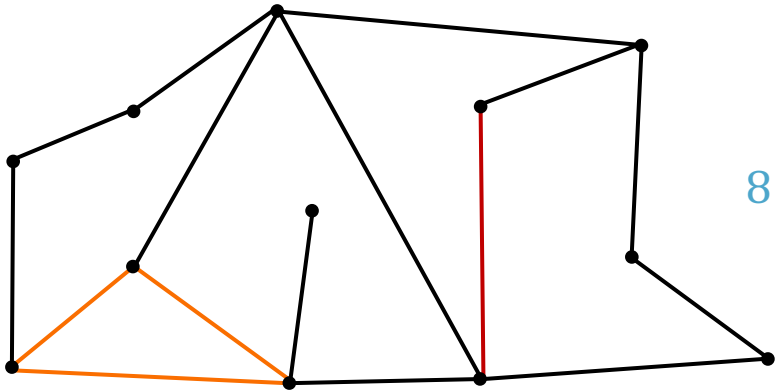
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

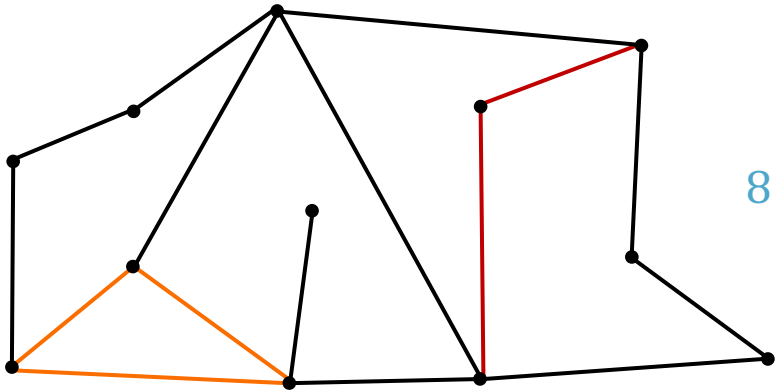
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

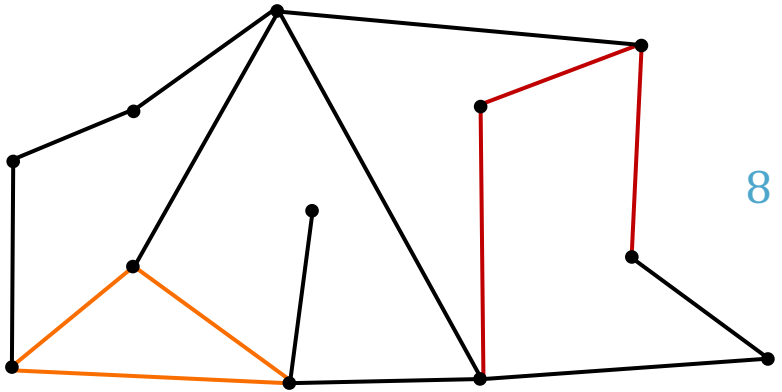
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

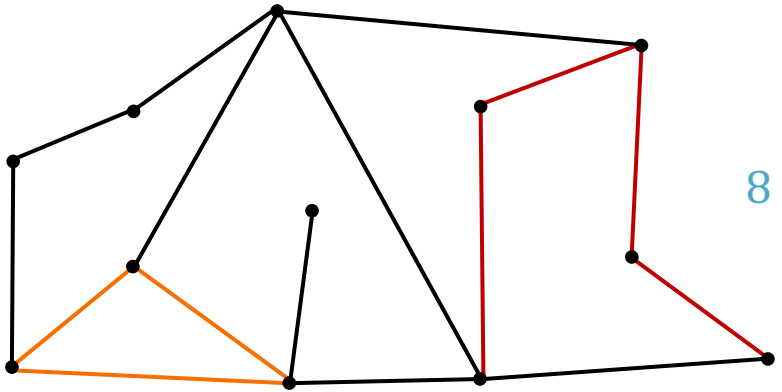
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

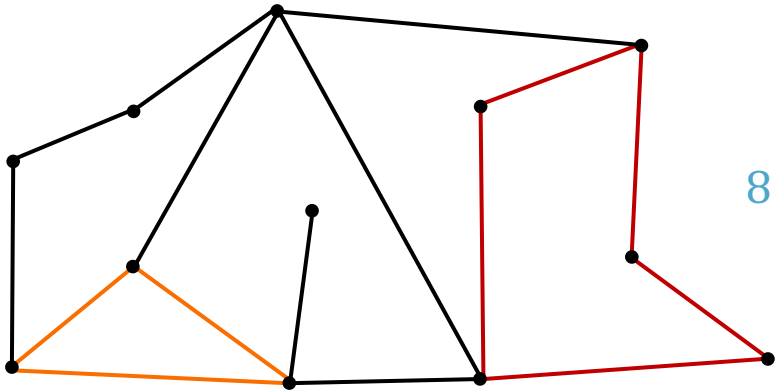
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

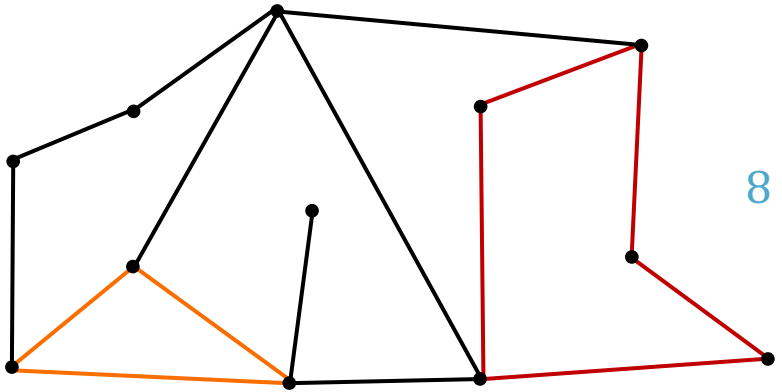
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

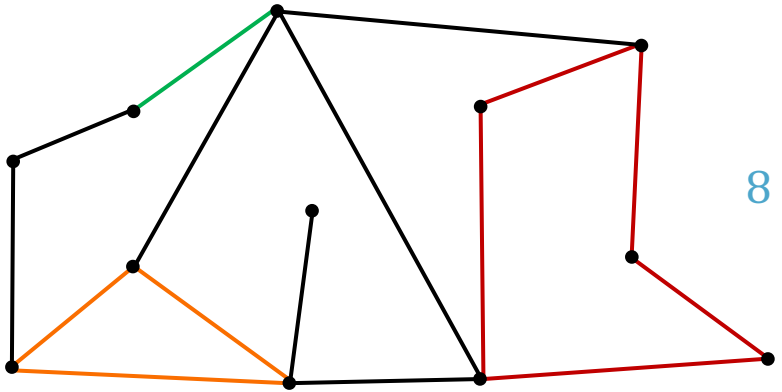
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

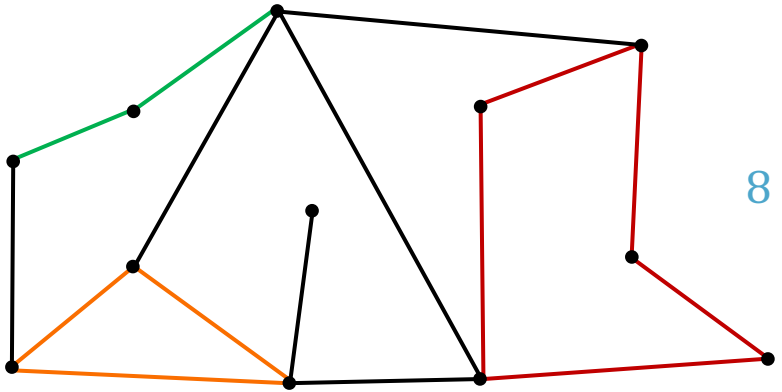
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

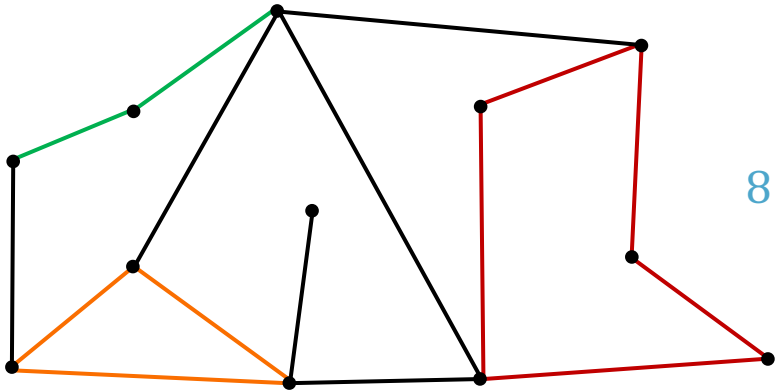
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

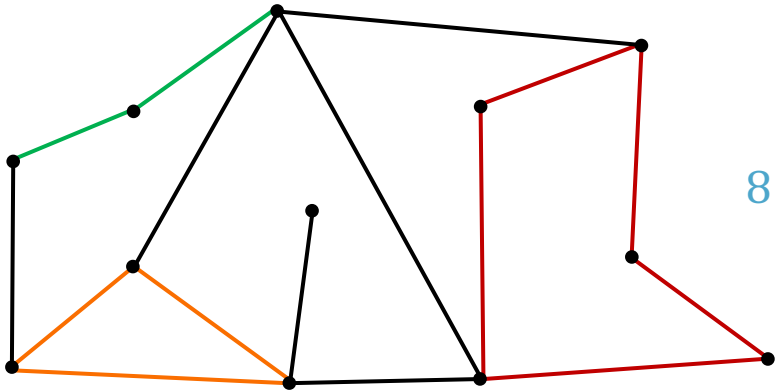
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

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

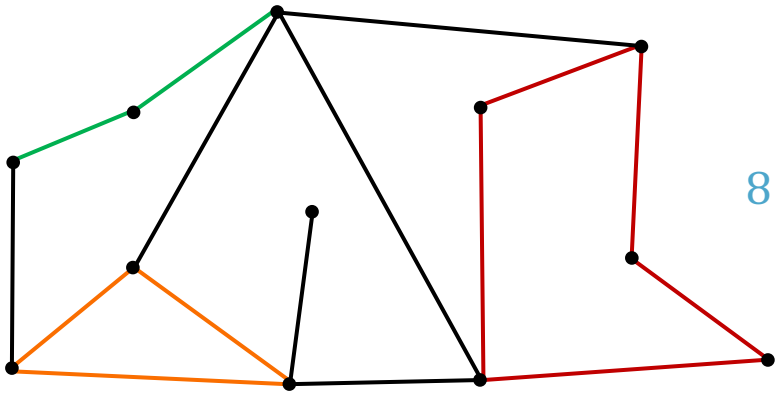
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



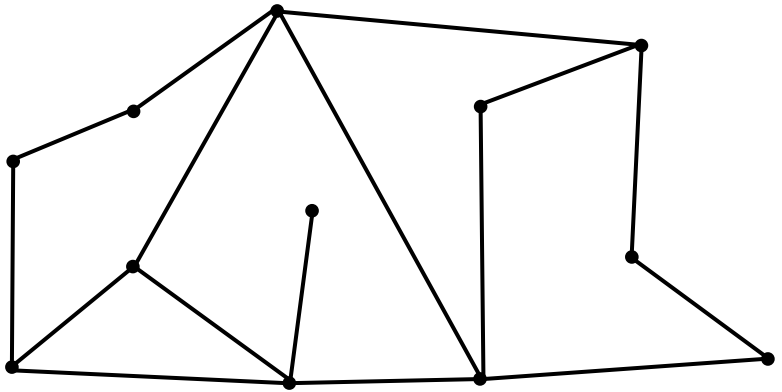
$8 \geq |VC| \geq 4$

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

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

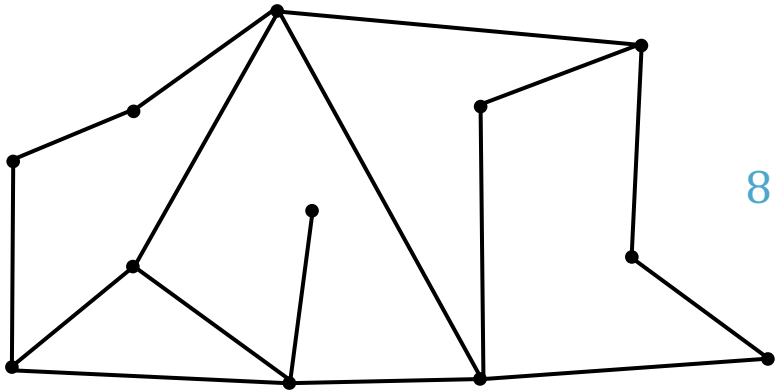
Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen



Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

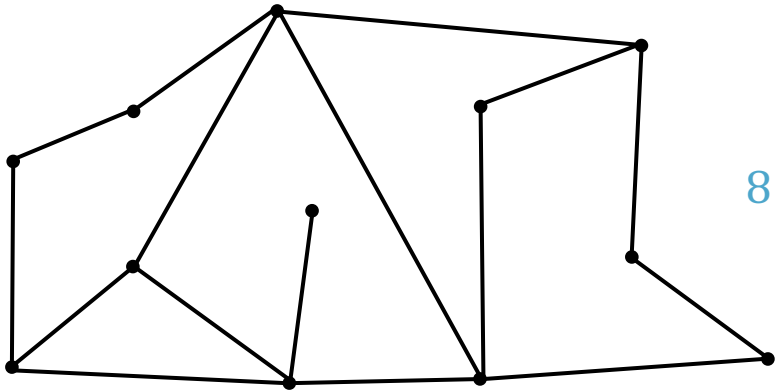


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

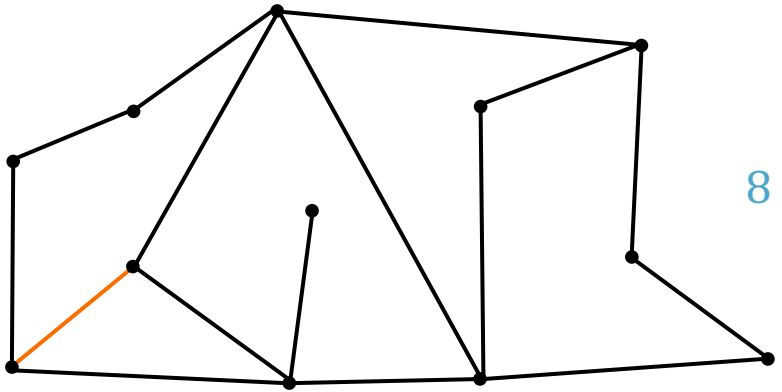


$$8 \geq |VC| \geq 4$$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

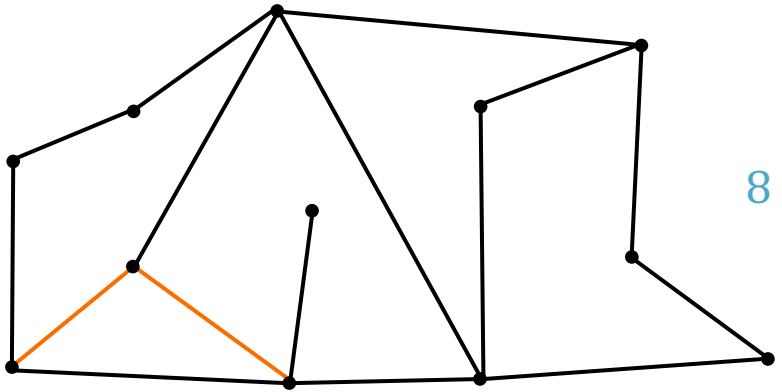


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!

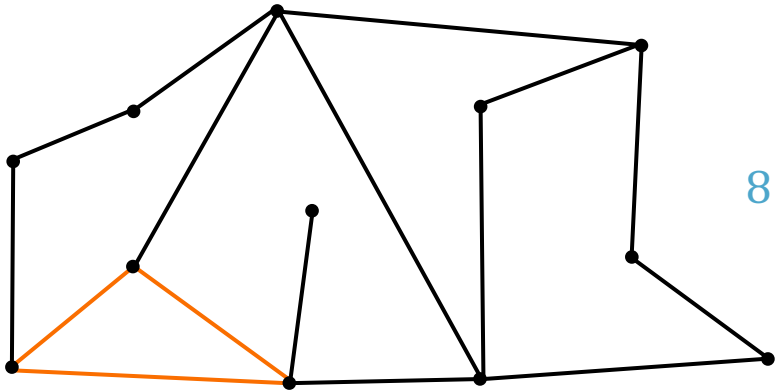


$8 \geq |VC| \geq 4$

Schranken Vertex Cover

- Vollständige Graphen
- Pfad Graphen
- Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

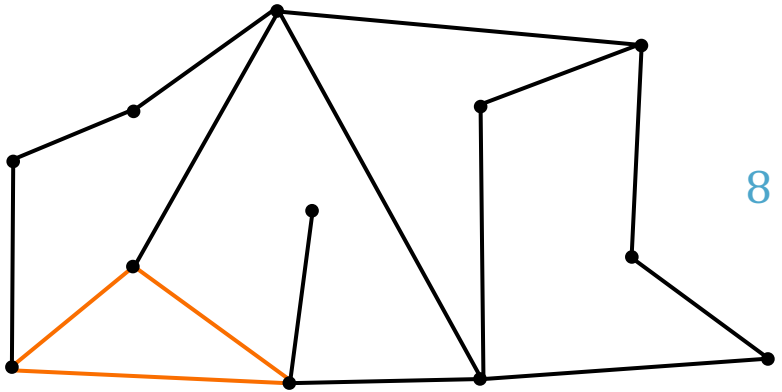
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

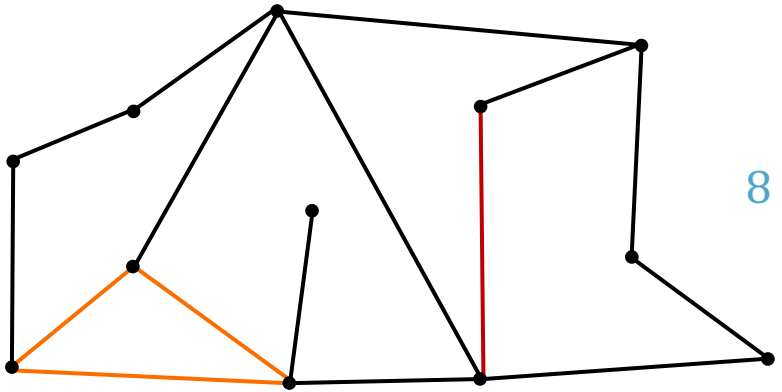
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

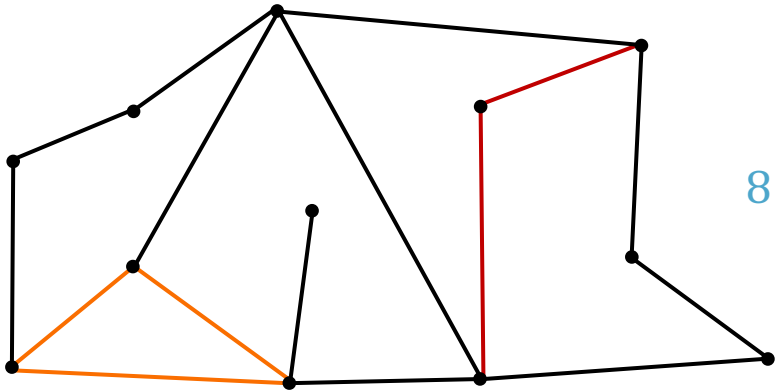
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

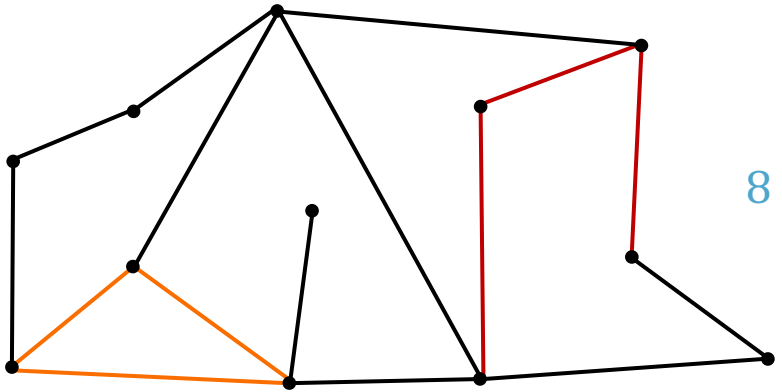
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

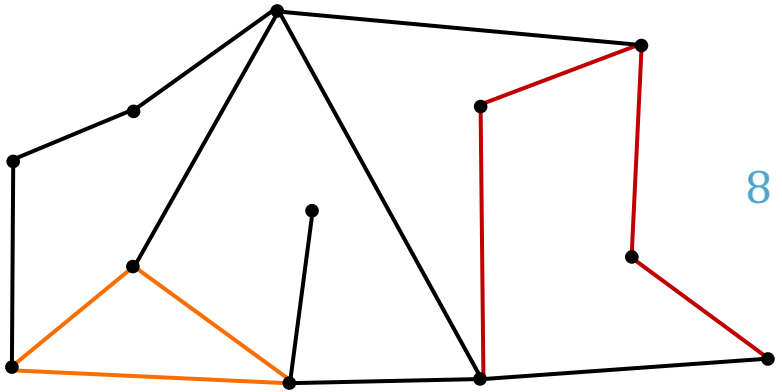
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

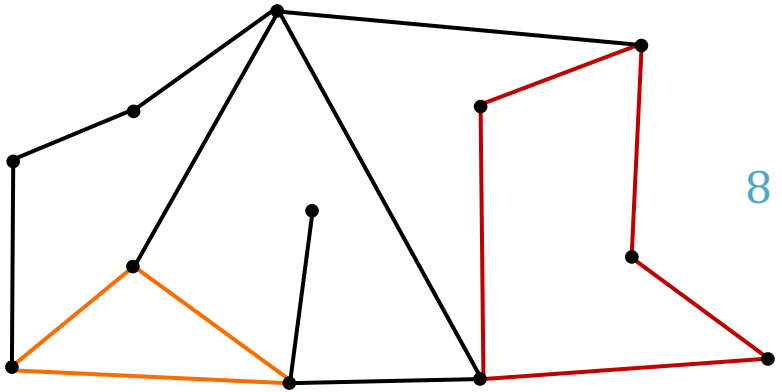
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

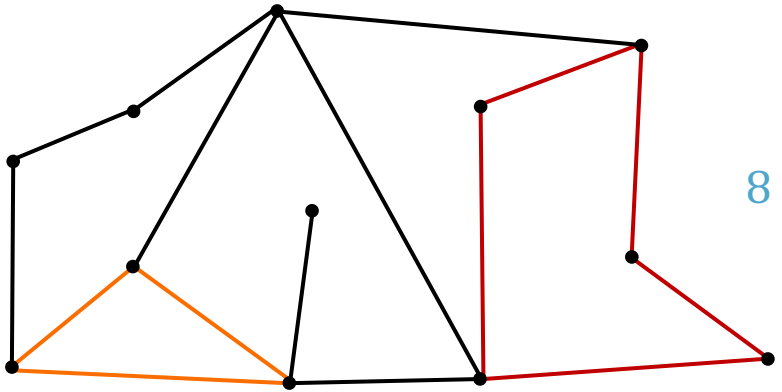
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

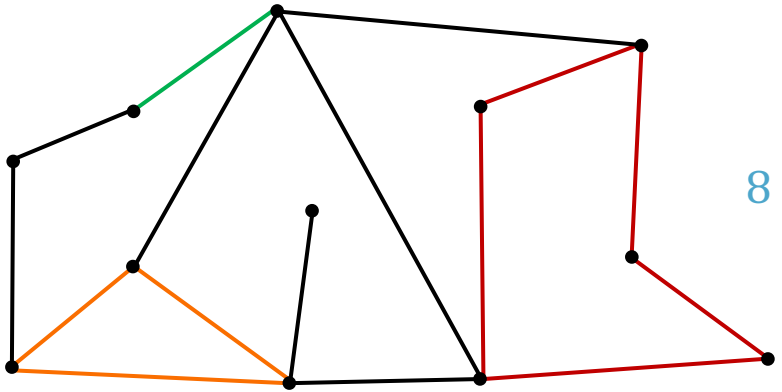
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

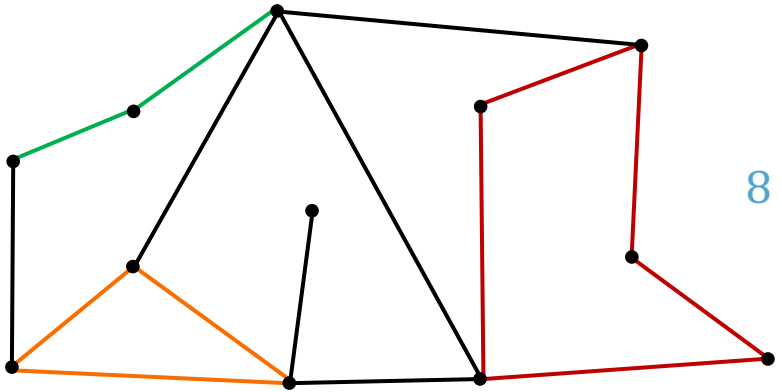
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

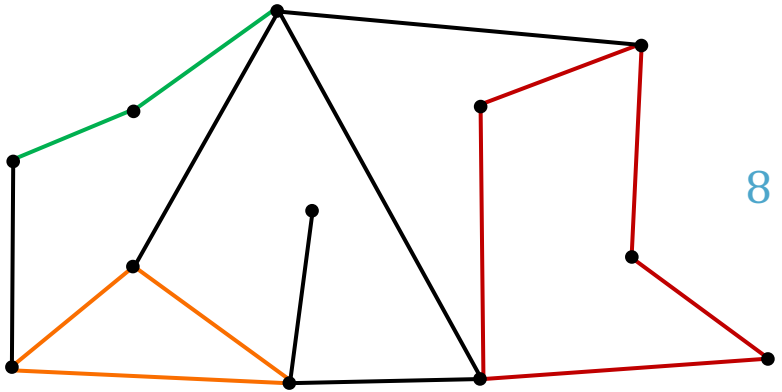
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

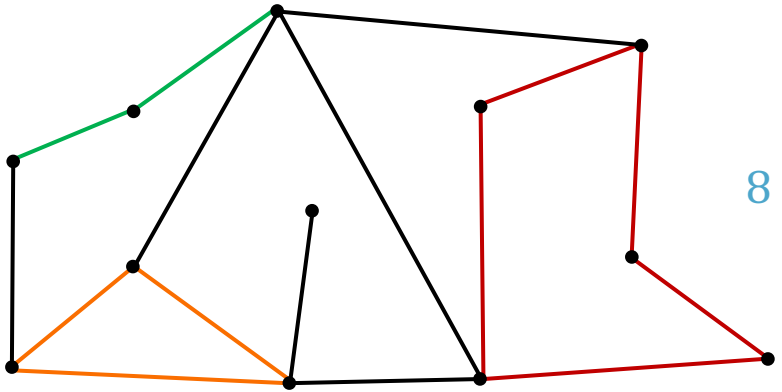
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

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

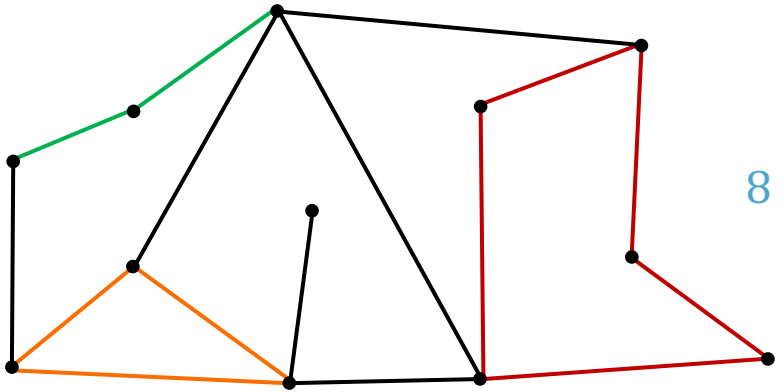
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$8 \geq |VC| \geq 4$

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

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

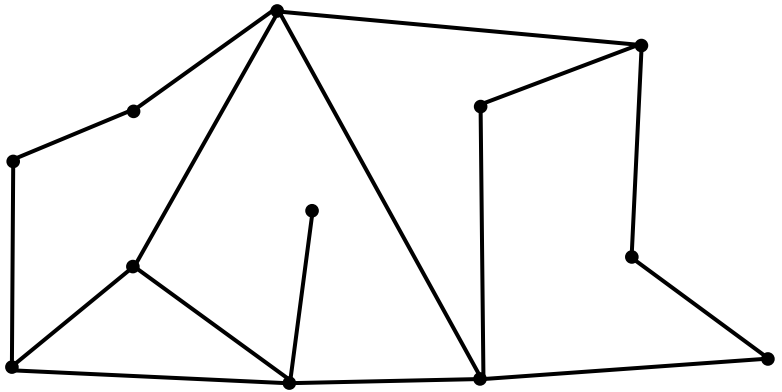
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

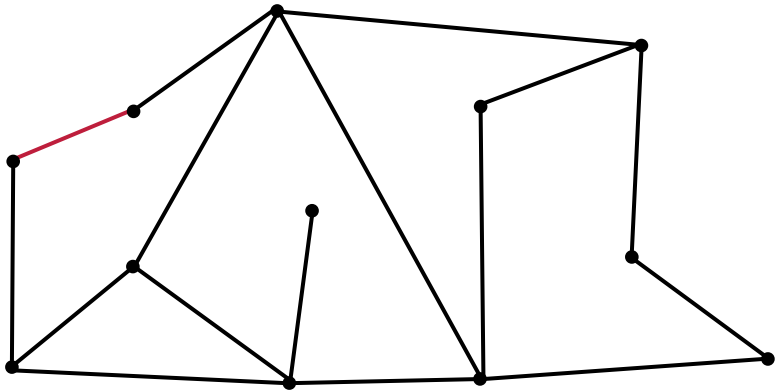
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

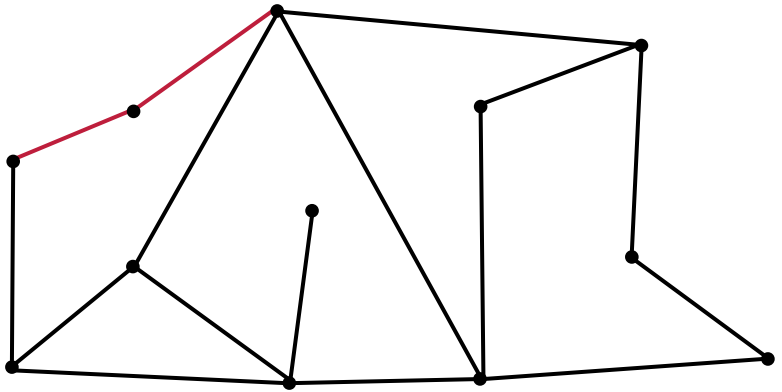
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

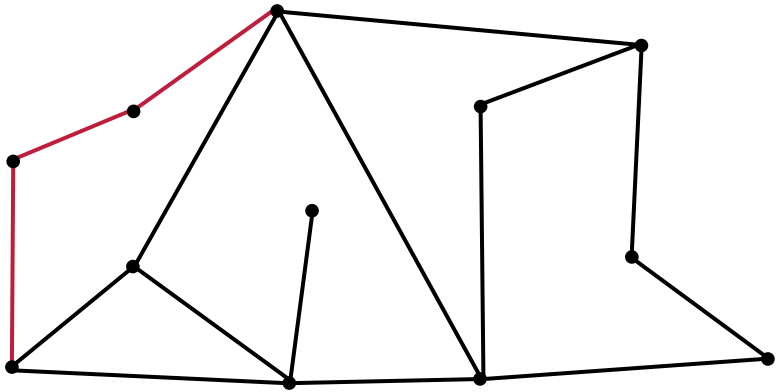
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

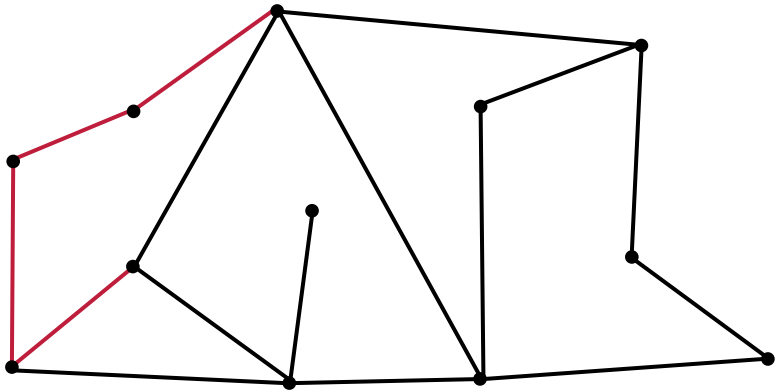
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

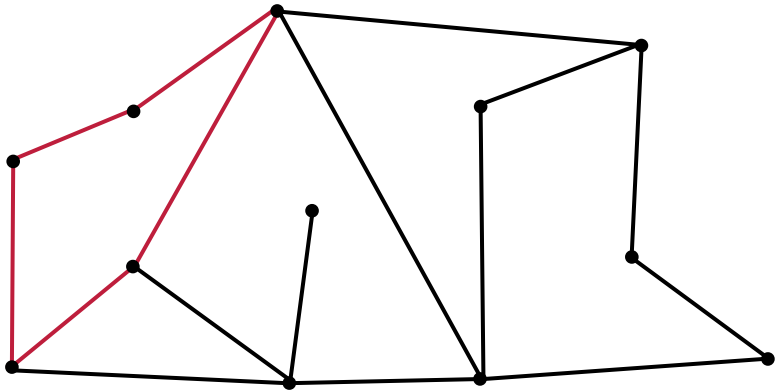
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

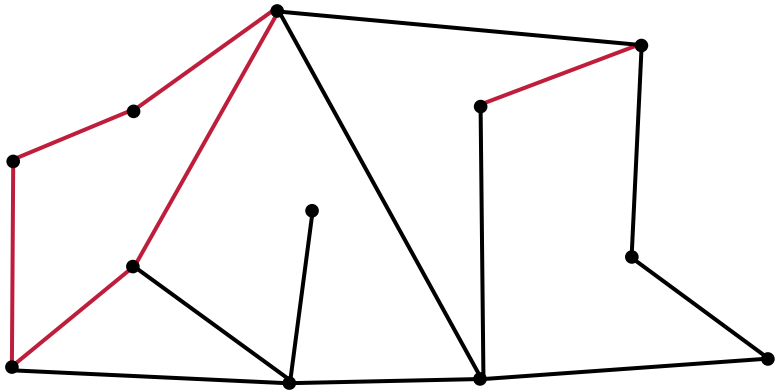
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

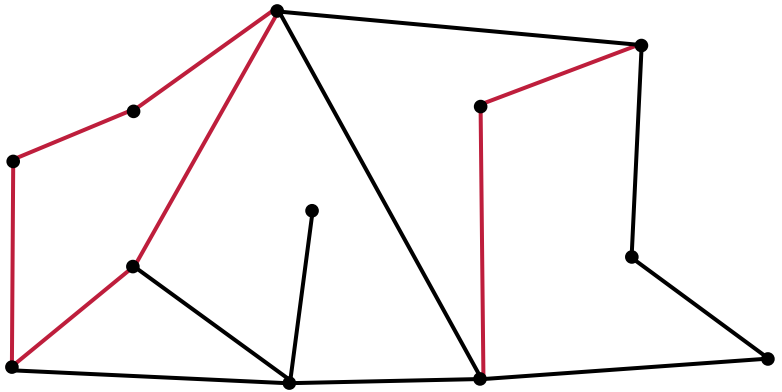
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

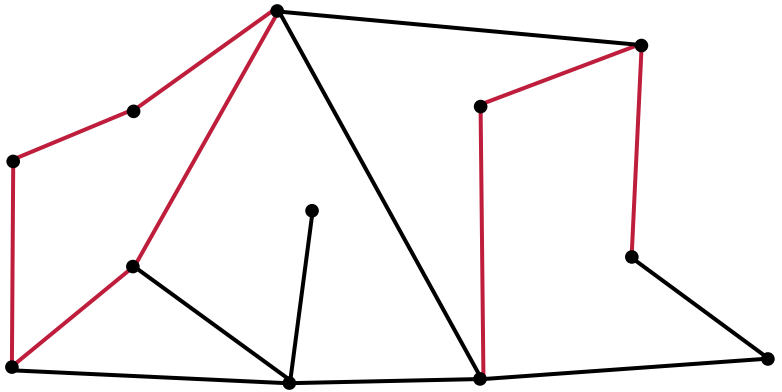
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

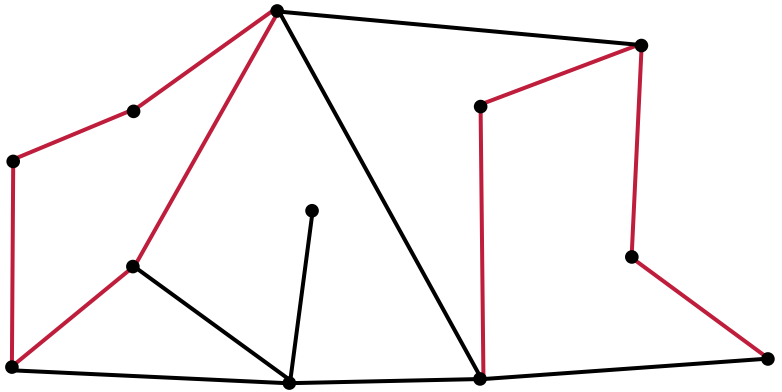
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

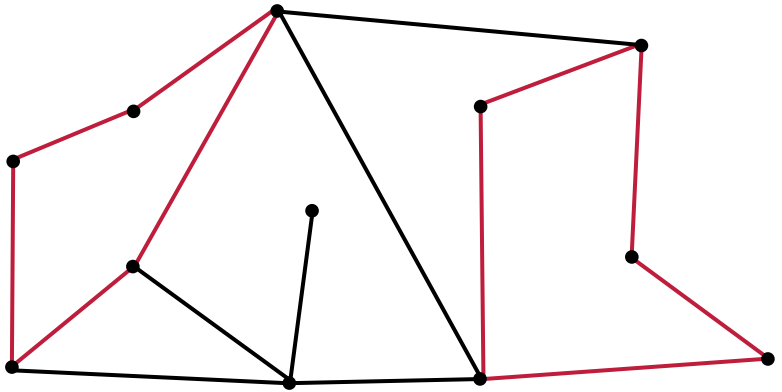
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

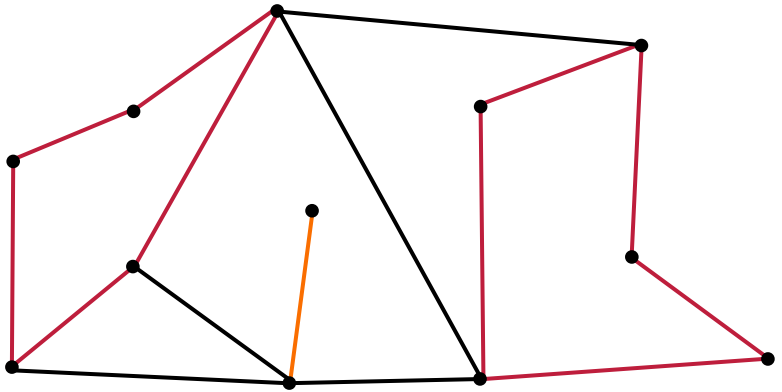
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



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

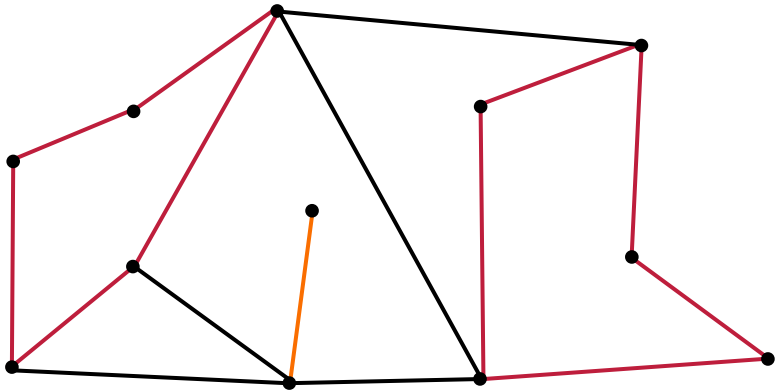
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$$8 \geq |VC| \geq 1 + 3 + 3 = 7$$

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

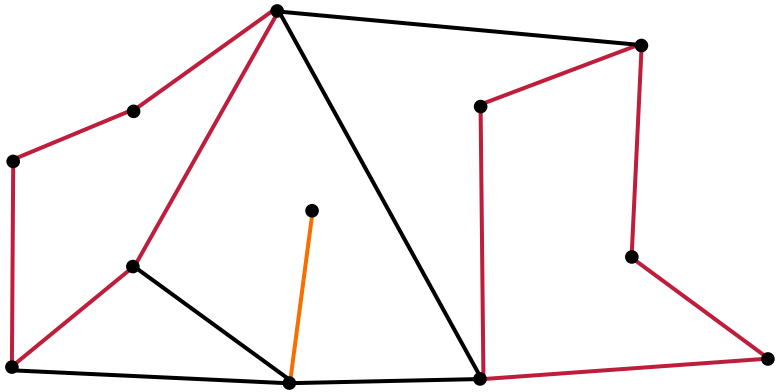
Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen

Teile Graph in einfache, unabhängige Teilgraphen!



$$8 \geq |VC| \geq 1 + 3 + 3 = 7$$

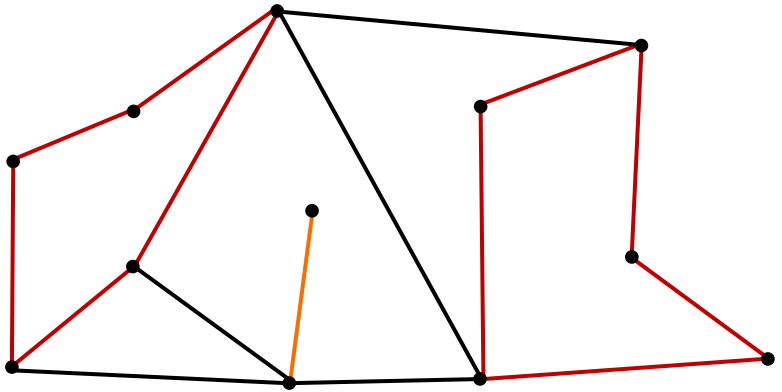
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



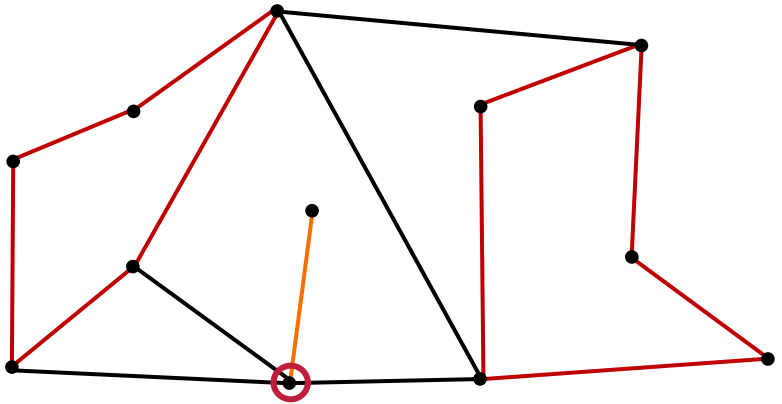
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



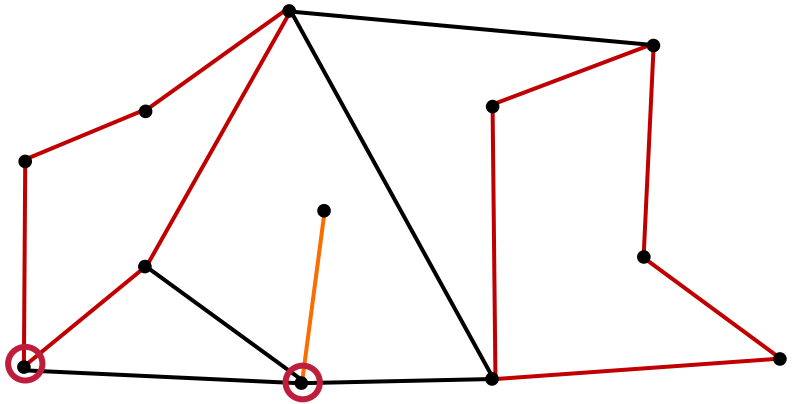
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



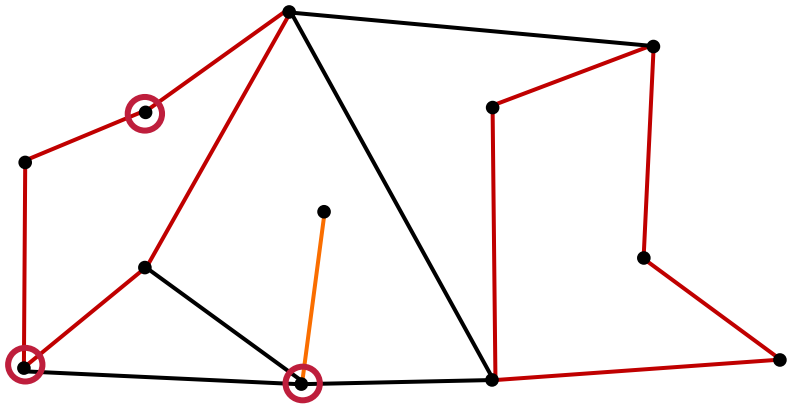
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



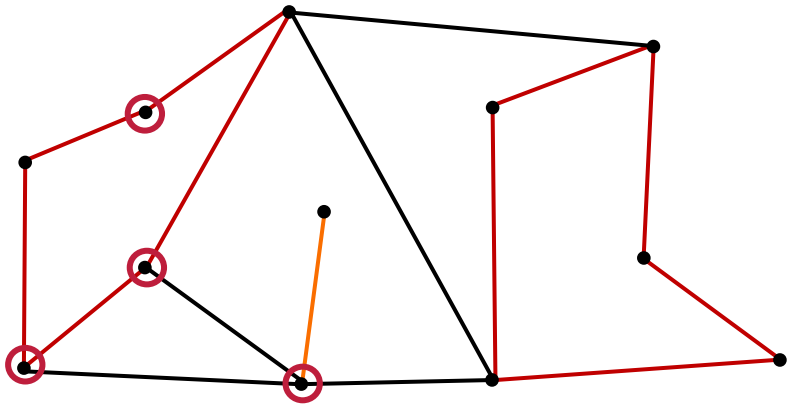
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



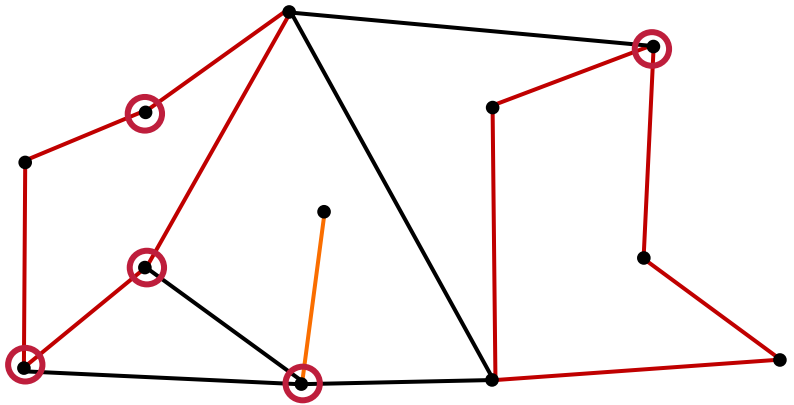
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



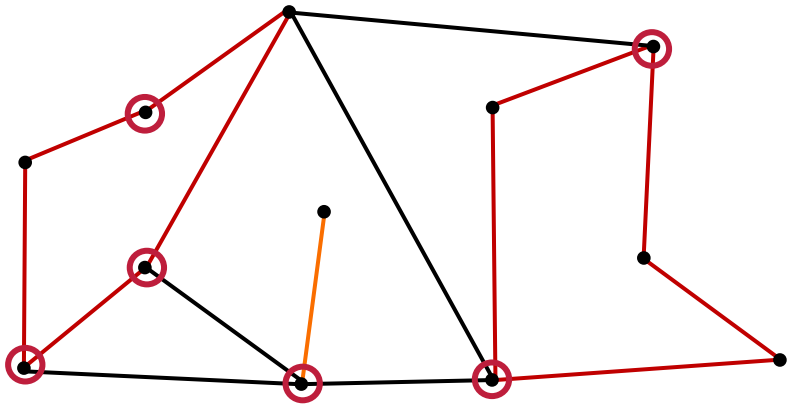
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



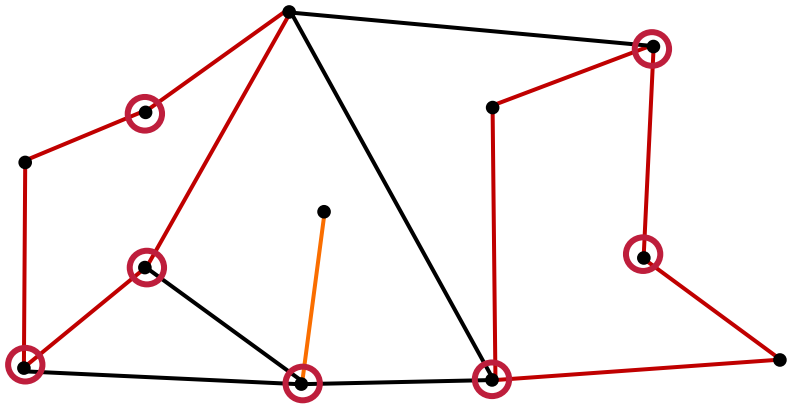
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



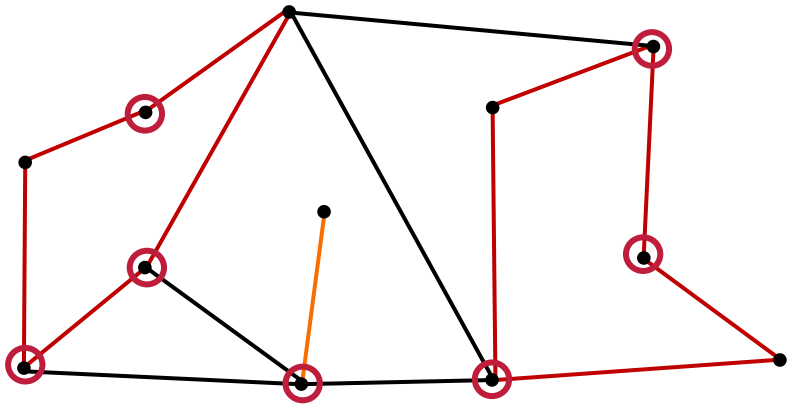
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder 8?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



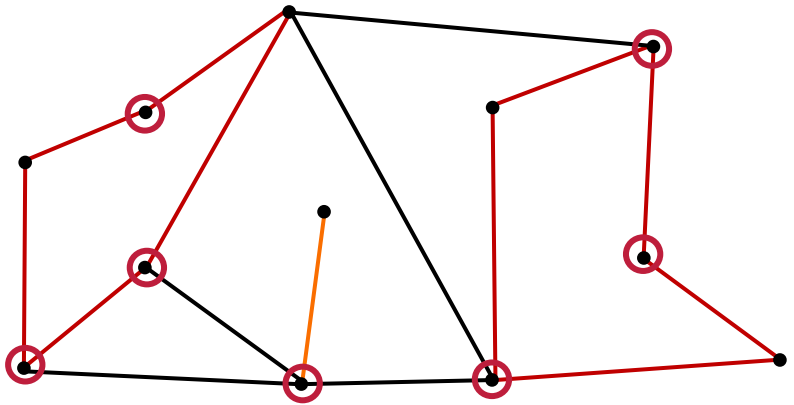
Wie groß ist ein kleinstes Vertex Cover? ~~6~~, 7 oder ~~8~~?

Schranken Vertex Cover

Vollständige Graphen

Pfad Graphen

Kreis Graphen



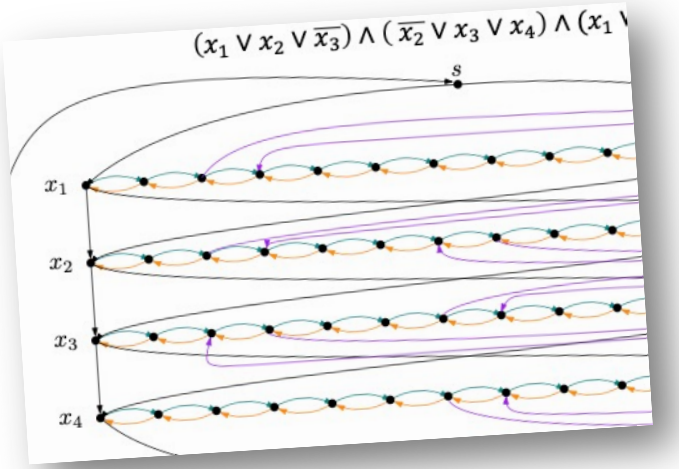
$$7 \geq |VC| \geq 1 + 3 + 3 = 7$$

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

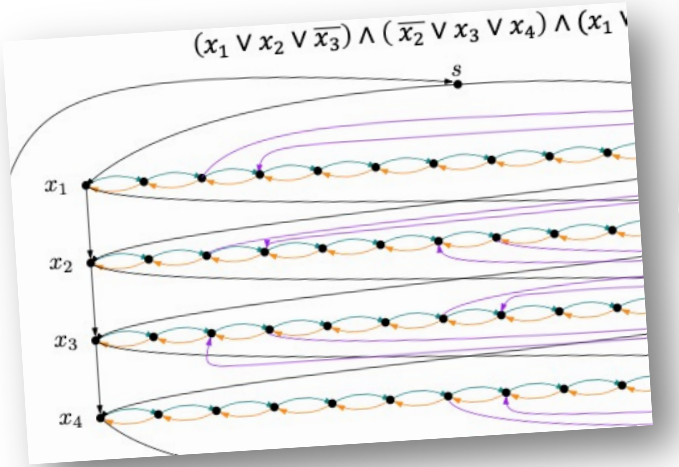
Fragen?

… und nächstes Mal …

… und nächstes Mal …



… und nächstes Mal …



Ein paar Probleme

3-SAT | DHC | HC

Traveling Salesman Problem TSP

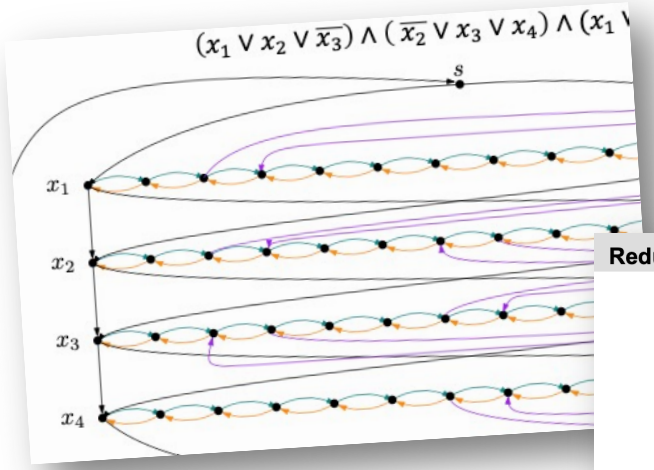
Gegeben

Vollsändiger Graph $G = (V, E)$ mit Kantenkosten $c: E \to \mathbb{R}^+$ und eine Zahl $k \in \mathbb{R}^+$

Frage

Gibt es eine Tour in G mit Kantenkosten maximal k?

… und nächstes Mal …



Ein paar Probleme

3-SAT DHC HC

Traveling Salesman Problem TSP

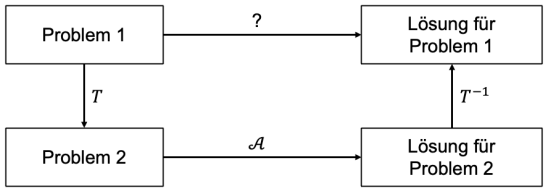
Gegeben

Vollständiger Graph $G = (V, E)$ mit Kantenkosten $c: E \to \mathbb{R}^+$ und eine Zahl $k \in \mathbb{R}^+$

Frage

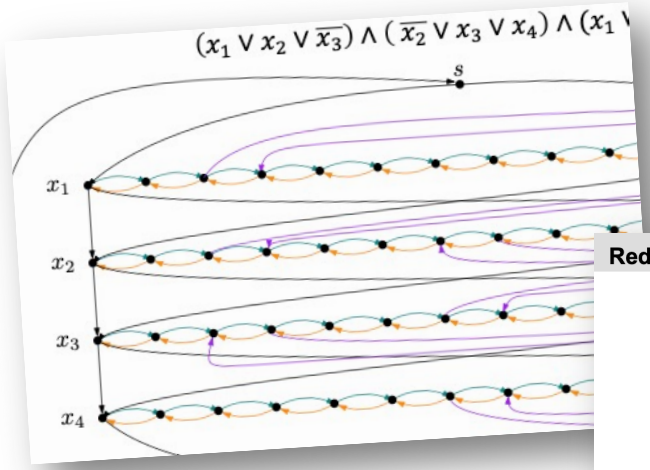
Gibt es eine Tour in G mit Kantenkosten

Reduktionen



Ist Problem 1 NP-schwer und besitzen T und T^{-1} polynomielle Laufzeit, dann muss Problem 2 auch NP-schwer sein.

… und nächstes Mal …



Ein paar Probleme

3-SAT DHC HC

Traveling Salesman Problem TSP

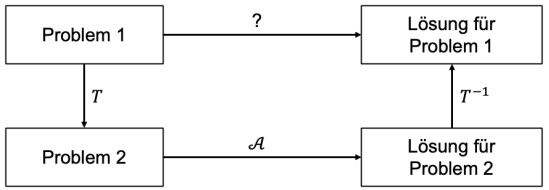
Gegeben

Vollsändiger Graph $G = (V, E)$ mit Kantenkosten $c: E \to \mathbb{R}^+$ und eine Zahl $k \in \mathbb{R}^+$

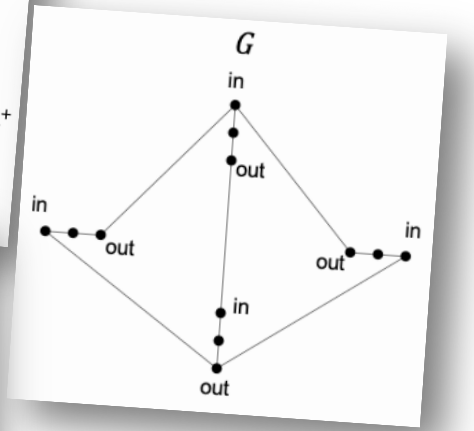
Frage

Gibt es eine Tour in G mit Kantenkosten

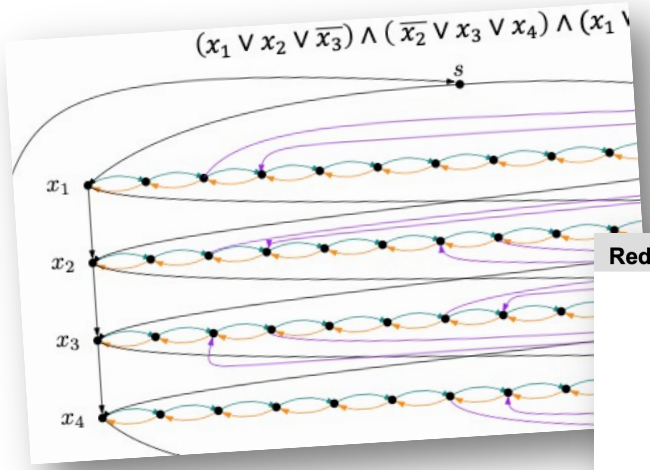
Reduktionen



Ist Problem 1 NP-schwer und besitzen T und T^{-1} polynomielle Laufzeit, dann muss Problem 2 auch NP-schwer sein.



… und nächstes Mal …



Ein paar Probleme

3-SAT DHC HC

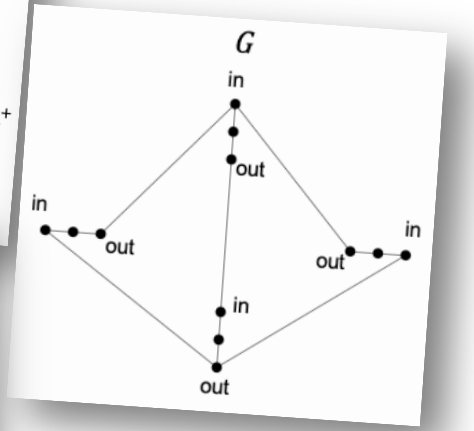
Traveling Salesman Problem TSP

Gegeben

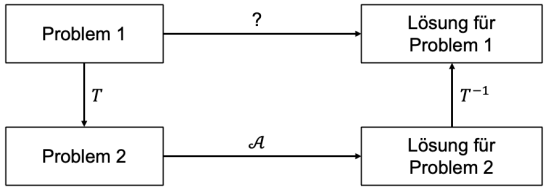
Vollsändiger Graph $G = (V, E)$ mit Kantenkosten $c: E \to \mathbb{R}^+$ und eine Zahl $k \in \mathbb{R}^+$

Frage

Gibt es eine Tour in G mit Kantenkosten



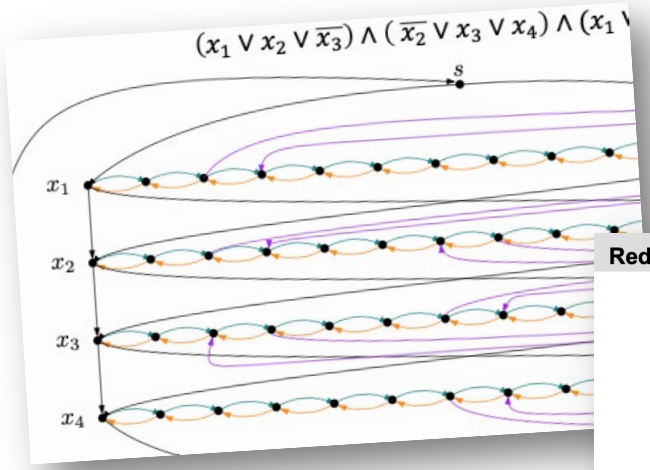
Reduktionen



Ist Problem 1 NP-schwer und besitzen T und T^{-1} polynomielle Laufzeit, dann muss Problem 2 auch NP-schwer sein.

… gibt es Komplexität & Reduktionen! 🔥🔥🔥

… und nächstes Mal …



Ein paar Probleme

3-SAT DHC HC

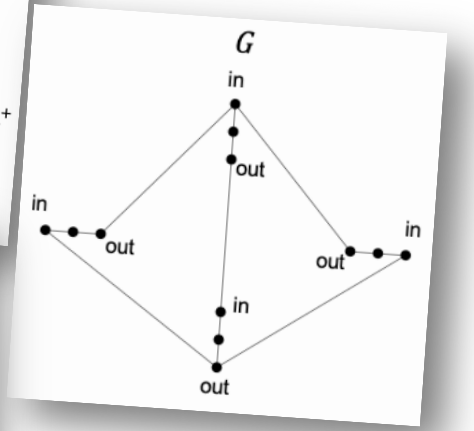
Traveling Salesman Problem TSP

Gegeben

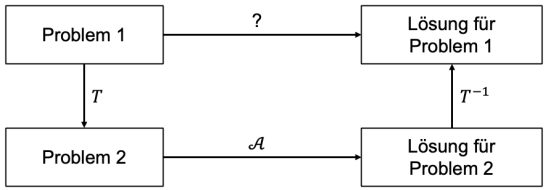
Vollständiger Graph $G = (V, E)$ mit Kantenkosten $c: E \to \mathbb{R}^+$ und eine Zahl $k \in \mathbb{R}^+$

Frage

Gibt es eine Tour in G mit Kantenkosten



Reduktionen



Ist Problem 1 NP-schwer und besitzen T und T^{-1} polynomielle Laufzeit, dann muss Problem 2 auch NP-schwer sein.

… gibt es Komplexität & Reduktionen! 🔥🔥🔥

am 26. Juni
(in zwei Wochen)