Karlsruher Institut für Technologie

Center Selection Problem

Adrian Cierpka

Institut für Theoretische Informatik - Proseminar NP-schwere Probleme

Übersicht

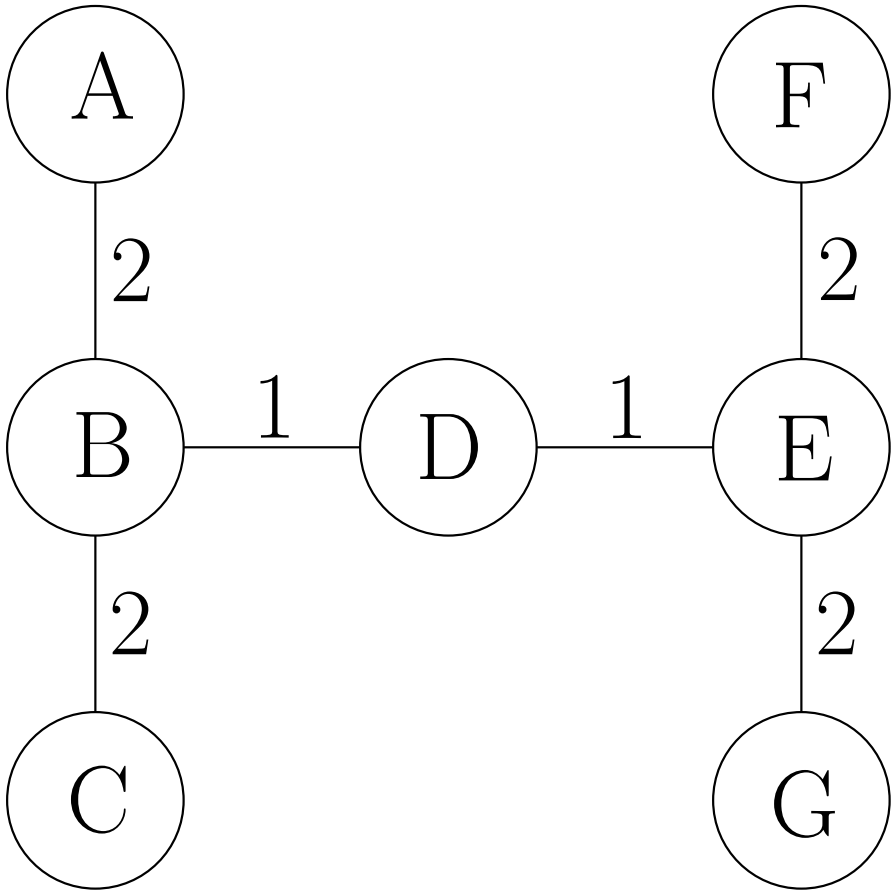
- Beispiel
- Formale Definition
- Algorithmen
  - Allgemeines
  - Pseudocode
  - Beispiel
  - Beweis der Korrektheit
- Beweis zur relativen 2-Approximation
  - Dominating Set

Motivation

- Wofür benötigt man das Center Selection Problem?
  - Platzierung von Einkauszentren
  - Standort für Server finden
  - Gute Mobilfunkabdeckung erreichen
  - Und vieles mehr ...

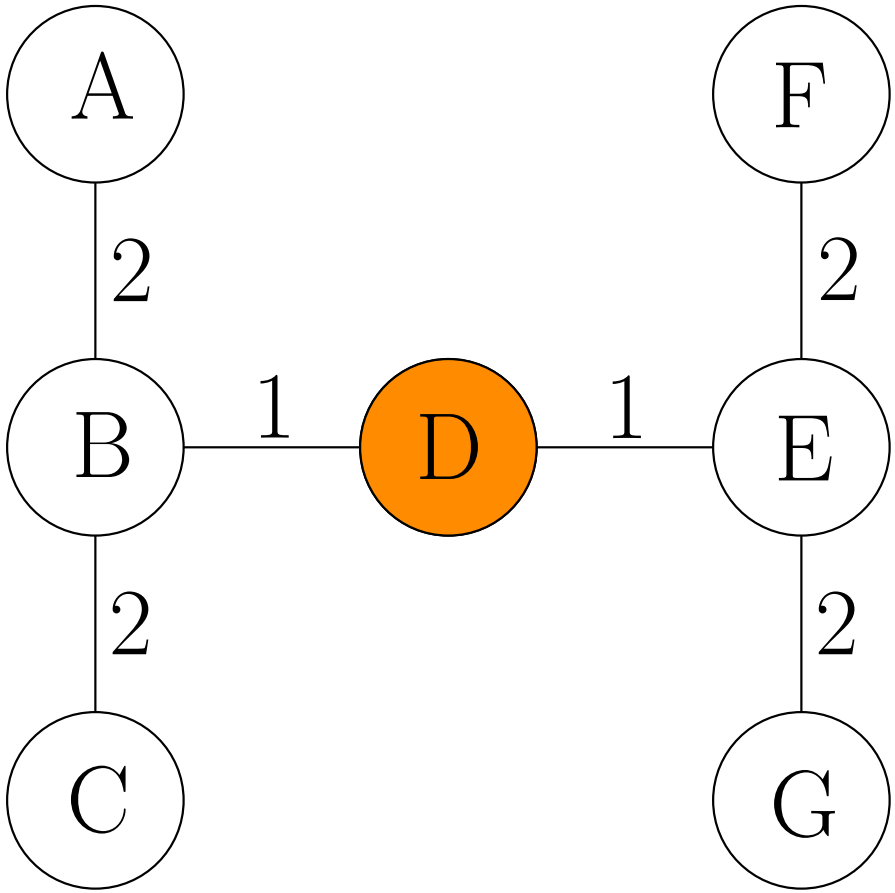
Beispiel 1

Ein Center soll gesetzt werden



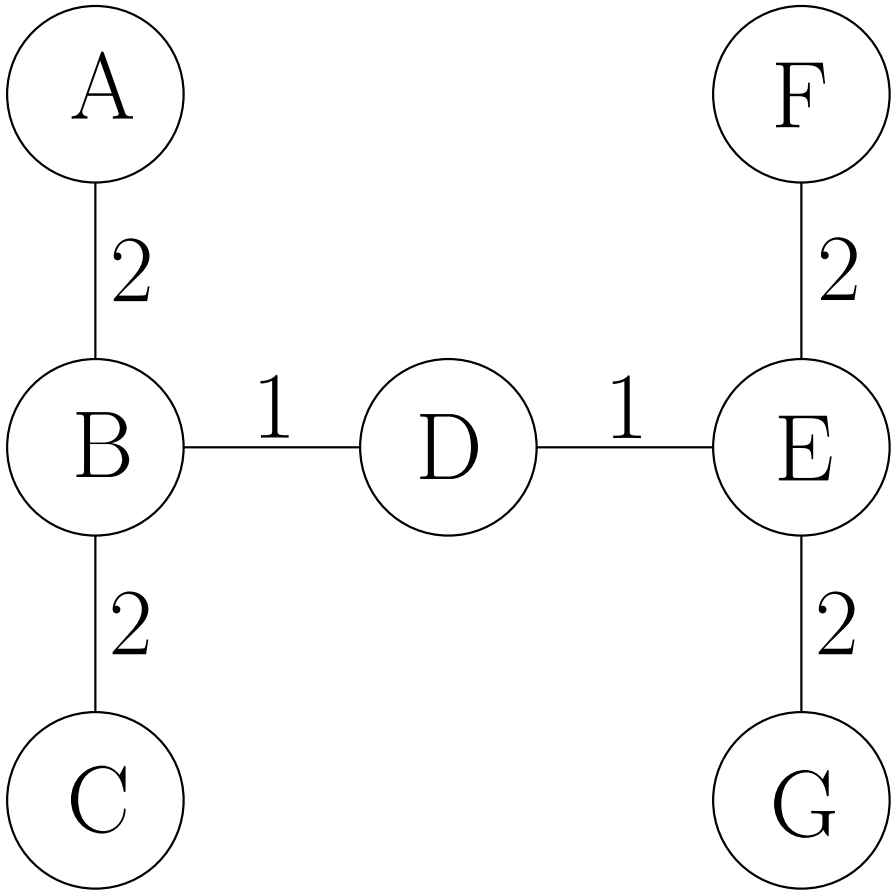
Beispiel 1

Optimale Lösung mit Radius = 3



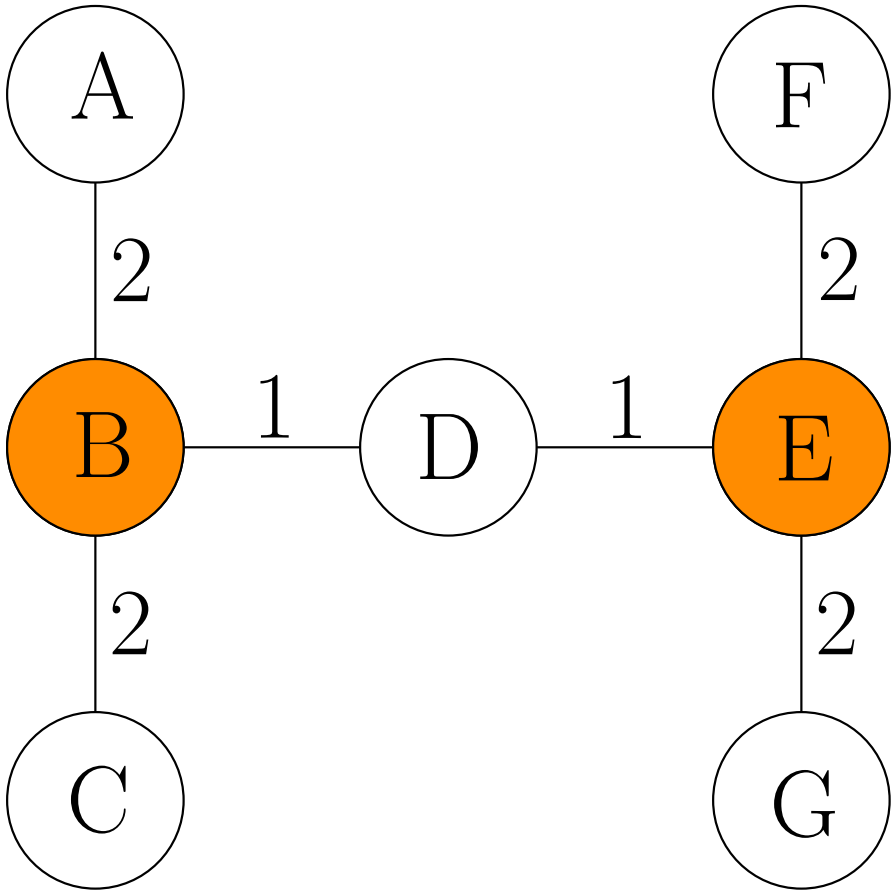
Beispiel 2

Zwei Center sollen gesetzt werden



Beispiel 2

Optimale Lösung mit Radius = 2



Formale Definition

- Gegeben: ungerichteter, gewichteter Graph $G = (V, E)$ und ein $k \in \mathbb{N}$
- Gesucht: Menge $C \subseteq V$ mit $|C| = k$, sodass Radius $r^C(V) = \max_{v \in V} dist(v, C)$ minimal
  - $dist(v, C) = \min_{c \in C} dist(v, c)$
  - $dist(v, u)$ (wobei $v, u \in V$) ist die Länge des kürzesten Weges von v zu u
- Alternativ:
  Gesucht: Menge $C \subseteq V$ mit $|C| = k$, sodass Radius r minimal ist und $\forall\, v \in V : r \geq dist(v, C)$

Formale Definition

- Gegeben: ungerichteter, gewichteter Graph $G = (V, E)$ und ein $k \in \mathbb{N}$
- Gesucht: Menge $C \subseteq V$ mit $|C| = k$, sodass Radius $r^C(V) = \max_{v \in V} dist(v, C)$ minimal
  - $dist(v, C) = \min_{c \in C} dist(v, c)$
  - $dist(v, u)$ (wobei $v, u \in V$) ist die Länge des kürzesten Weges von v zu u
- Alternativ:
  Gesucht: Menge $C \subseteq V$ mit $|C| = k$, sodass Radius r minimal ist und $\forall\, v \in V : r \geq dist(v, C)$

Algorithmen

- zwei Approximationsalgorithmen
- beide relative Gütegarantie von 2
- polynomielle Laufzeit
- Erster Algorithmus:
  - benötigt den Radius einer optimalen Lösung
- Zweiter Algorithmus:
  - baut auf Erkenntnissen von dem ersten Algorithmus auf
  - benötigt **keinen** optimalen Radius

Erster Algorithmus: Allgemeines

- Approximationsalgorithmus mit polynomieller Laufzeit
- relative Gütegarantie von 2
- benötigt neben Graph $G = (V, E)$ und $k \in \mathbb{N}$ den optimalen Radius $r^C(V)$

Erster Algorithmus: Grundsätzliche Idee

Was bringt es, den optimalen Radius zu kennen?

Wir definieren

- "c in der Nähe von c^*" $\Leftrightarrow dist(c, c^*) \leq r$
- C^* ist Menge der Center und r der Radius einer optimalen Lösung auf einem Graphen $G = (V, E)$ mit $|C^*| = k$

Wir wissen

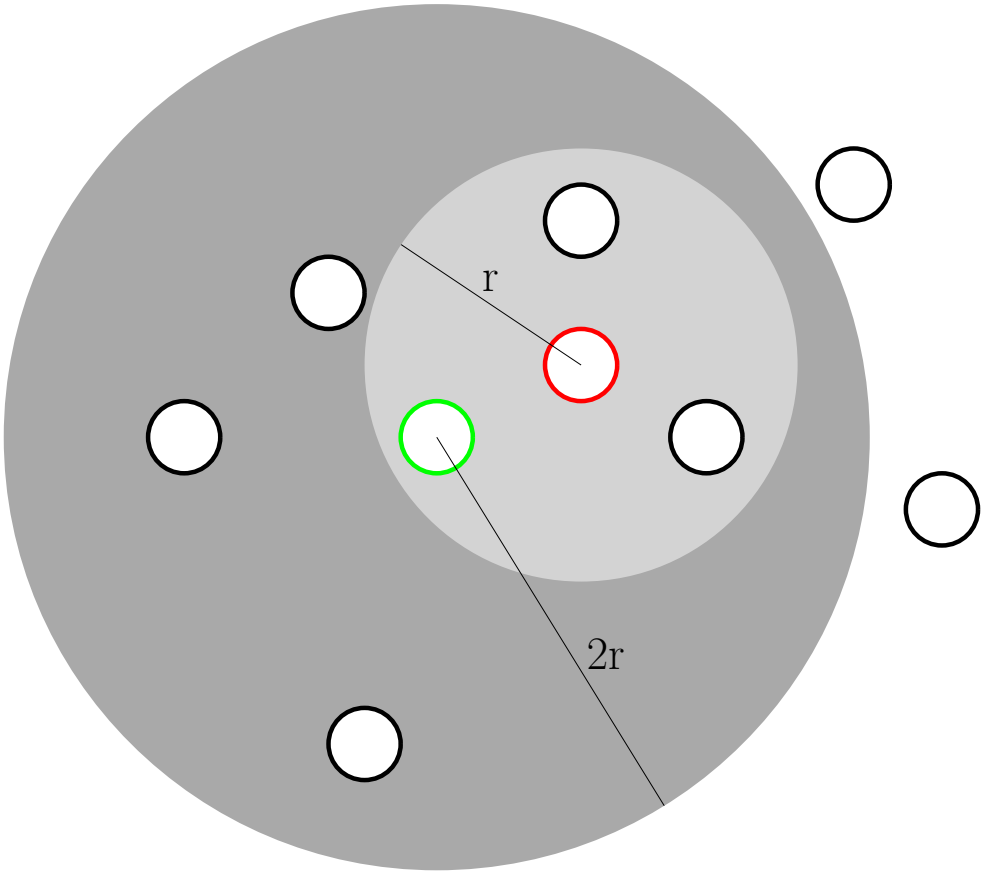
Für alle Knoten v gilt: $dist(v, C^*) \leq r$

Erster Algorithmus: Grundsätzliche Idee

- Veranschaulichung:
- Es wird ein zufälliger Knoten $v \in V$ gewählt
  - v hat mindestens ein Center $c^* \in C^*$ in der Nähe
  - Wie groß muss der Radius eines Centers auf v sein, um alle Knoten abzudecken, die auch von c^* abgedeckt wurden?
  - Lösung: $2r$

Erster Algorithmus: Grundsätzliche Idee

Veranschaulichung auf euklidischer Ebene:



Grün: ein beliebiger Punkt *v*

Rot: Center *c** aus der optimalen Lösung und in der Nähe von *v*

Erster Algorithmus: Pseudocode

V' will represent the vertices that still need to be covered;
Initialize $V' = V$;
Let $C \neq \emptyset$;
while $V' \neq \emptyset$ **do**
    Select any vertex $v \in V'$ and add v to C ;
    Delete all vertices from V' that are at distance at most $2r$ from v ;
end
if $|C| \leq k$ **then**
    Return C as the selected set of vertices;
else
    Claim (correctly) that there is no set of k centers with covering radius at most r;
end

Erster Algorithmus: Pseudocode

V' will represent the vertices that still need to be covered;
Initialize $V' = V$;
Let $C \neq \emptyset$;
while $V' \neq \emptyset$ **do**
    Select any vertex $v \in V'$ and add v to C ;
    Delete all vertices from V' that are at distance at most $2r$ from v ;
end
if $|C| \leq k$ **then**
    Return C as the selected set of vertices;
else
    Claim (correctly) that there is no set of k centers with covering radius at most r;
end

Erster Algorithmus: Pseudocode

V' will represent the vertices that still need to be covered;
Initialize $V' = V$;
Let $C \neq \emptyset$;
while $V' \neq \emptyset$ **do**
  Select any vertex $v \in V'$ and add v to C ;
  Delete all vertices from V' that are at distance at most $2r$ from v ;
end
if $|C| \leq k$ **then**
  Return C as the selected set of vertices;
else
  Claim (correctly) that there is no set of k centers with covering radius at most r;
end

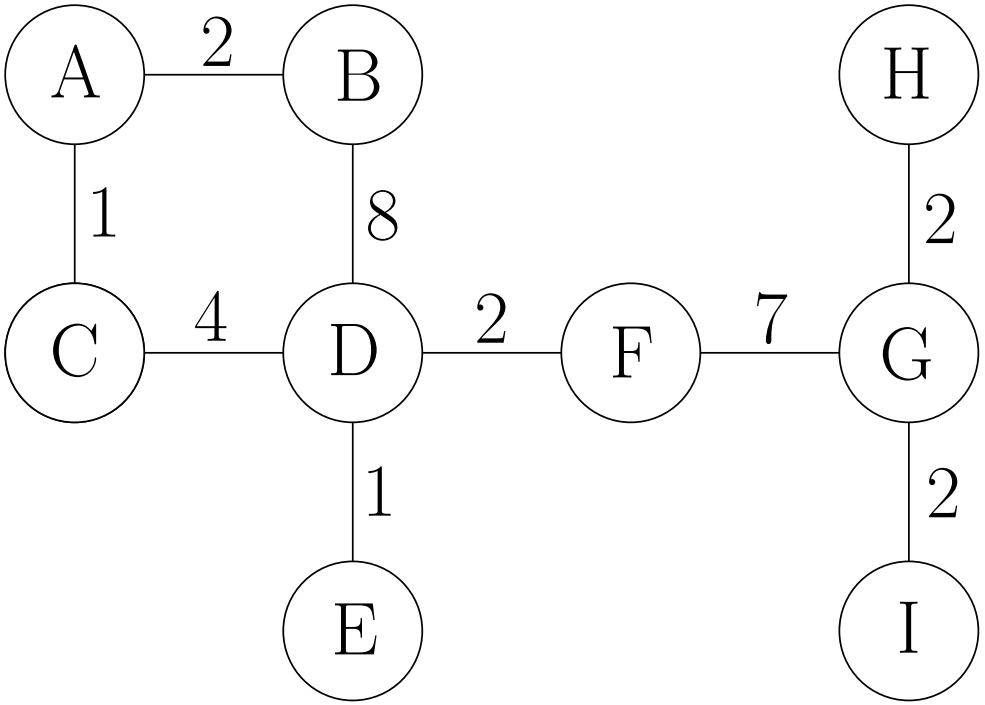
Erster Algorithmus: Pseudocode

V' will represent the vertices that still need to be covered;
Initialize $V' = V$;
Let $C \neq \emptyset$;
while $V' \neq \emptyset$ **do**
  Select any vertex $v \in V'$ and add v to C ;
  Delete all vertices from V' that are at distance at most $2r$ from v ;
end
if $|C| \leq k$ **then**
  Return C as the selected set of vertices;
else
  Claim (correctly) that there is no set of k centers with covering radius at most r;
end

Erster Algorithmus: Pseudocode

V' will represent the vertices that still need to be covered;
Initialize $V' = V$;
Let $C \neq \emptyset$;
while $V' \neq \emptyset$ **do**
  Select any vertex $v \in V'$ and add v to C ;
  Delete all vertices from V' that are at distance at most $2r$ from v ;
end
if $|C| \leq k$ **then**
  Return C as the selected set of vertices;
else
  Claim (correctly) that there is no set of k centers with covering radius at most r;
end

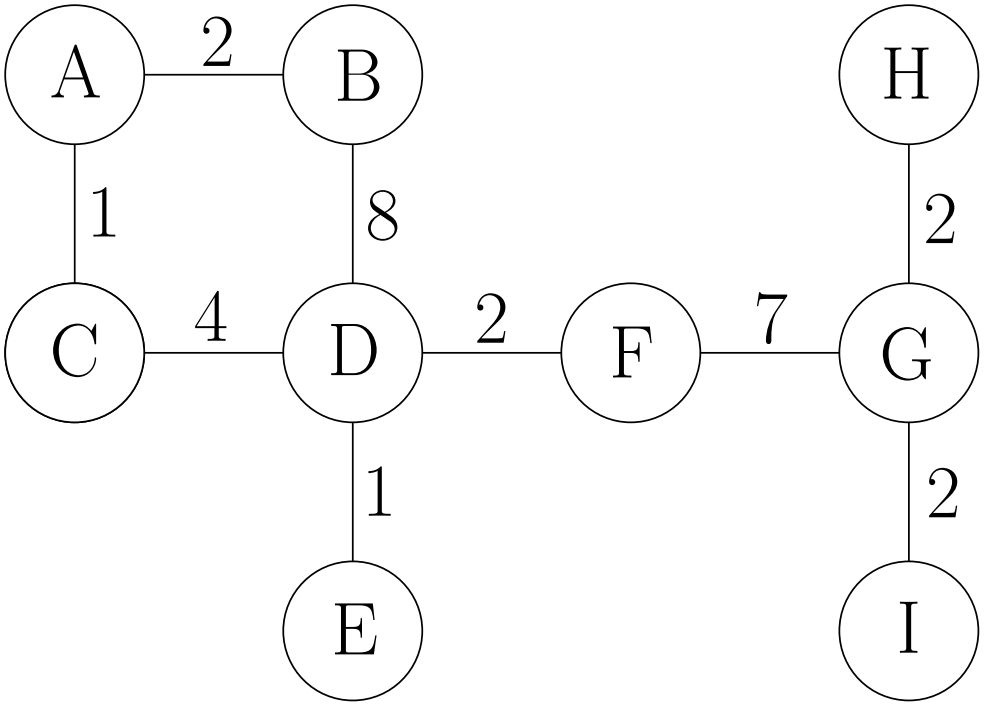
Erster Algorithmus: Beispiel



- Sei $G = (V, E)$ der links dargestellte Graph
- Sei $k = 4$
- Der optimale Radius ist $r^C(V) = 2$

V' will represent the vertices that still need to be covered;
Initialize $V' = V$;
Let $C \neq \emptyset$;
. . .

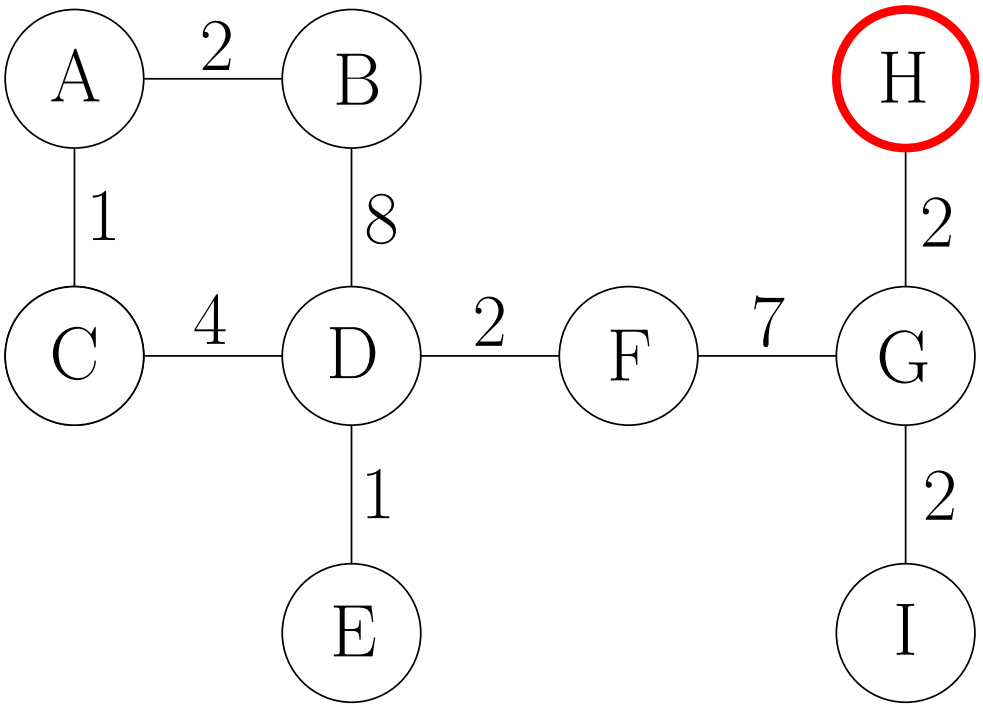
Erster Algorithmus: Beispiel



$V' = \{A, B, C, D, E, F, G, H, I\}$
$C = \{\}$

V' will represent the vertices that still need to be covered;
Initialize $V' = V$;
Let $C \neq \emptyset$;
. . .

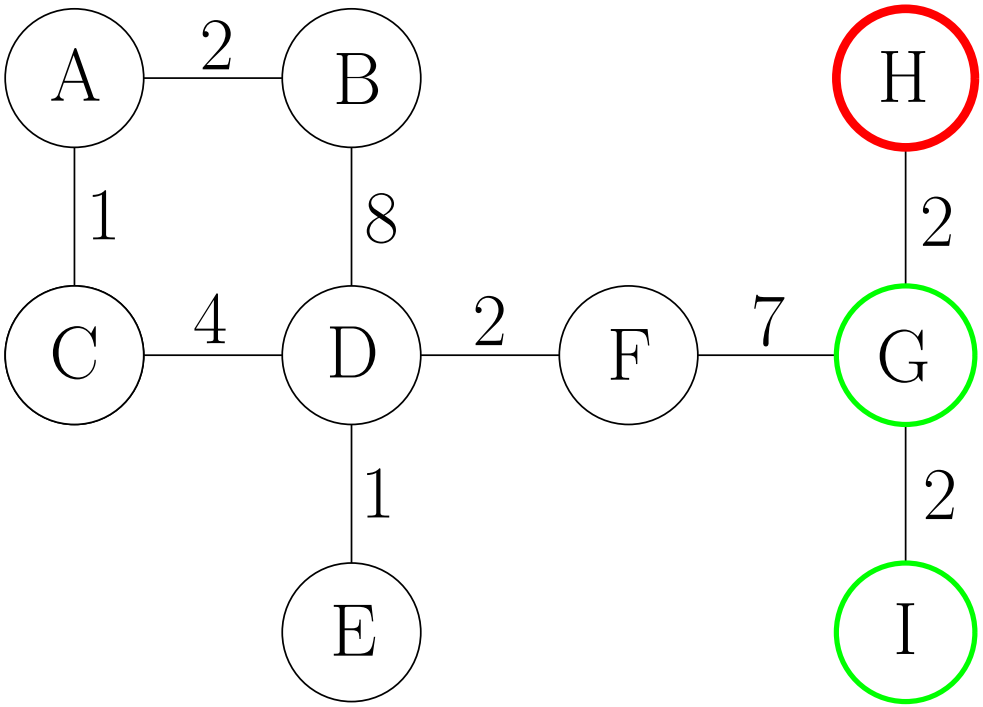
Erster Algorithmus: Beispiel



$V' = \{A, B, C, D, E, F, G, I\}$
$C = \{H\}$

while $V' \neq \emptyset$ **do**
  Select any vertex $v \in V'$ and add v to C ;
  Delete all vertices from V' that are at distance at most $2r$ from v ;
end

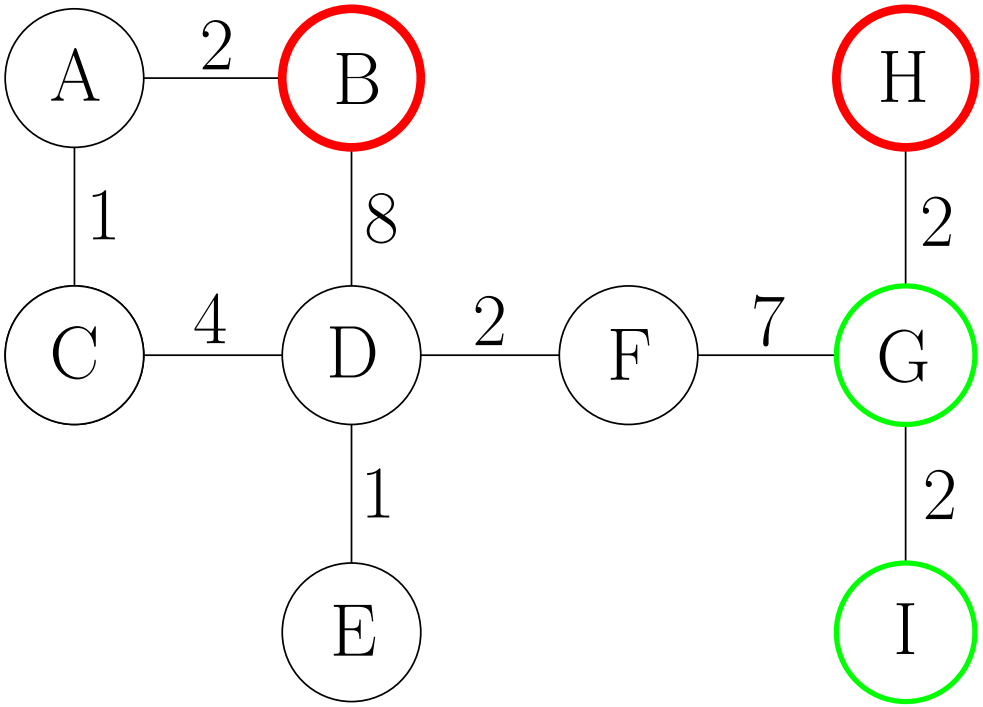
Erster Algorithmus: Beispiel



$V' = \{A, B, C, D, E, F\}$
$C = \{H\}$

while $V' \neq \emptyset$ **do**
  Select any vertex $v \in V'$ and add v to C ;
  Delete all vertices from V' that are at distance at most $2r$ from v ;
end

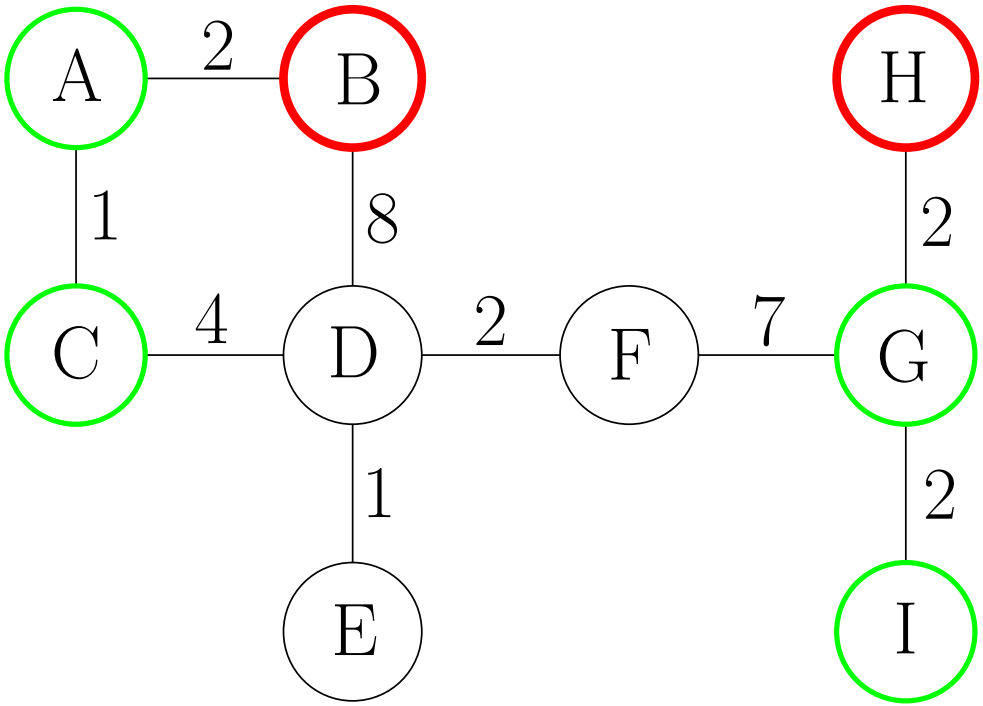
Erster Algorithmus: Beispiel



$V' = \{A, C, D, E, F\}$
$C = \{H, B\}$

while $V' \neq \emptyset$ **do**
  Select any vertex $v \in V'$ and add v to C ;
  Delete all vertices from V' that are at distance at most $2r$ from v ;
end

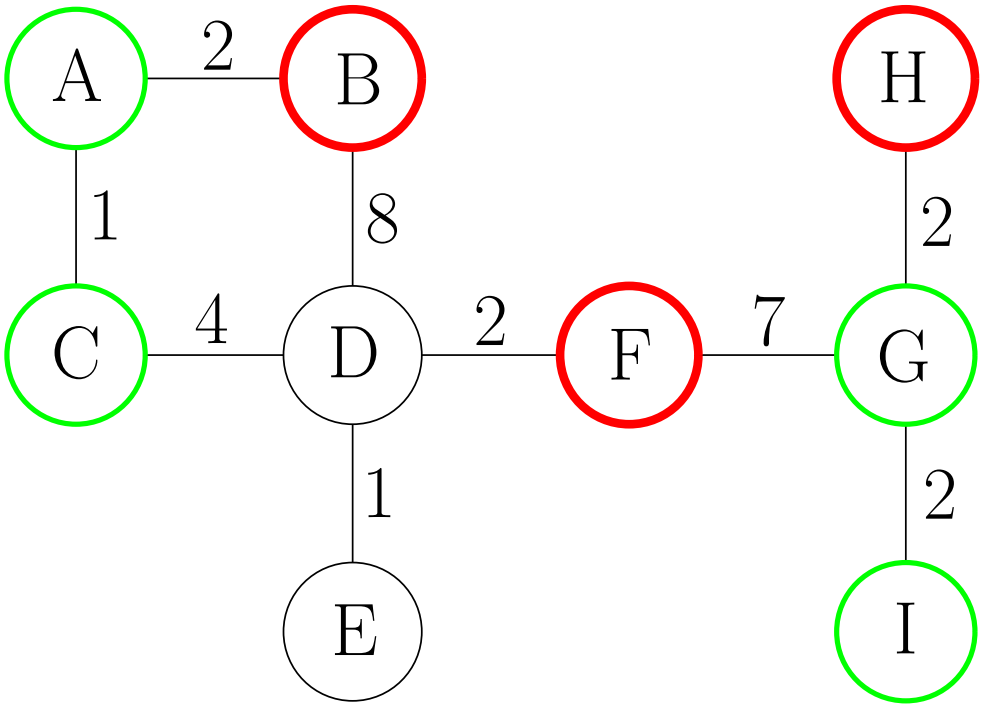
Erster Algorithmus: Beispiel



$V' = \{D, E, F\}$
$C = \{H, B\}$

while $V' \neq \emptyset$ **do**
  Select any vertex $v \in V'$ and add v to C ;
  Delete all vertices from V' that are at distance at most $2r$ from v ;
end

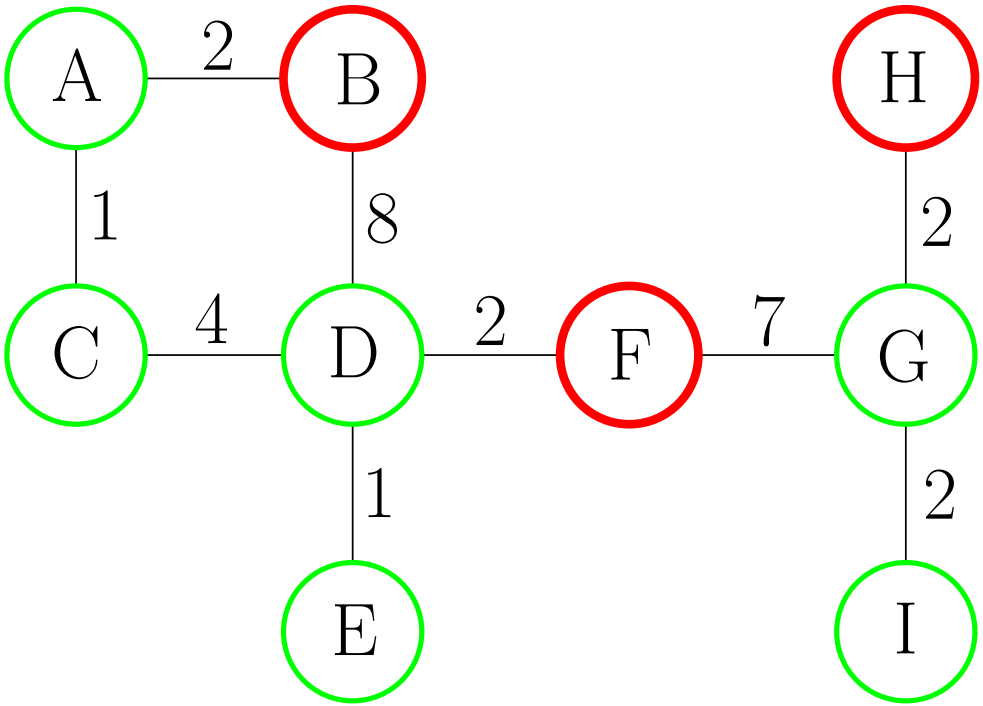
Erster Algorithmus: Beispiel



$V' = \{D, E\}$
$C = \{H, B, F\}$

while $V' \neq \emptyset$ **do**
  Select any vertex $v \in V'$ and add v to C ;
  Delete all vertices from V' that are at distance at most $2r$ from v ;
end

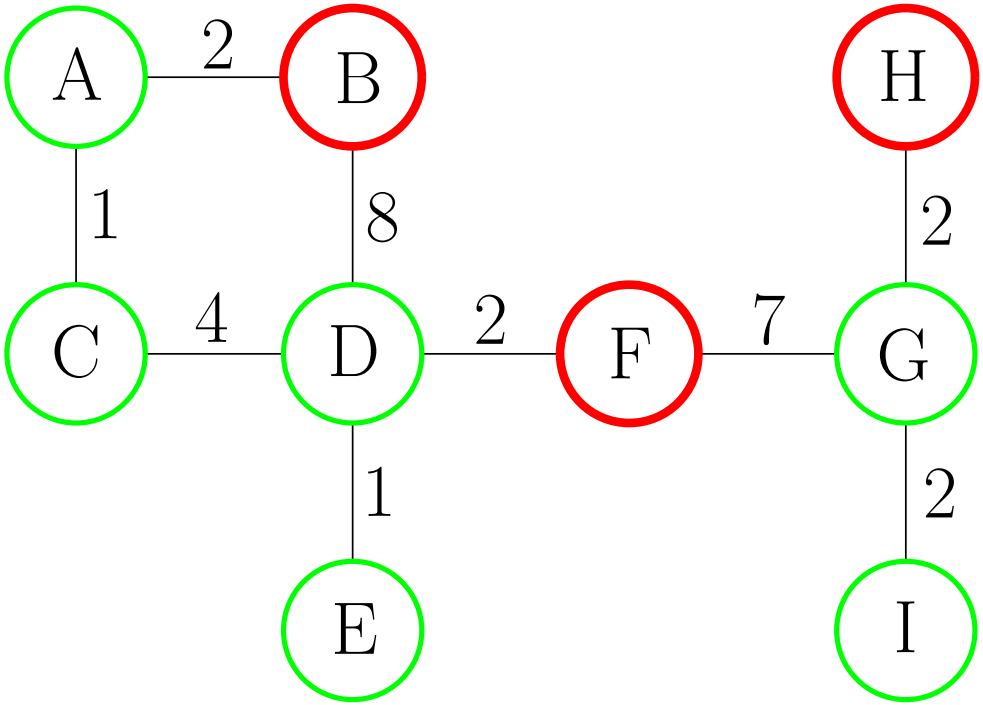
Erster Algorithmus: Beispiel



$$V' = \{\}$$
$$C = \{H, B, F\}$$

while $V' \neq \emptyset$ **do**
  Select any vertex $v \in V'$ and add v to C ;
  Delete all vertices from V' that are at distance at most $2r$ from v ;
end

Erster Algorithmus: Beispiel



$$V' = \{\}$$
$$C = \{H, B, F\}$$

if $|C| \leq k$ **then**
  Return C as the selected set of vertices;
else
  Claim (correctly) that there is no set of k centers with covering radius at most r;
end

Erster Algorithmus: Beweis der Korrektheit

Zu Beweisen

Falls der Algorithmus mehr als k Center wählt, so folgt, dass es keine optimale Lösung mit k Centern und Radius r gibt

Wir definieren:

- c in der Nähe von $c^* \Leftrightarrow dist(c, c^*) \leq r$
- Die Menge der Center die der Algorithmus liefert sei C
- Die Menge der Center der optimalen Lösung sei C^*

Erster Algorithmus: Beweis der Korrektheit

- Beweis durch Widerspruch:
- Annahme:
  - Algorithmus wählt mehr als k Center
  - Es existiert eine optimale Lösung mit maximal k Centern und einem Radius $r(C^*) \leq r$
- Zuerst muss gezeigt werden: Kein Center $c^* \in C^*$ kann in der Nähe von zwei Centern $c, c' \in C$ sein

Erster Algorithmus: Beweis der Korrektheit

Zu beweisen

Kein Center $c^* \in C^*$ kann in der Nähe von zwei Centern $c, c' \in C$ sein

Widerspruchsbeweis:

- Wir wissen: Distanz zwischen c und c' ist größer als $2r$, also $dist(c, c') > 2r$
- Annahme: $dist(c, c^*) \leq r$ und $dist(c^*, c') \leq r$
- Widerspruch zur Dreiecksungleichung:
$$2r < dist(c, c') \leq dist(c, c^*) + dist(c^*, c')$$

□

Erster Algorithmus: Beweis der Korrektheit

- Wir wissen nun: Kein Center $c^* \in C^*$ kann in der Nähe von zwei Centern $c, c' \in C$ sein.
- Wir wissen außerdem: jedes Center $c \in C$ hat mindestens ein Center $c^* \in C^*$ in der Nähe
- Daraus folgt:
  - $|C^*| \geq |C|$
  - Mit der Annahme $|C| > k$ gilt nun: $|C^*| > k$
  - $|C^*| > k$ ist ein Widerspruch zur Annahme, dass $|C^*| \leq k$

□

Erster Algorithmus: Problem mit dem Radius

- Wie das Problem lösen, wenn man den optimalen Radius nicht kennt?
- Lösung: Radius approximieren
- Binäre Suche mithilfe des Algorithmus
- Anfangswerte:
  $r_{min} = 0$ und $r_{max} = \max_{v,v' \in V} dist(v, v')$
- Ergebnis ist so sehr nah an einer 2-Approximation dran

Zweiter Algorithmus: Allgemeines

- relative Gütegarantie von 2
- benötigt **keinen** optimalen Radius
- baut auf Erkenntnissen von dem ersten Algorithmus auf

Zweiter Algorithmus: Grundsätzliche Idee

- Im ersten Algorithmus: zufälligen Knoten v zu C hinzufügen, für welchen gilt: $dist(v, C) > 2r$
- Einfach den Knoten v mit größter $dist(v, C)$ nehmen?
- Fallunterscheidung:
  - Fall 1: es existiert mindestens ein Knoten v' mit $dist(v', C) > 2r$
    $\Rightarrow$ Knoten v muss einer dieser Knoten sein
  - Fall 2: es existiert kein Knoten v' mit $dist(v', C) > 2r$
    $\Rightarrow$ Alle Knoten mit $2r$ abgedeckt

Zweiter Algorithmus: Pseudocode

Assume $k \leq |V|$ (else define $C = V$);
Select any vertex v and let $C = \{v\}$;
while $|C| < k$ **do**
    Select a vertex $v \in V$ that maximizes $dist(v, C)$;
    Add vertex v to C;
end
Return C as the selected set of vertices;

Zweiter Algorithmus: Pseudocode

Assume $k \leq |V|$ (else define $C = V$);
Select any vertex v and let $C = \{v\}$;
while $|C| < k$ **do**
    Select a vertex $v \in V$ that maximizes $dist(v, C)$;
    Add vertex v to C;
end
Return C as the selected set of vertices;

Zweiter Algorithmus: Pseudocode

Assume $k \leq |V|$ (else define $C = V$);
Select any vertex v and let $C = \{v\}$;
while $|C| < k$ **do**
    Select a vertex $v \in V$ that maximizes $dist(v, C)$;
    Add vertex v to C;
end
Return C as the selected set of vertices;

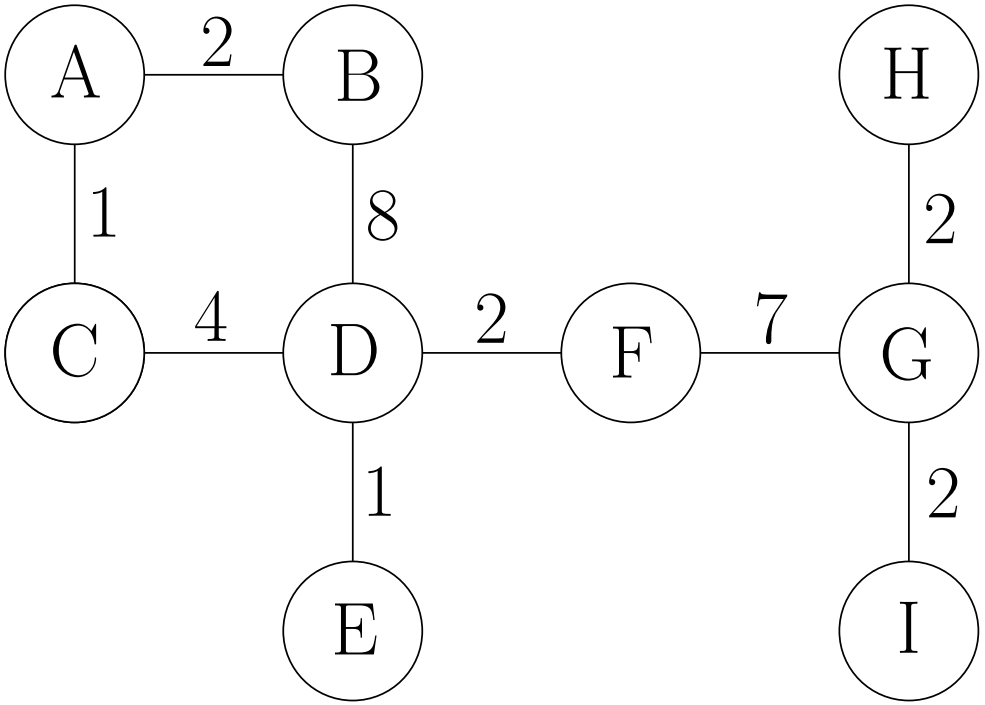
Zweiter Algorithmus: Pseudocode

Assume $k \leq |V|$ (else define $C = V$);
Select any vertex v and let $C = \{v\}$;
while $|C| < k$ **do**
  Select a vertex $v \in V$ that maximizes $dist(v, C)$;
  Add vertex v to C;
end
Return C as the selected set of vertices;

Zweiter Algorithmus: Pseudocode

Assume $k \leq |V|$ (else define $C = V$);
Select any vertex v and let $C = \{v\}$;
while $|C| < k$ **do**
  Select a vertex $v \in V$ that maximizes $dist(v, C)$;
  Add vertex v to C;
end
Return C as the selected set of vertices;

Zweiter Algorithmus: Beispiel

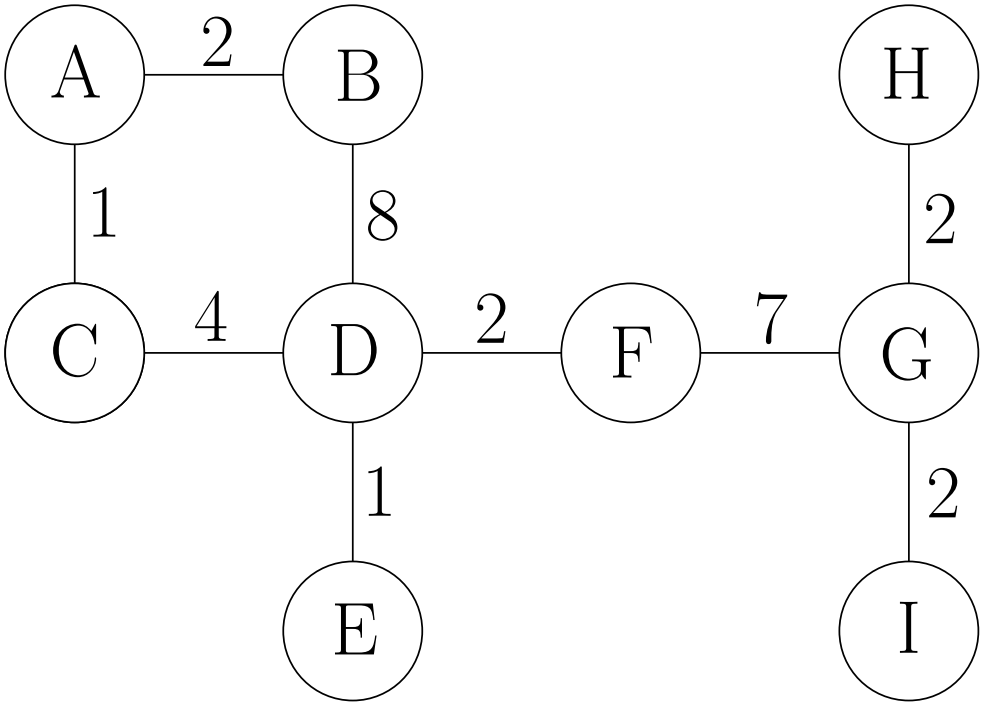


- Sei $G = (V, E)$ der links dargestellte Graph
- Sei $k = 4$

Assume $k \leq |V|$ (else define $C = V$);
Select any vertex v and let $C = \{v\}$;

. . .

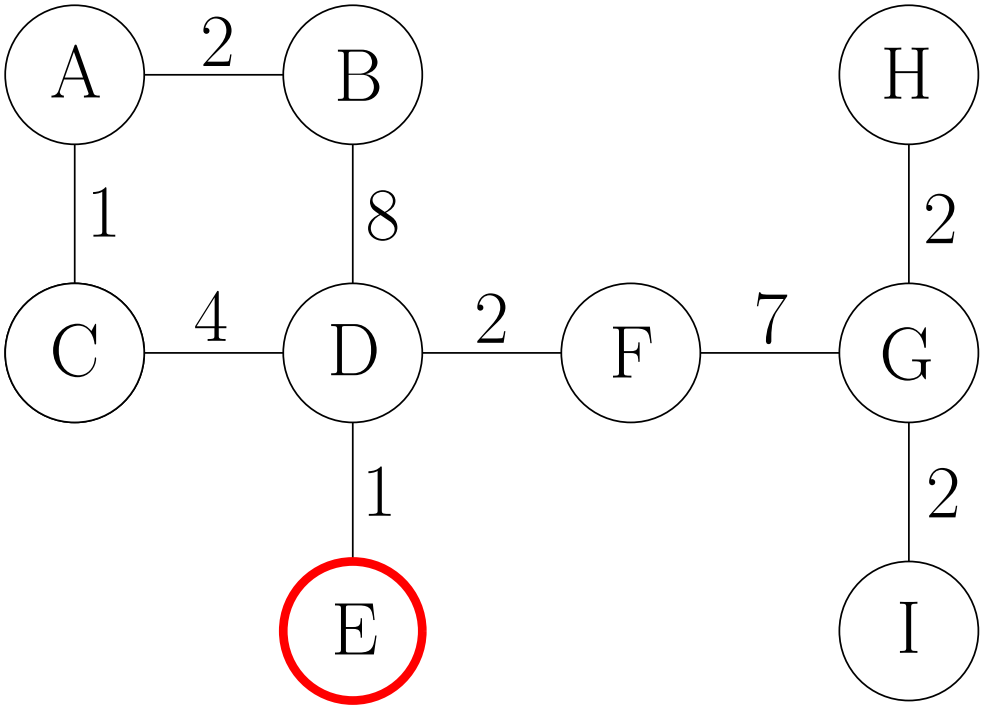
Zweiter Algorithmus: Beispiel



$C = \{\}$

Assume $k \leq |V|$ (else define $C = V$);
Select any vertex v and let $C = \{v\}$;
. . .

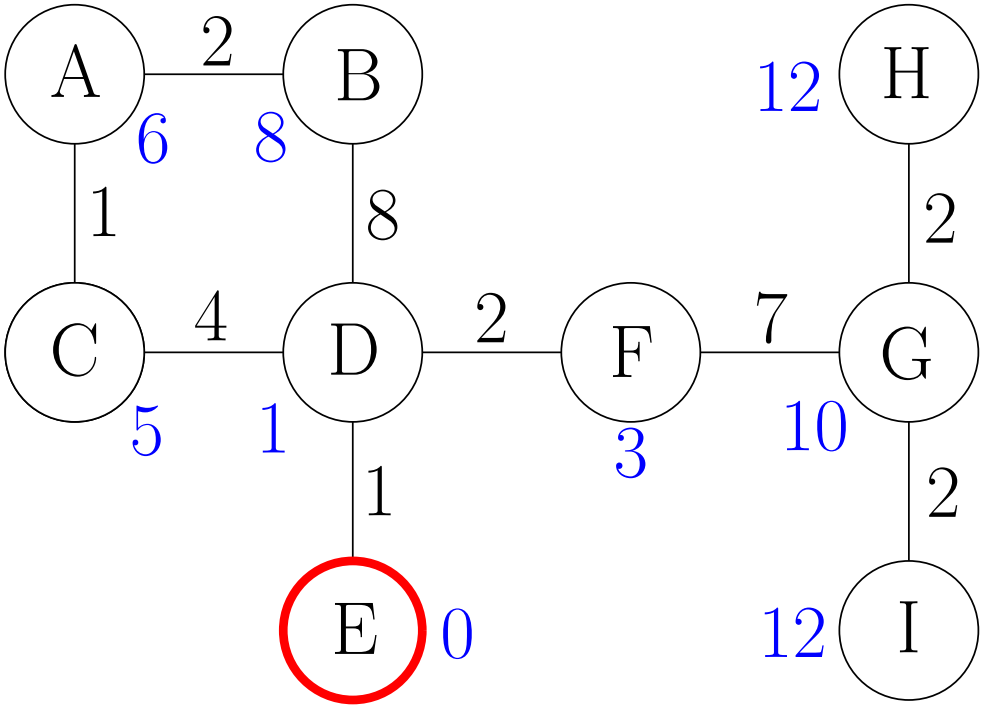
Zweiter Algorithmus: Beispiel



$C = \{E\}$

Assume $k \leq |V|$ (else define $C = V$);
Select any vertex v and let $C = \{v\}$;
. . .

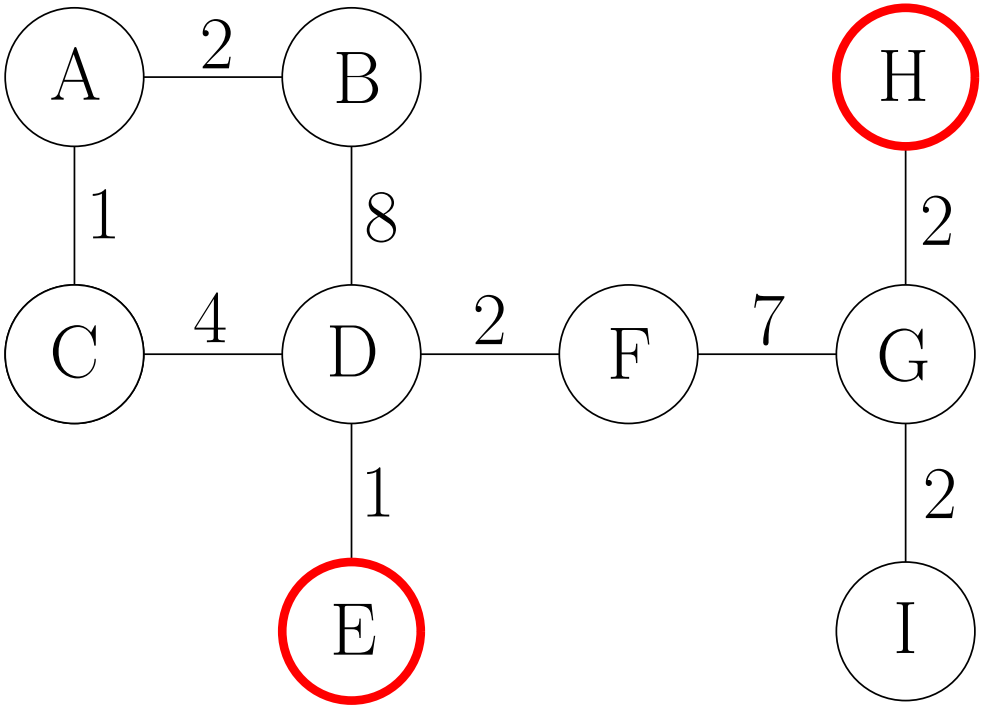
Zweiter Algorithmus: Beispiel



$C = \{E\}$

while $|C| < k$ **do**
  Select a vertex $v \in V$ that maximizes $dist(v, C)$;
  Add vertex v to C;
end

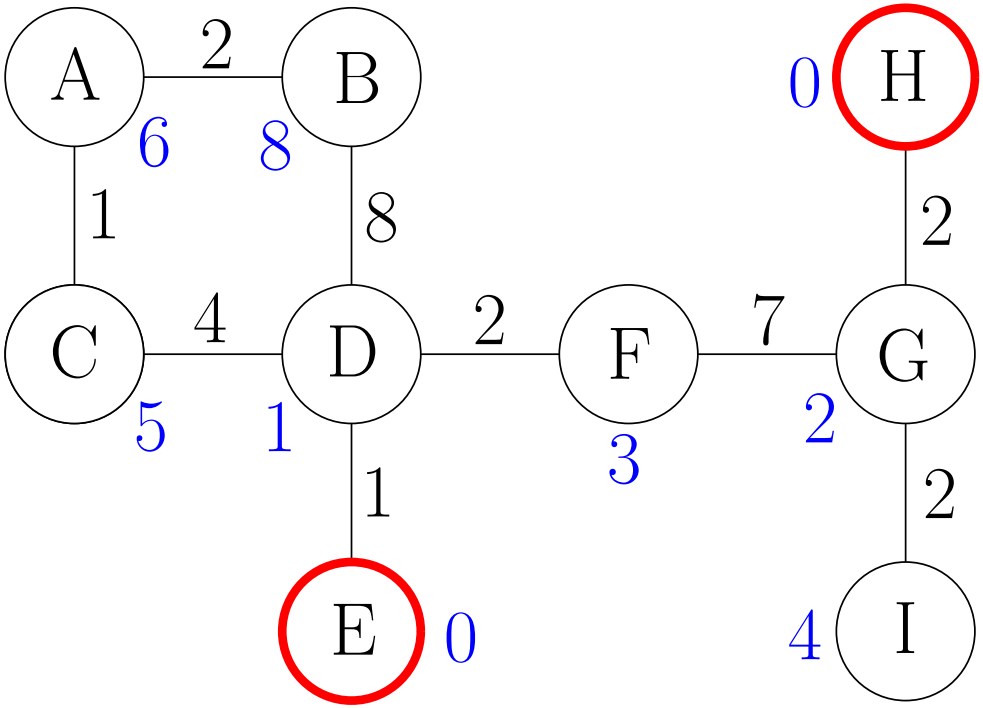
Zweiter Algorithmus: Beispiel



$C = \{E, H\}$

while $|C| < k$ **do**
  Select a vertex $v \in V$ that maximizes $dist(v, C)$;
  Add vertex v to C;
end

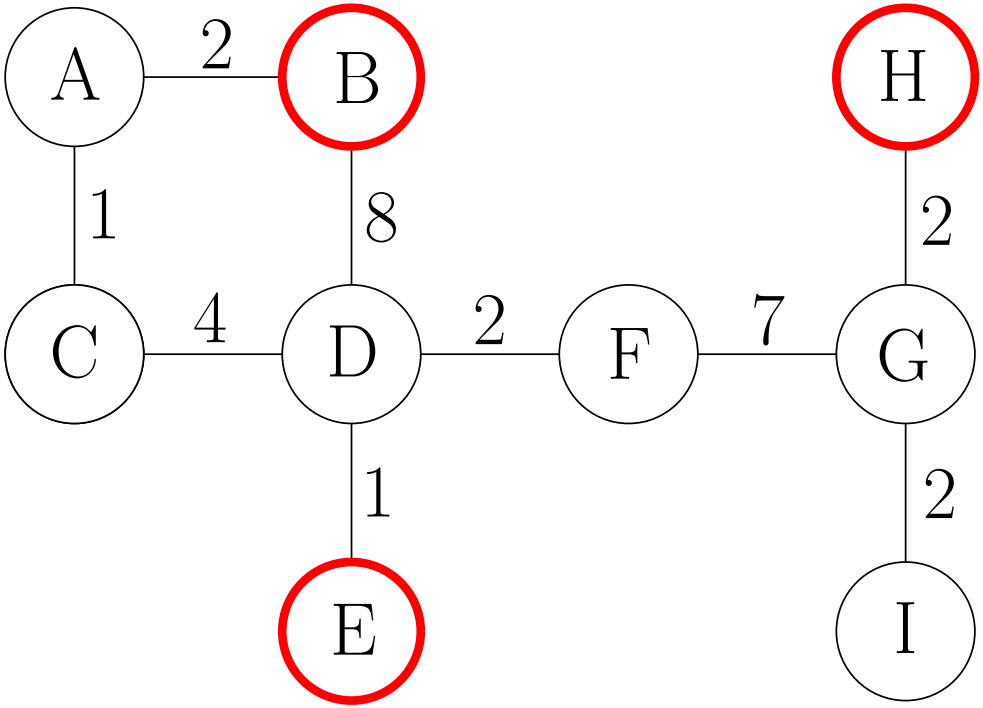
Zweiter Algorithmus: Beispiel



$C = \{E, H\}$

while $|C| < k$ **do**
  Select a vertex $v \in V$ that maximizes $dist(v, C)$;
  Add vertex v to C;
end

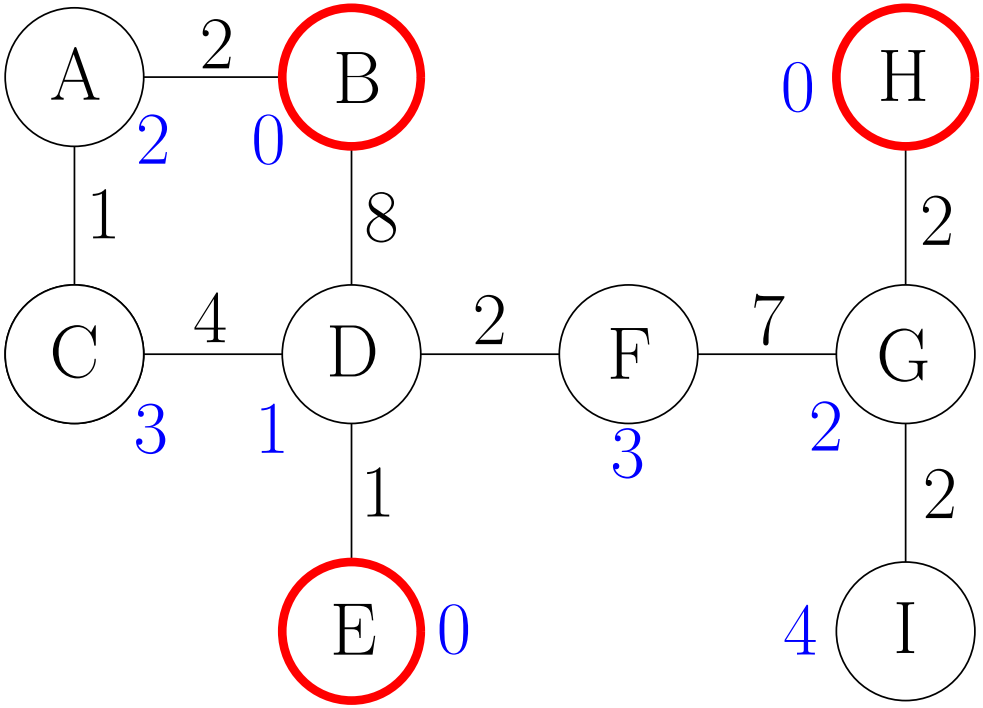
Zweiter Algorithmus: Beispiel



$C = \{E, H, B\}$

while $|C| < k$ **do**
  Select a vertex $v \in V$ that maximizes $dist(v, C)$;
  Add vertex v to C;
end

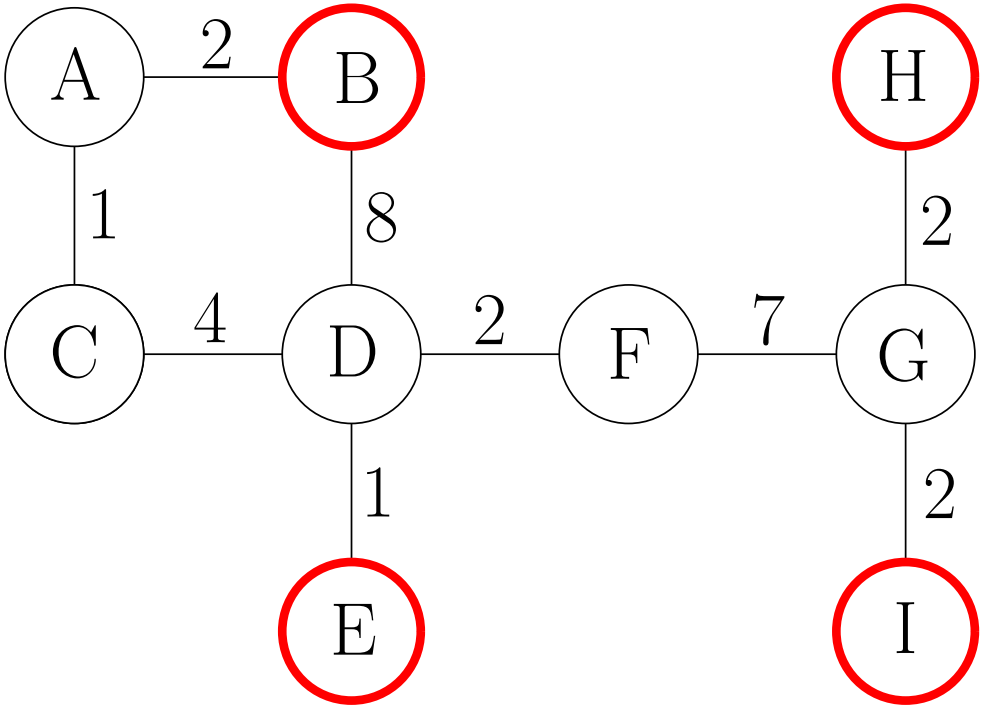
Zweiter Algorithmus: Beispiel



$C = \{E, H, B\}$

while $|C| < k$ **do**
  Select a vertex $v \in V$ that maximizes $dist(v, C)$;
  Add vertex v to C;
end

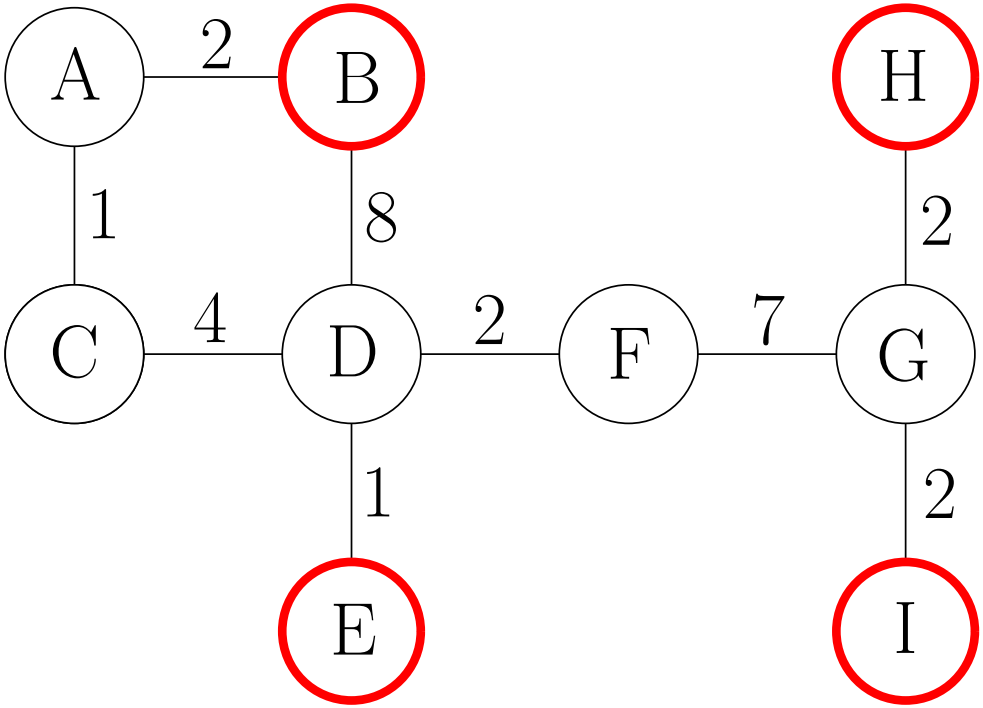
Zweiter Algorithmus: Beispiel



$C = \{E, H, B, I\}$

while $|C| < k$ **do**
  Select a vertex $v \in V$ that maximizes $dist(v, C)$;
  Add vertex v to C;
end

Zweiter Algorithmus: Beispiel

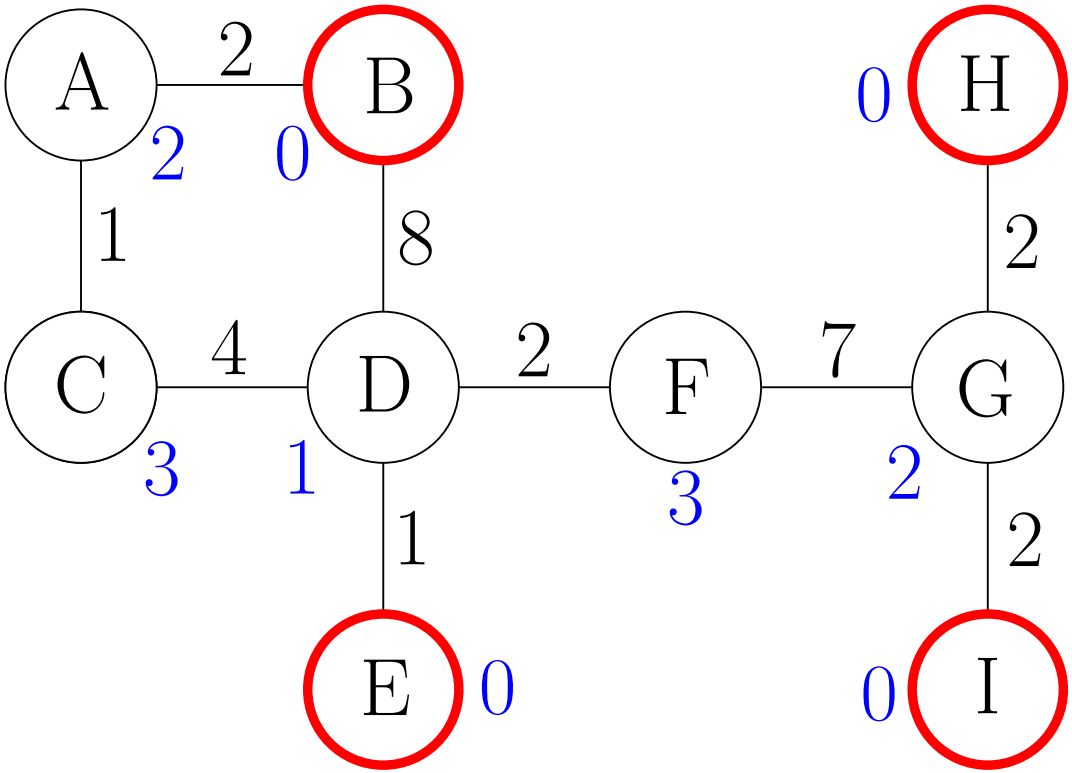


$C = \{E, H, B, I\}$

Return C as the selected set of vertices

Zweiter Algorithmus: Beispiel

Lösung des Algorithmus mit Radius $r^C(V) = 3$:



Zweiter Algorithmus: Beweis der Korrektheit

Wir definieren:

- Sei C die Menge der Center, die der Algorithmus liefert
- C^* ist die Menge der Center aus der optimalen Lösung
- Der optimale Radius $r(C^*)$ ist der minimale Radius für k Center

Zweiter Algorithmus: Beweis der Korrektheit

Zu beweisen

Der Radius der Menge C, die der Algorithmus ausgibt, ist $\leq 2r$

- Vorgehen:
  - Beweis durch Widerspruch
  - Annahme: Algorithmus liefert Menge C aus k Centern mit $r(C) > 2r$

Zweiter Algorithmus: Beweis der Korrektheit

Beweis durch Widerspruch:

Annahme

Algorithmus liefert Menge C aus k Centern mit $r(C) > 2r$

- Wir können zeigen: Dieser Algorithmus ist eine korrekte Implementierung der ersten k Iterationen der While-Schleife des ersten Algorithmus
- Erster Algorithmus würde mehr als k Knoten wählen
  $\Rightarrow$ Unsere definierte optimale Lösung kann so nicht existieren
  $\Rightarrow$ Widerspruch

□

Zweiter Algorithmus: Beweis der Korrektheit

Noch zu beweisen

Der zweite Algorithmus ist eine korrekte Implementierung der ersten k Iterationen der While-Schleife des ersten Algorithmus

Äquivalent

Der zweite Algorithmus wählt in den ersten k Iterationen nur Knoten mit einer Distanz $> 2r$ aus

Zweiter Algorithmus: Beweis der Korrektheit

Zu beweisen

Der zweite Algorithmus wählt in den ersten k Iterationen nur Knoten mit einer Distanz $> 2r$ aus

Unsere Annahme war

Der Algorithmus liefert Menge C aus k Centern mit $r(C) > 2r$

$\Rightarrow$ es existiert ein Knoten v mit $dist(v, C) > 2r$

- Annahme:
  - Wir sind in einer beliebigen Iteration < k
  - die Menge der Center ist hier C'
  - $c' \in V$ wird als nächstes zu C' hinzugefügt
- Es folgt: $dist(c', C') \geq dist(v, C') \geq dist(v, C) > 2r$
  $\Rightarrow dist(c', C') > 2r$ für alle c' □

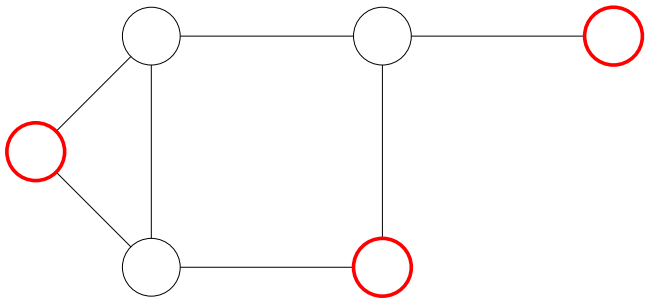
Minimale relative Gütegarantie

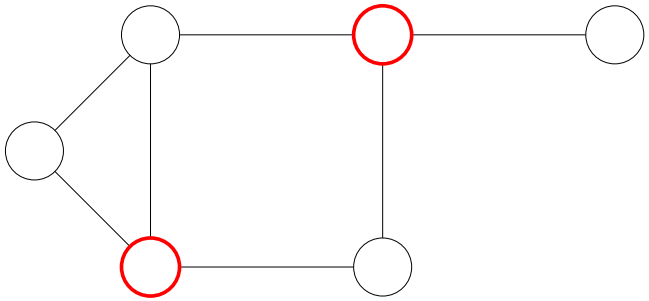
- Existiert ein Algorithmus mit einer relativen Gütegarantie unter 2?
- Nein, falls NP $\neq$ P.
- Vorgehen: Reduktion von NP-vollständigem Problem "Dominating Set" auf das Center Selection Problem

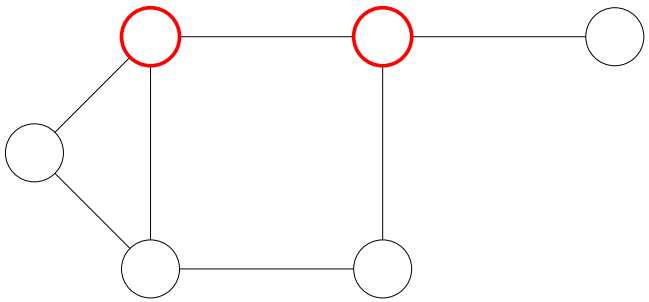
Dominating Set

- NP-vollständiges Problem
- Entscheidungsproblem:
  - Gegeben: ungerichteter, ungewichteter Graph $G = (V, E)$ und ein $k \in \mathbb{N}$
  - Gesucht: Existiert eine Menge $D \subseteq V$ mit $|D| \leq k$, sodass jeder Knoten $v \in V \setminus D$ adjazent zu einem Knoten aus D ist

Dominating Set: Beispiel

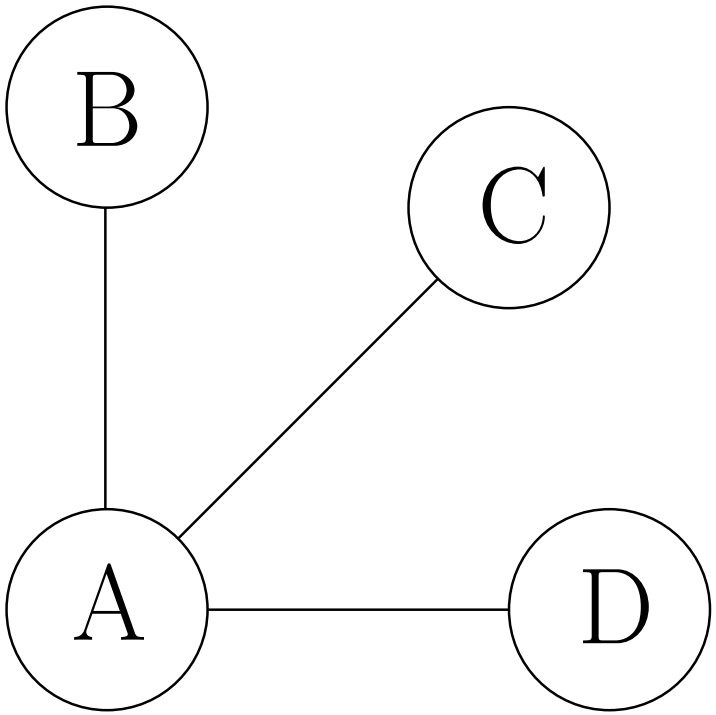






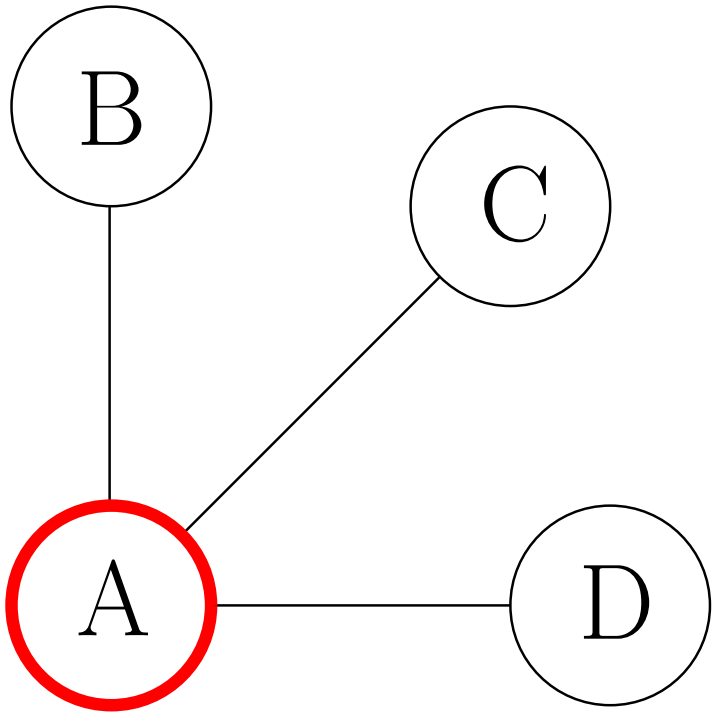
- Dominating Sets mit
  - $|D| = 3$
  - $|D| = 2$
  - $|D| = 2$

Dominating Set: Beispiel



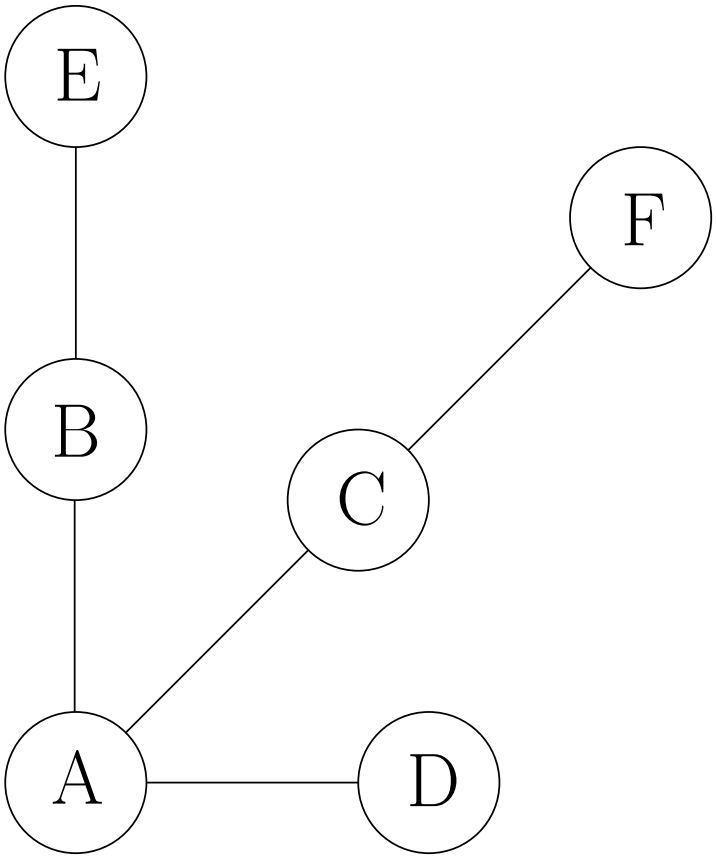
- Graph $G = (V, E)$ ist der links dargestellte Graph
- $k = 1$
- existiert ein Dominating Set mit $|D| \leq k$?

Dominating Set: Beispiel



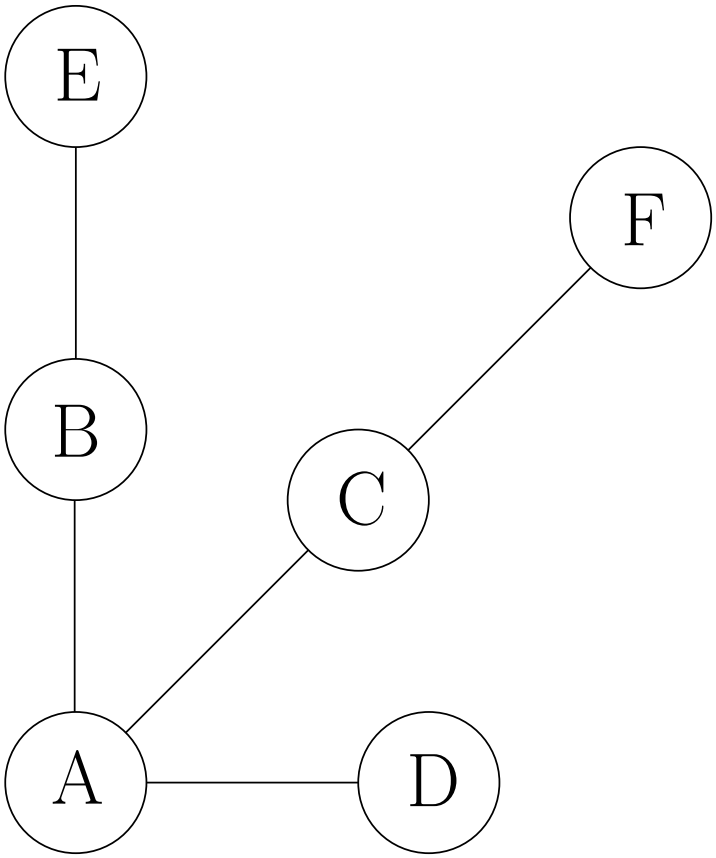
- Ja, ein solches Dominating Set existiert
- $D = \{A\}$

Dominating Set: Beispiel



- Graph $G = (V, E)$ ist der links dargestellte Graph
- $k = 2$
- existiert ein Dominating Set mit $|D| \leq k$?

Dominating Set: Beispiel



- Nein, ein solches Dominating Set existiert nicht

Beweis

Zu beweisen

Falls NP $\neq$ P gilt, so kann kein Approximationsalgorithmus mit relativer Gütergarantie kleiner 2 existieren

- Beweis durch Widerspruch
- Annahme: NP $\neq$ P und es existiert ein Approximationsalgorithmus mit relativer Gütegarantie kleiner 2

Beweis

- Reduktion von Dominating Set auf Center Selection Problem
  - Sei $G = (V, E)$ ein Graph, für welchen bestimmt werden soll, ob ein Dominating Set mit $|D| \leq k$ existiert
  - Wir definieren $G' = (V', E')$ wie folgt:
    - $V' = V$
    - $E' = E$
    - Kantengewicht an allen Kanten ist 1

Beweis

Zu zeigen

Algorithmus löst Dominating Set

2 Fälle:

- Fall 1: In G ist ein Dominating Set mit $|D| \leq k$
  - In G' ist eine Menge C von Centern mit $|C| = k$ und $r = 1$
  - Der Algorithmus liefert k Center mit $r < 2$
- Fall 2: In G ist kein Dominating Set mit $|D| \leq k$
  - Die optimale Lösung des Center Selection Problem auf G' hat einen Radius $r \geq 2$
  - Der Algorithmus findet eine Menge C mit $r \geq 2$
- Folge: unser Algorithmus entscheidet das Dominating Set Problem in Polynomialzeit

Beweis

- Der Algorithmus entscheidet das Dominating Set Problem in Polynomialzeit
- So muss gelten: NP = P
- NP = P ist ein Widerspruch zur Annahme
- Bewiesen: Es kann kein Approximationsalgorithmus mit einer Gütegarantie < 2 existieren, falls NP $\neq$ P □

Vielen Dank für eure Aufmerksamkeit!