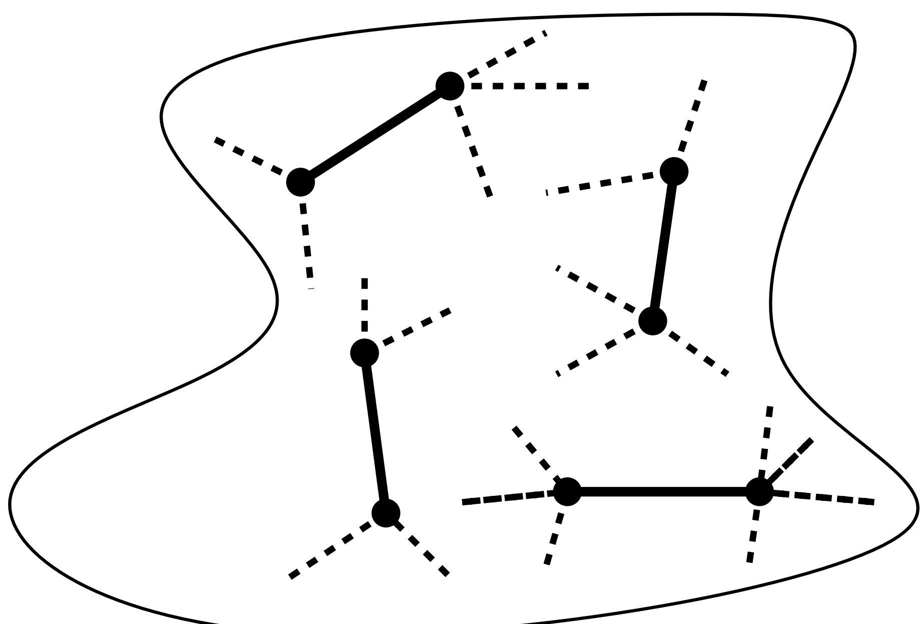
Introduction to Approximation Algorithms, part I

20-12 2022, Mikkel Abrahamsen,
Department of Computer Science

APPROX-VERTEX-COVER(G)
    $C := \emptyset$
    while $E(G) \neq \emptyset$
        choose $uv \in E(G)$
        $C := C \cup \{u, v\}$
        remove all edges incident on u or v from $E(G)$
    return C



The big picture

Last time: Fast exponential algorithms (good for small instances) and parameterized algorithms (good for special cases).

The big picture

Last time: Fast exponential algorithms (good for small instances) and parameterized algorithms (good for special cases).

Today: Approximation algorithms (good when suboptimal solutions are acceptable).

Definition

Def.: An algorithm for an optimization problem has *approximation ratio* $\rho(n)$ if for every input of size n,

$$\max\left\{\frac{C}{C^*}, \frac{C^*}{C}\right\} \leq \rho(n).$$

Definition

Def.: An algorithm for an optimization problem has *approximation ratio* $\rho(n)$ if for every input of size n,

$$\max\left\{\frac{C}{C^*}, \frac{C^*}{C}\right\} \le \rho(n).$$

C^* :=cost(opt. sol.)

C :=cost(produced sol.)

Definition

Def.: An algorithm for an optimization problem has *approximation ratio* $\rho(n)$ if for every input of size n,

$$\max\left\{\boxed{\frac{C}{C^*}}, \boxed{\frac{C^*}{C}}\right\} \leq \rho(n).$$

C^* :=cost(opt. sol.)

C :=cost(produced sol.)

minimization problem

maximization problem

Vertex Cover

Def.: Let $G = (V, E)$ be a graph. A set $V' \subseteq V$ of vertices is a *vertex cover* if for all $uv \in E$, we have $u \in V'$ or $v \in V'$.

Vertex Cover

Def.: Let $G = (V, E)$ be a graph. A set $V' \subseteq V$ of vertices is a *vertex cover* if for all $uv \in E$, we have $u \in V'$ or $v \in V'$.

NP-hard!

Vertex Cover

Def.: Let $G = (V, E)$ be a graph. A set $V' \subseteq V$ of vertices is a *vertex cover* if for all $uv \in E$, we have $u \in V'$ or $v \in V'$.

NP-hard!

```
APPROX-VERTEX-COVER(G)
    C := ∅
    while E(G) ≠ ∅
        choose uv ∈ E(G)
        C := C ∪ {u, v}
        remove all edges incident on u or v from E(G)
    return C
```

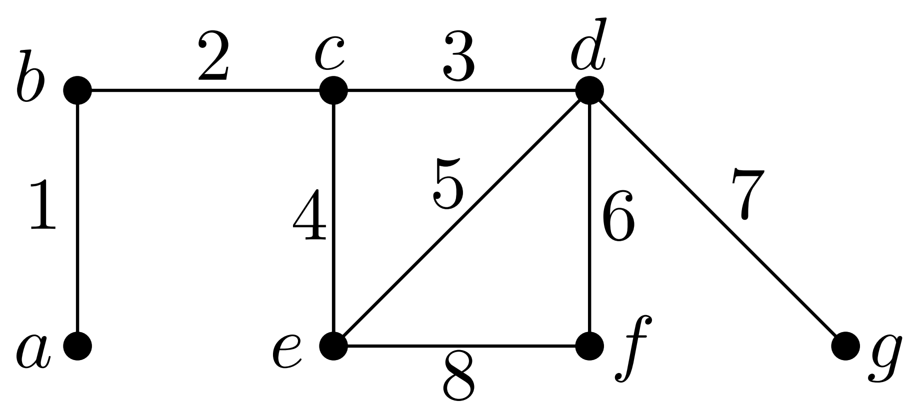
Vertex Cover

Def.: Let $G = (V, E)$ be a graph. A set $V' \subseteq V$ of vertices is a *vertex cover* if for all $uv \in E$, we have $u \in V'$ or $v \in V'$.

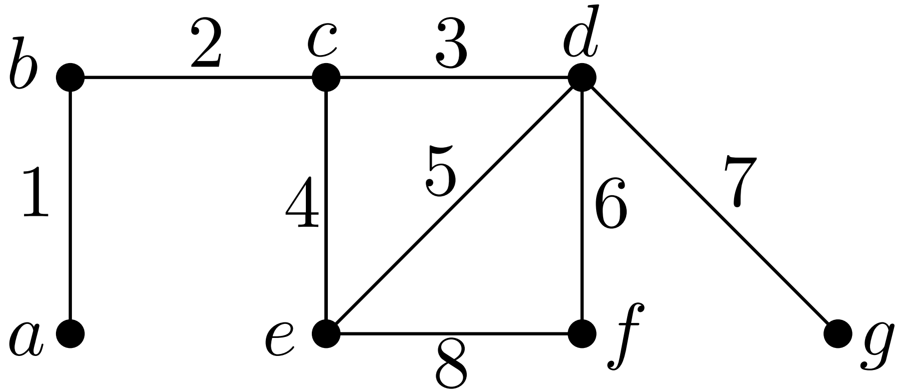
NP-hard!

```
APPROX-VERTEX-COVER(G)
  C := ∅
  while E(G) ≠ ∅
    choose uv ∈ E(G)
    C := C ∪ {u, v}
    remove all edges incident on u or v from E(G)
  return C
```

Exercise:



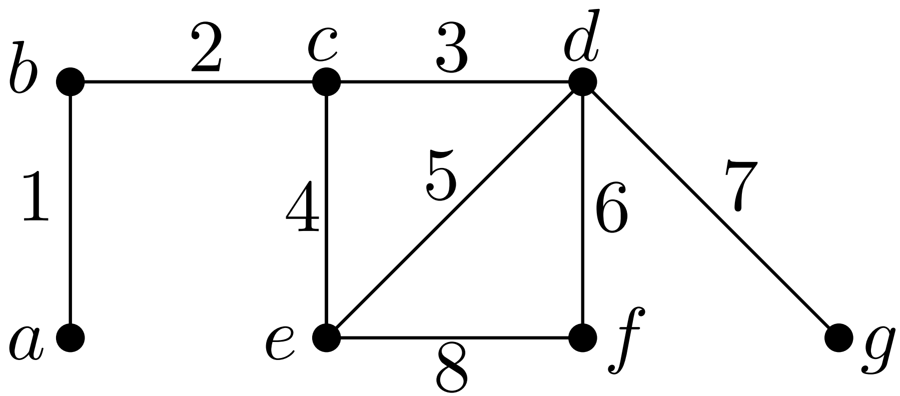
Implementation



Adjacency lists:

$L[a] = \{b\}$
$L[b] = \{a, c\}$
$L[c] = \{b, d, e\}$
$\vdots$

Implementation



Adjacency lists:

$L[a] = \{b\}$ $\quad E[a] = [1]$

$L[b] = \{a, c\}$ $\quad E[b] = [1, 2]$

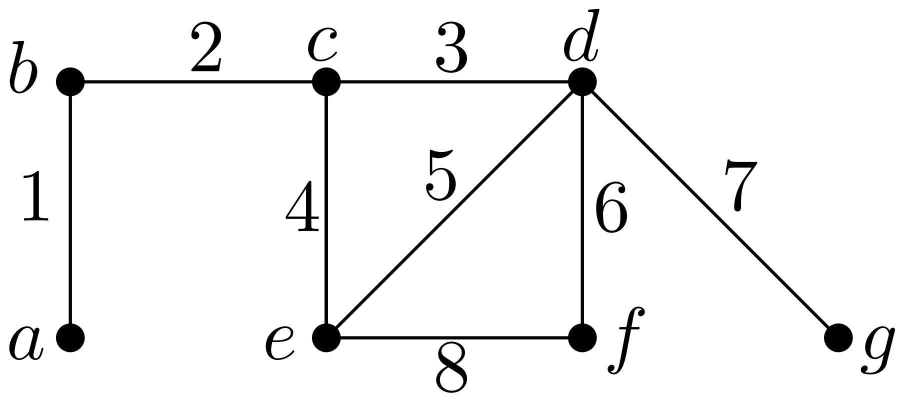
$L[c] = \{b, d, e\}$ $\quad E[c] = [2, 3, 4]$

$\vdots \qquad \vdots$

Array of edges

$$[(a, b, 1),\ (b, c, 1),\ (c, d, 1),\ (c, e, 1), (d, e, 1),\ \ldots]$$

Implementation



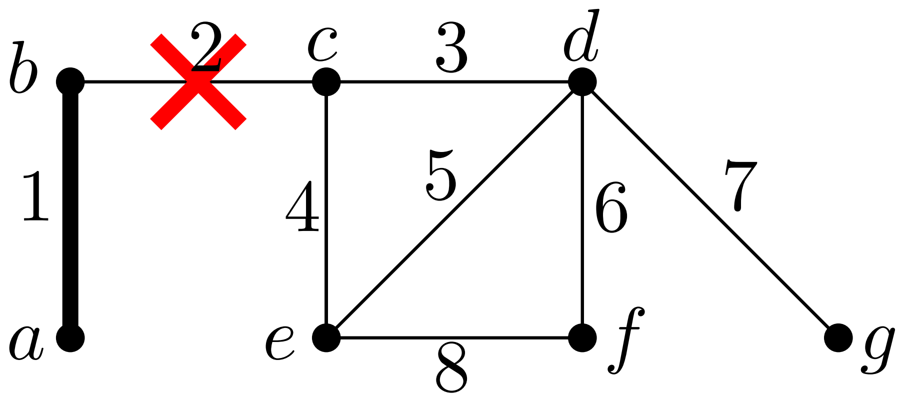
Adjacency lists:

$$L[a] = \{b\} \qquad E[a] = [1]$$
$$L[b] = \{a, c\} \qquad E[b] = [1, 2]$$
$$L[c] = \{b, d, e\} \qquad E[c] = [2, 3, 4]$$
$$\vdots \qquad \vdots$$

Array of edges

$$[\underline{(a, b, 1)},\ (b, c, 1),\ (c, d, 1),\ (c, e, 1), (d, e, 1),\ \ldots]$$

Implementation



Adjacency lists:

$L[a] = \{b\}$ $\quad E[a] = [1]$

$L[b] = \{a, c\}$ $\quad E[b] = [1, 2]$

$L[c] = \{b, d, e\}$ $\quad E[c] = [2, 3, 4]$

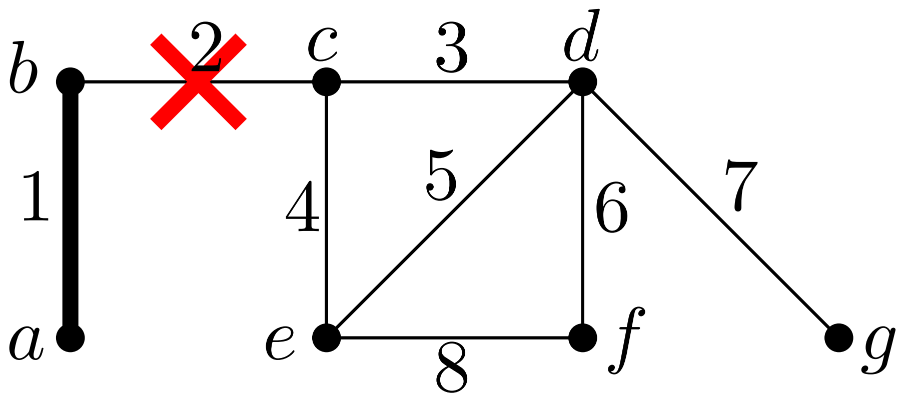
$\vdots$ $\quad \vdots$

Array of edges

$$[(a, b, 1),\ (b, c, 1),\ (c, d, 1),\ (c, e, 1), (d, e, 1),\ \ldots]$$

$$[(a, b, 0),\ (b, c, 0),\ (c, d, 1),\ (c, e, 1), (d, e, 1),\ \ldots]$$

Implementation



Adjacency lists:

$L[a] = \{b\}$ $E[a] = [1]$

$L[b] = \{a, c\}$ $E[b] = [1, 2]$

$L[c] = \{b, d, e\}$ $E[c] = [2, 3, 4]$

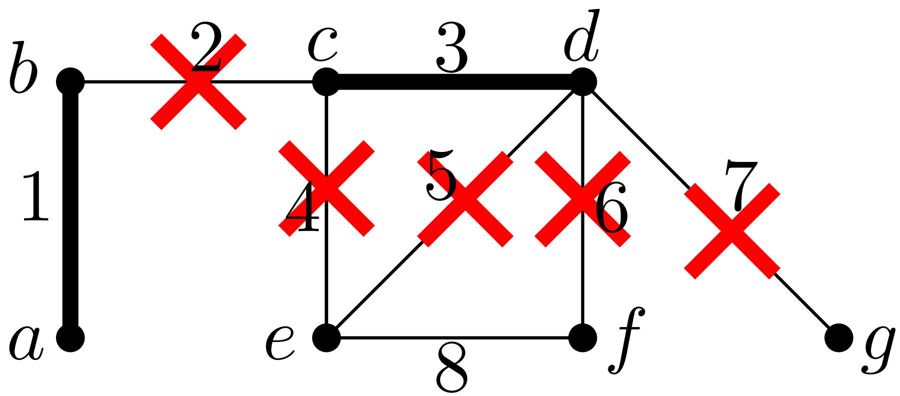
$\vdots$ $\vdots$

Array of edges

$$[(a, b, 1),\ (b, c, 1),\ (c, d, 1),\ (c, e, 1),\ (d, e, 1),\ \ldots]$$

$$[(a, b, 0),\ (b, c, 0),\ \underline{(c, d, 1)},\ (c, e, 1),\ (d, e, 1),\ \ldots]$$

Implementation



Adjacency lists:

$L[a] = \{b\}$ $\quad E[a] = [1]$

$L[b] = \{a, c\}$ $\quad E[b] = [1, 2]$

$L[c] = \{b, d, e\}$ $\quad E[c] = [2, 3, 4]$

$\vdots \qquad \vdots$

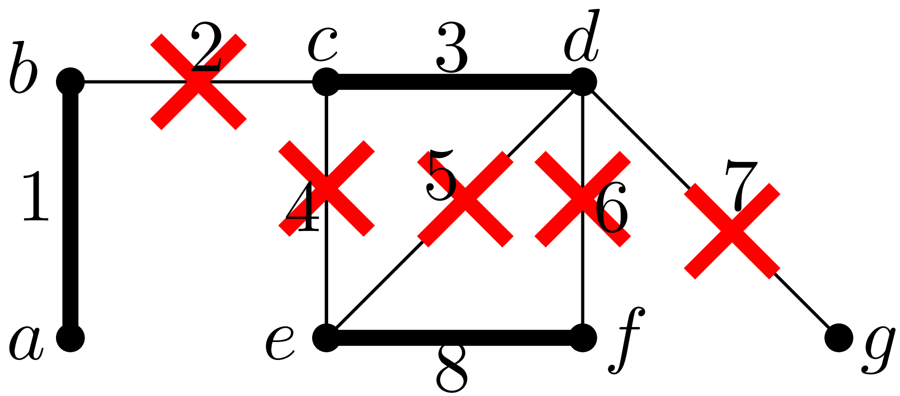
Array of edges

$$[(a, b, 1),\ (b, c, 1),\ (c, d, 1),\ (c, e, 1),\ (d, e, 1),\ \ldots]$$

$$[(a, b, 0),\ (b, c, 0),\ (c, d, 1),\ (c, e, 1),\ (d, e, 1),\ \ldots]$$

$$[(a, b, 0),\ (b, c, 0),\ (c, d, 0),\ (c, e, 0),\ (d, e, 0),\ \ldots]$$

Implementation



Adjacency lists:

$L[a] = \{b\}$ $\quad E[a] = [1]$

$L[b] = \{a, c\}$ $\quad E[b] = [1, 2]$

$L[c] = \{b, d, e\}$ $\quad E[c] = [2, 3, 4]$

$\vdots \qquad \vdots$

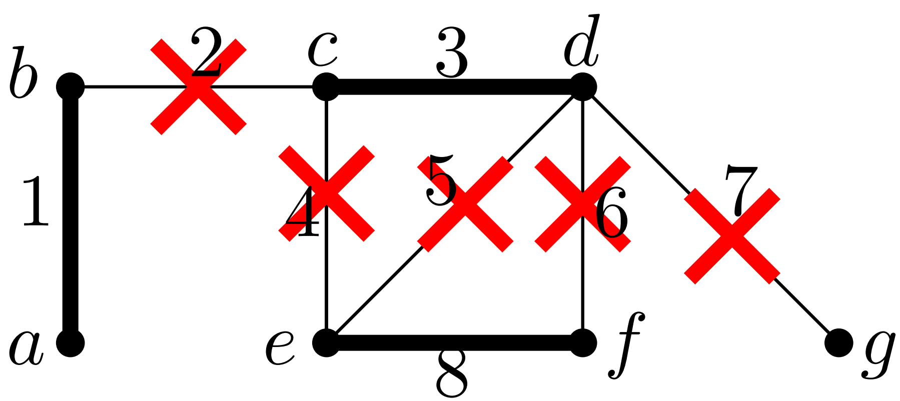
Array of edges

$$[(a, b, 1),\ (b, c, 1),\ (c, d, 1),\ (c, e, 1),\ (d, e, 1),\ \ldots]$$

$$[(a, b, 0),\ (b, c, 0),\ (c, d, 1),\ (c, e, 1),\ (d, e, 1),\ \ldots]$$

$$[(a, b, 0),\ (b, c, 0),\ (c, d, 0),\ (c, e, 0),\ (d, e, 0),\ \ldots]$$

Implementation



Adjacency lists:

$$L[a] = \{b\} \qquad E[a] = [1]$$
$$L[b] = \{a, c\} \qquad E[b] = [1, 2]$$
$$L[c] = \{b, d, e\} \qquad E[c] = [2, 3, 4]$$
$$\vdots \qquad \vdots$$

Array of edges

$$[(a, b, 1),\ (b, c, 1),\ (c, d, 1),\ (c, e, 1),\ (d, e, 1),\ \ldots]$$
$$[(a, b, 0),\ (b, c, 0),\ (c, d, 1),\ (c, e, 1),\ (d, e, 1),\ \ldots]$$
$$[(a, b, 0),\ (b, c, 0),\ (c, d, 0),\ (c, e, 0),\ (d, e, 0),\ \ldots]$$

Running time: $O(|V| + |E|)$

Theorem

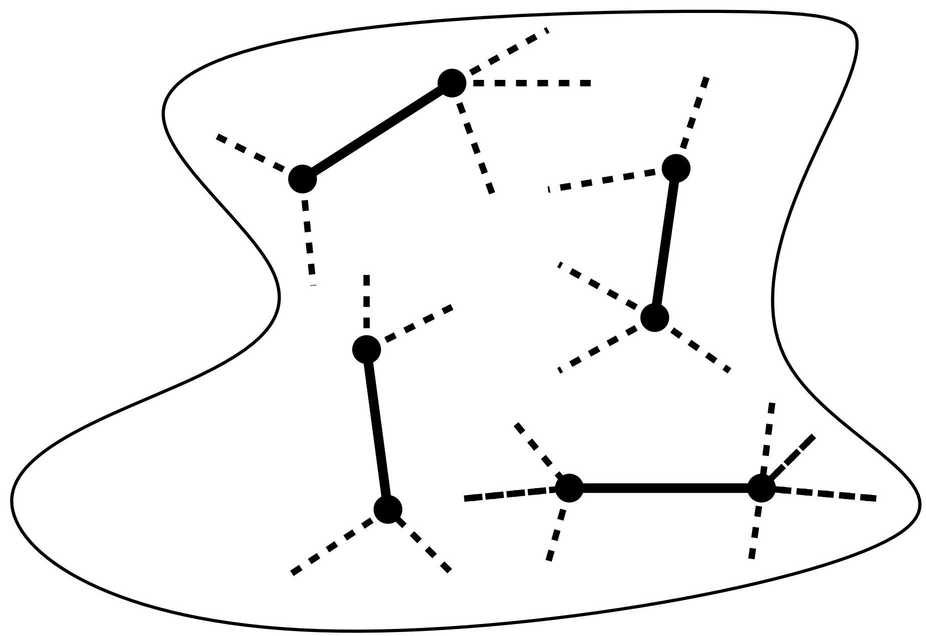
Thm.: APPROX-VERTEX-COVER is a 2-approximation algorithm.

Theorem

Thm.: APPROX-VERTEX-COVER is a 2-approximation algorithm.

Proof: Let C^* be an optimal cover.

Let $A \subset E$ be the edges chosen by the algorithm.



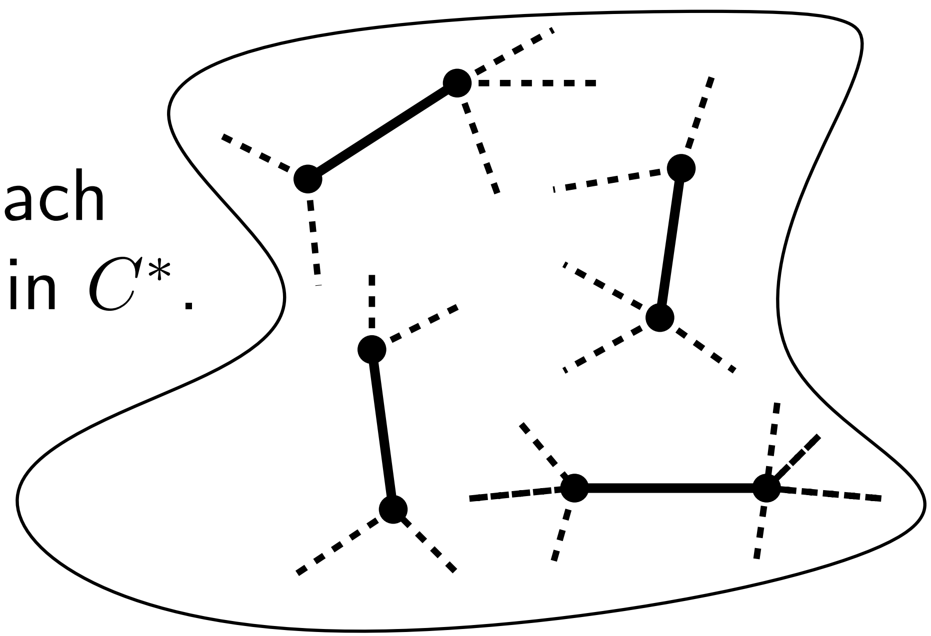
Theorem

Thm.: APPROX-VERTEX-COVER is a 2-approximation algorithm.

Proof: Let C^* be an optimal cover.

Let $A \subset E$ be the edges chosen by the algorithm.

An endpoint of each $uv \in A$ must be in C^*.

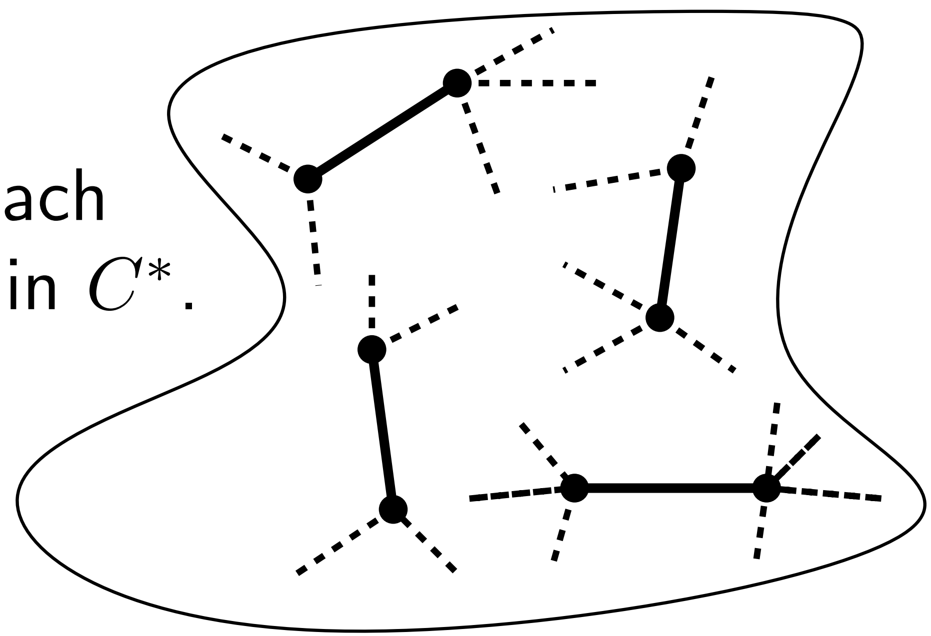


Theorem

Thm.: APPROX-VERTEX-COVER is a 2-approximation algorithm.

Proof: Let C^* be an optimal cover.
Let $A \subset E$ be the edges chosen by the algorithm.

An endpoint of each $uv \in A$ must be in C^*.



Hence,

$$|C^*| \geq |A| = |C|/2 \Longrightarrow \frac{|C|}{|C^*|} \leq 2.$$

Reflection and methodology

How can we prove $C/C^* \leq 2$ when we don't know C^*?

Answer: By proving $C \leq 2|A|$ and $|A| \leq C^*$.

Reflection and methodology

How can we prove $C/C^* \leq 2$ when we don't know C^*?

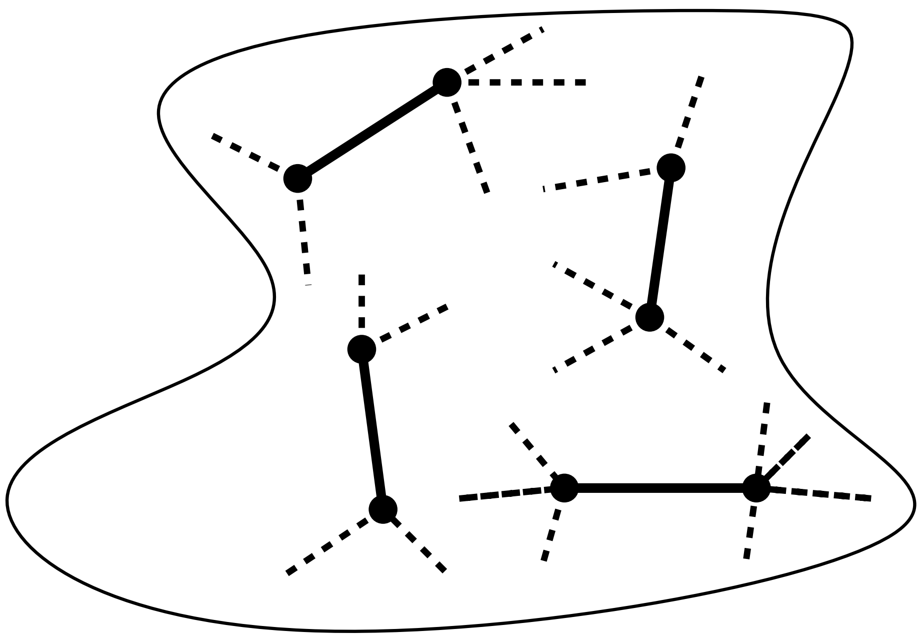
Answer: By proving $C \leq 2|A|$ and $|A| \leq C^*$.

General technique: Find a parameter $\square$ such that $C \leq \rho \cdot \square$ and $\square \leq C^*$.

For vertex cover: $\square = |A|$ and $\rho = 2$.

Question

Try to guess: Is there an approximation algorithm with a better approximation ratio?



History

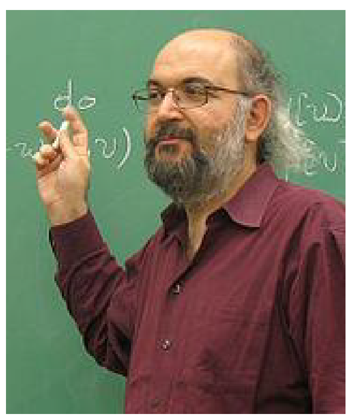
1972: Karp's 21 NP-complete problems
(including vertex cover, set cover, Hamiltonian cycle and subset sum)

Karp

Turing Award

History

19xx: Many $\leq 2 - o(1)$.



...

Gavril Yannakakis

History

Assuming P≠NP:
1999: Håstad, $\geq 7/6$
2005: Dinur & Safra, ≥ 1.38
2018: Khot, Minzer, Safra, ≥ 1.41


Håstad


Dinur


Safra


Khot


Minzer

History

2008: Khot & Regev, $\geq 2 - \varepsilon$ assuming the
Unique Game Conjecture.

Some, but not all people believe it.

Khot

Nevanlinna prize 2016

Regev

Traveling Salesperson

Given a complete undirected graph $G = (V, E)$.

For all $u, v \in V$, we are given $c(uv) \in \{0, 1, \ldots\}$.

Goal: Find minimum weight cycle through all vertices.

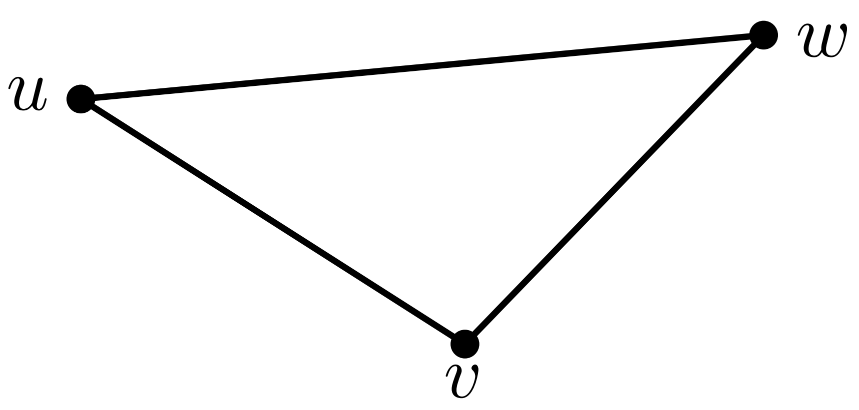
Traveling Salesperson

Given a complete undirected graph $G = (V, E)$.

For all $u, v \in V$, we are given $c(uv) \in \{0, 1, \ldots\}$.

Goal: Find minimum weight cycle through all vertices.

Assume: Triangle inequality: $c(uw) \leq c(uv) + c(vw)$.



Traveling Salesperson

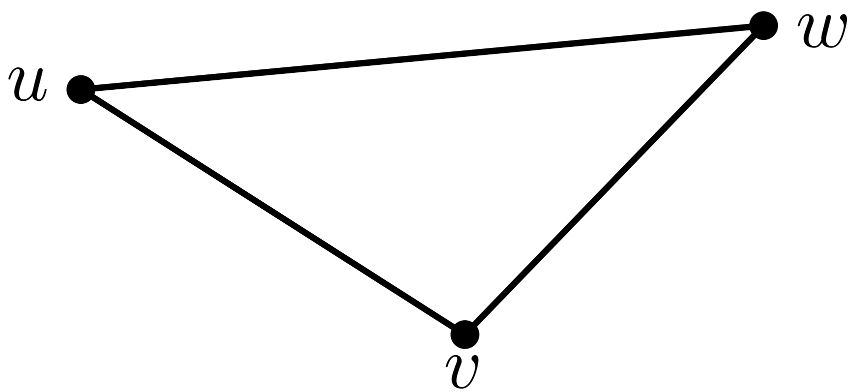
Given a complete undirected graph $G = (V, E)$.

For all $u, v \in V$, we are given $c(uv) \in \{0, 1, \ldots\}$.

Goal: Find minimum weight cycle through all vertices.

Assume: Triangle inequality: $c(uw) \leq c(uv) + c(vw)$.

Still NP-hard!

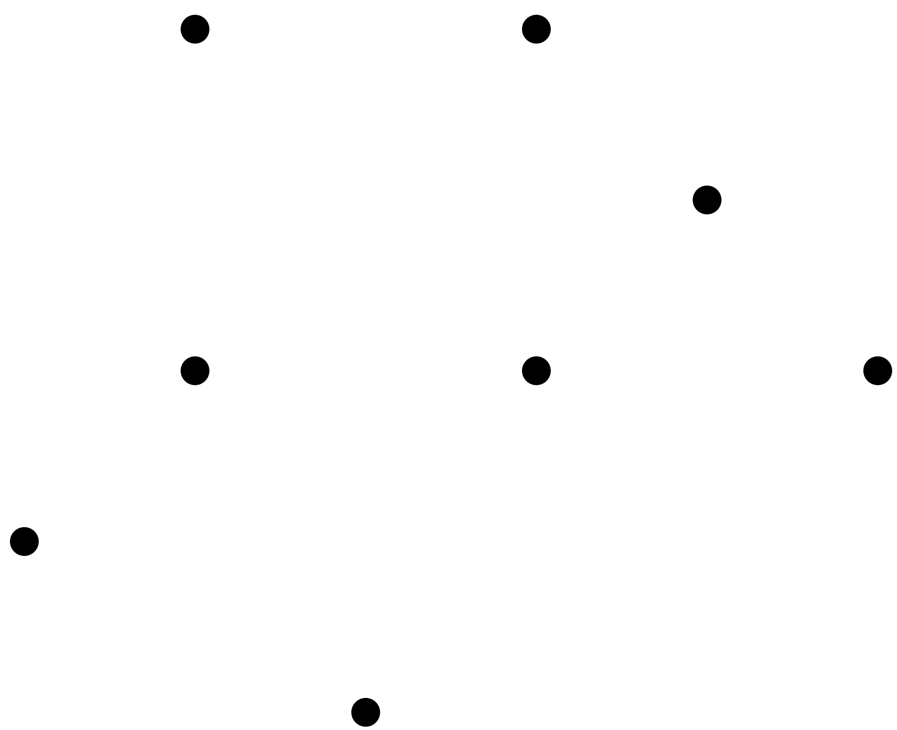


Algorithm

APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H

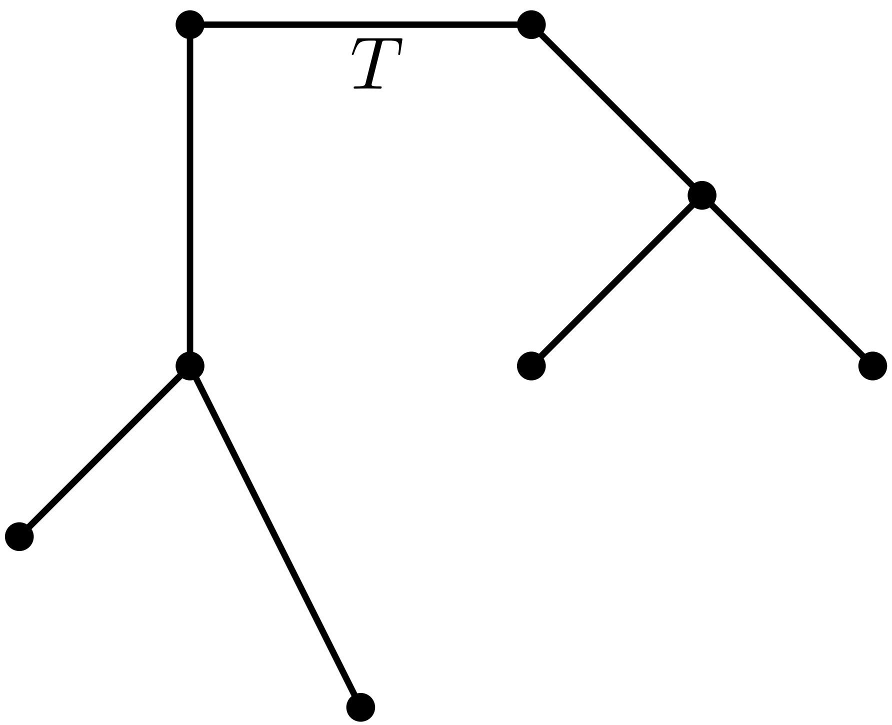
Algorithm

APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H



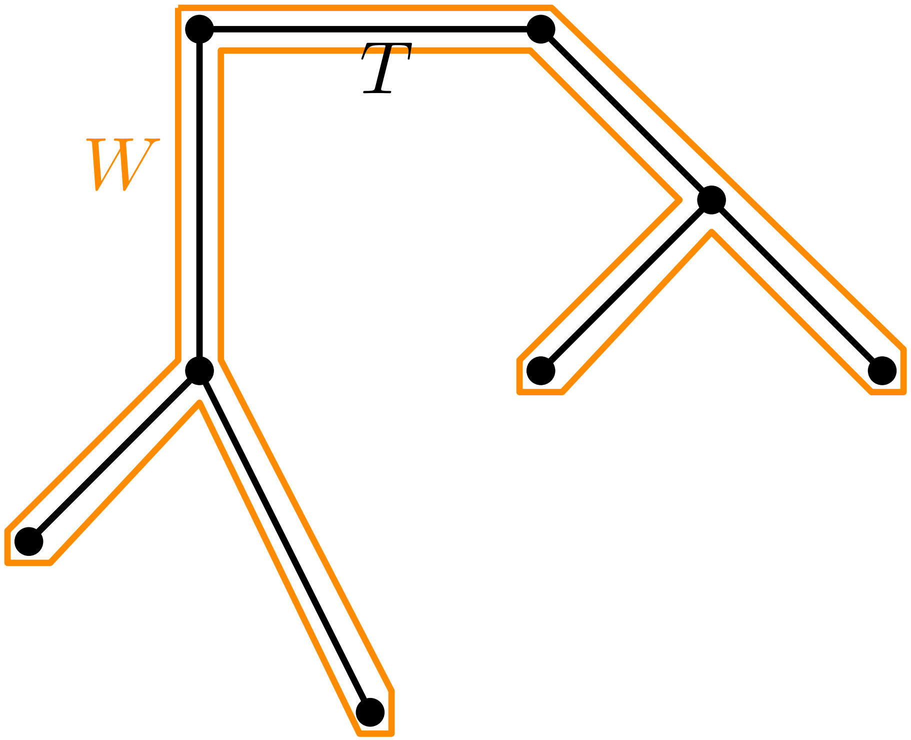
Algorithm

APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H



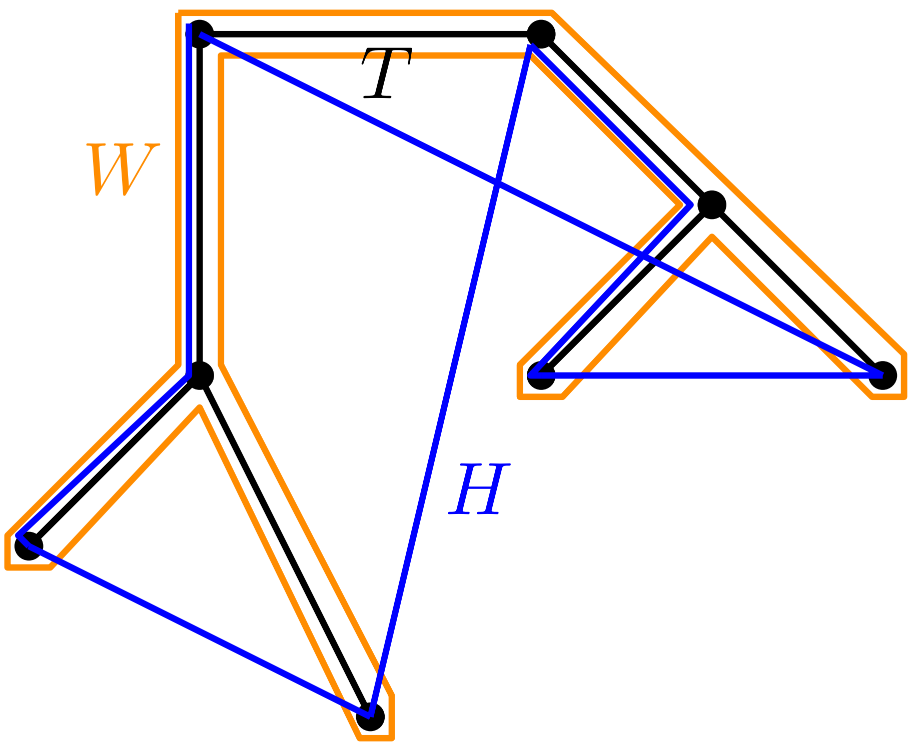
Algorithm

APPROX-TSP(G, c)
Find MST T
Make Euler tour W using each edge of T twice
Shortcut W to H by skipping duplicates
Return H



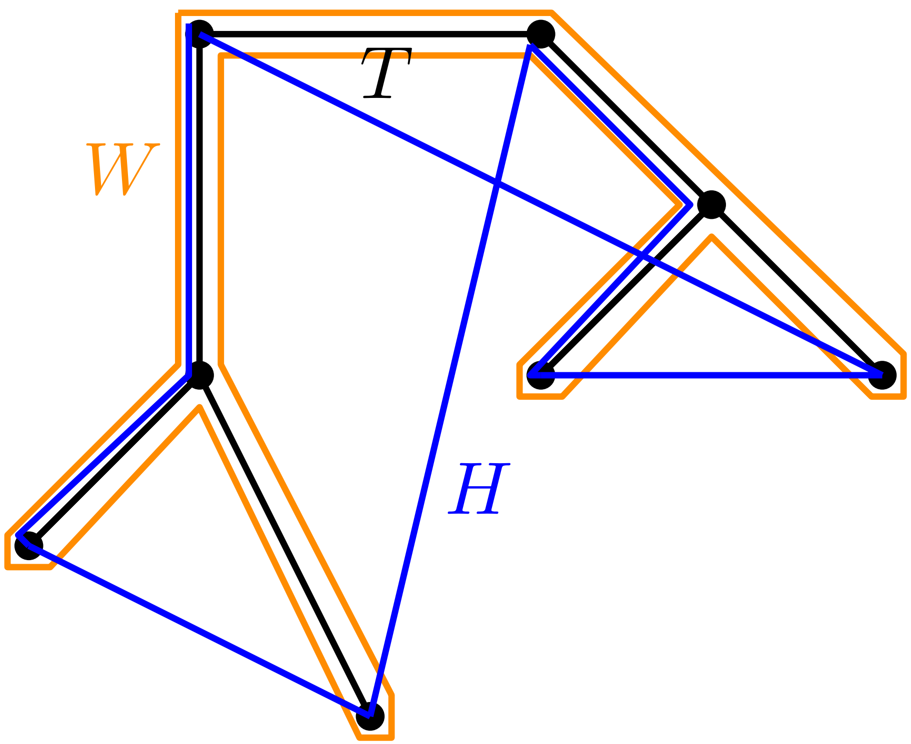
Algorithm

APPROX-TSP(G, c)
    Find MST T
    Make Euler tour W using each edge of T twice
    Shortcut W to H by skipping duplicates
    Return H

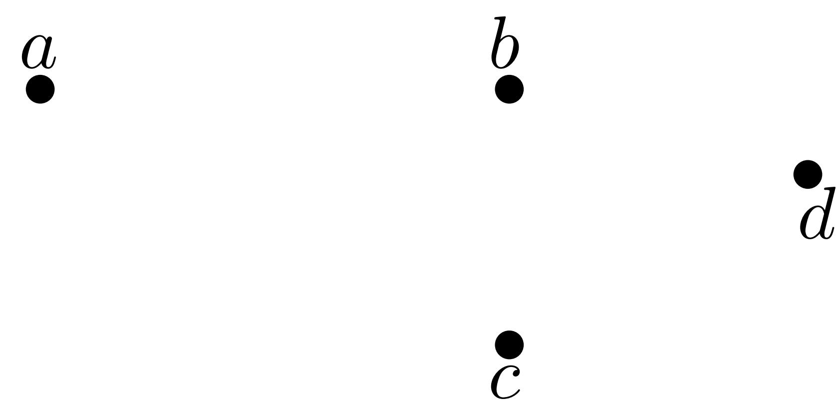


Algorithm

APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H

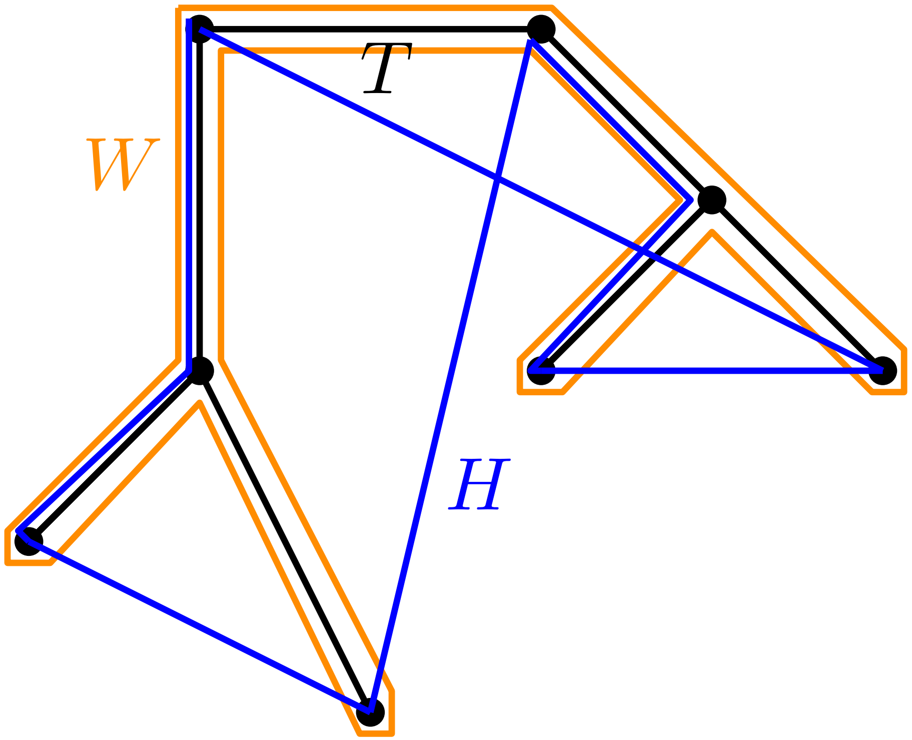


Exercise: Run the algorithm on this instance.



Theorem

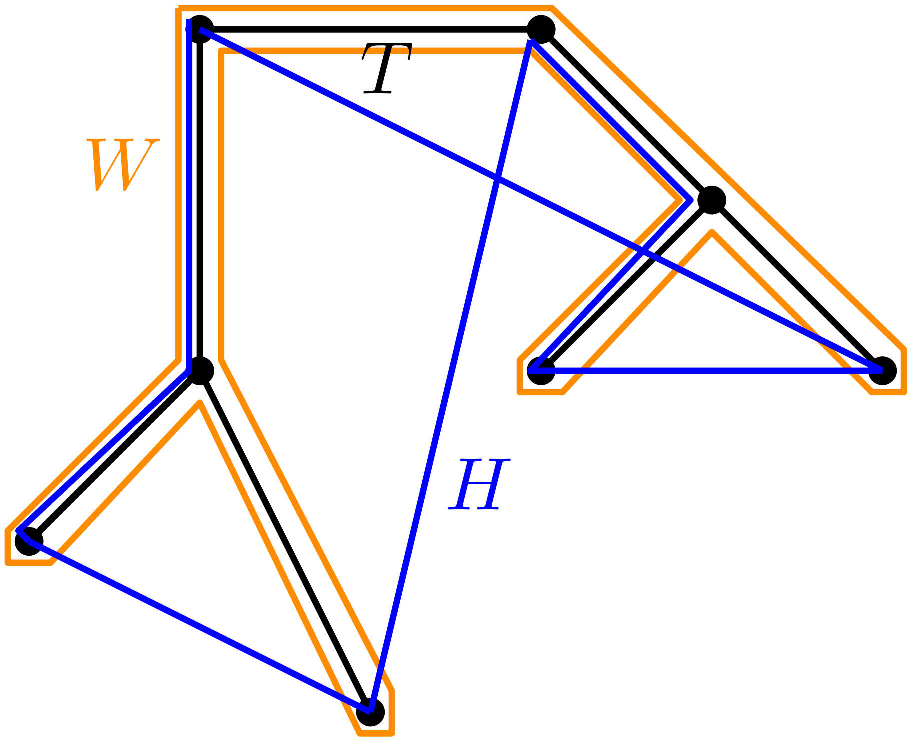
APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H



Thm.: APPROX-TSP is a poly-time 2-approx. alg.

Theorem

APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H



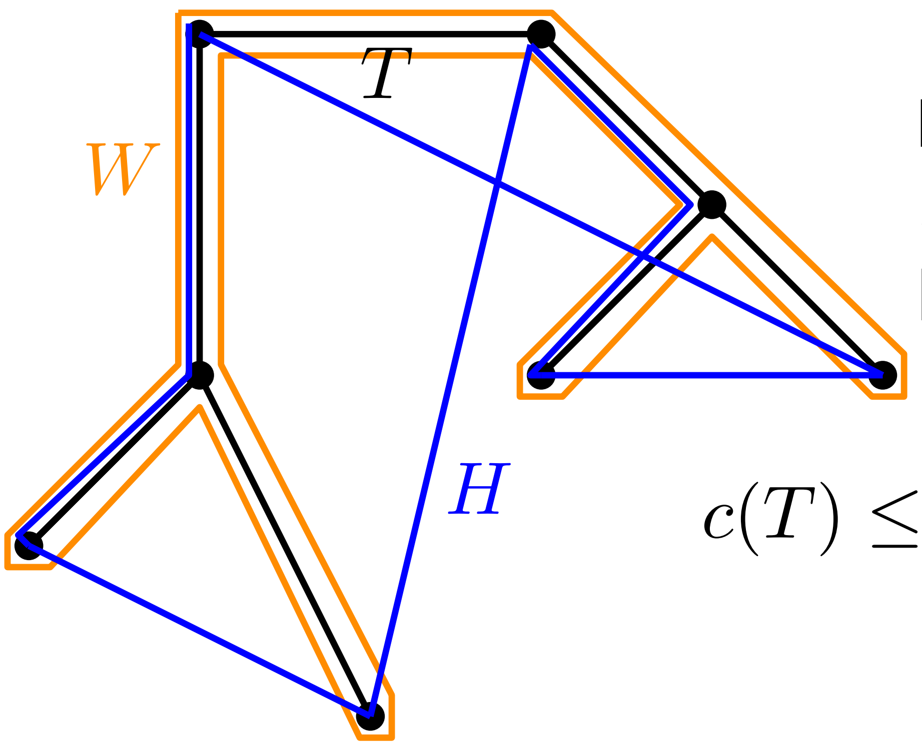
Thm.: APPROX-TSP is a poly-time 2-approx. alg.

Proof: Poly-time?

Let H^* be an opt. sol.

Theorem

APPROX-TSP(G, c)
Find MST T
Make Euler tour W using each edge of T twice
Shortcut W to H by skipping duplicates
Return H



Thm.: APPROX-TSP is a poly-time 2-approx. alg.

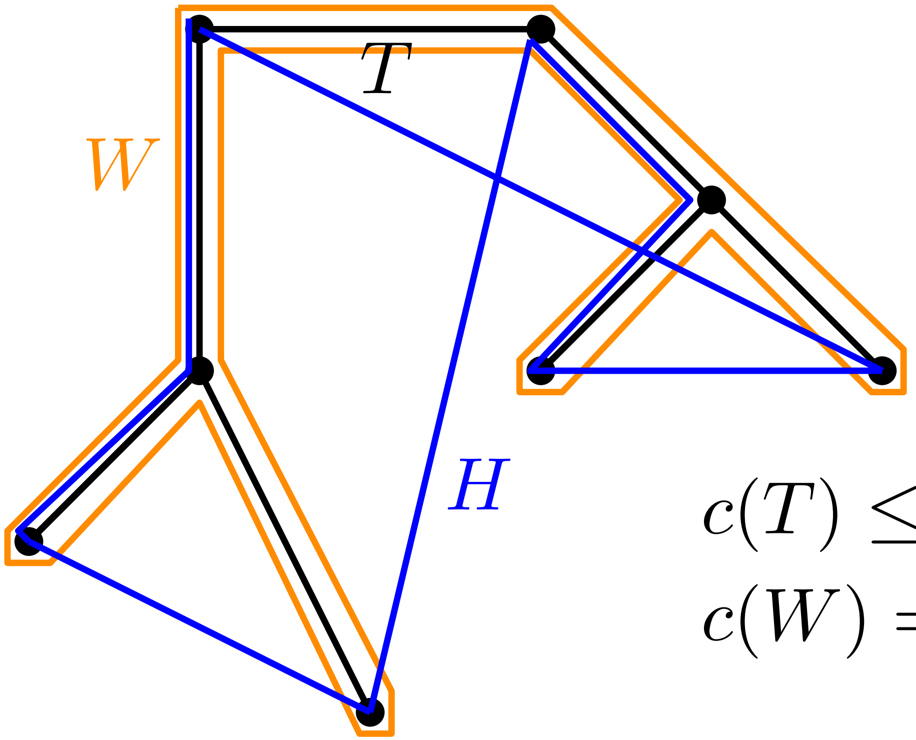
Proof: Poly-time?

Let H^* be an opt. sol.

$$c(T) \leq c(H^*)$$

Theorem

APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H



Thm.: APPROX-TSP is a poly-time 2-approx. alg.

Proof: Poly-time?

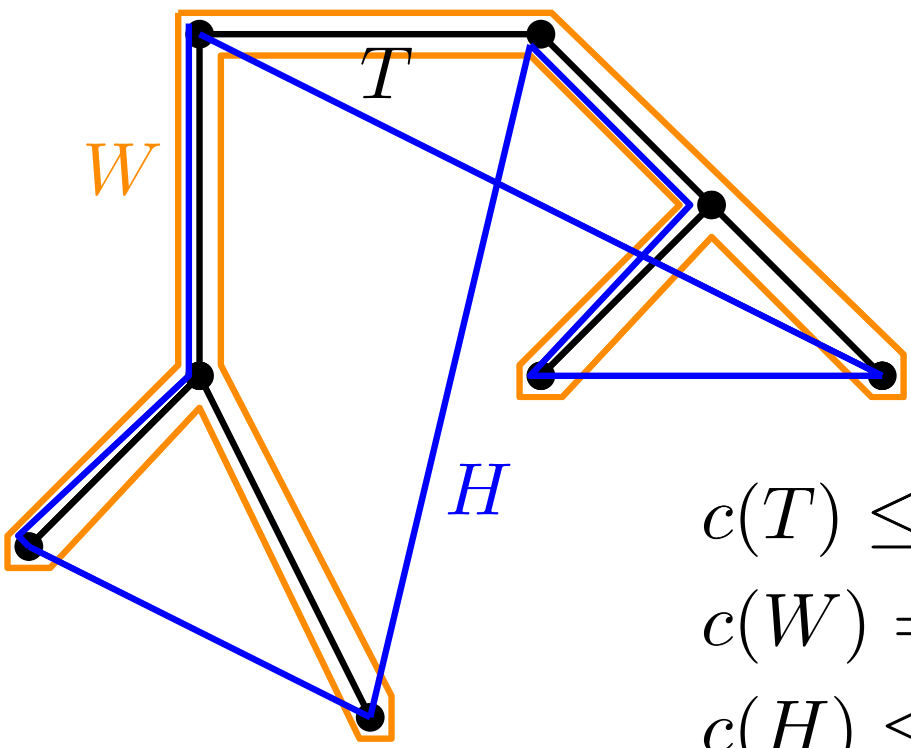
Let H^* be an opt. sol.

$$c(T) \leq c(H^*)$$

$$c(W) = 2c(T)$$

Theorem

APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H



Thm.: APPROX-TSP is a poly-time 2-approx. alg.

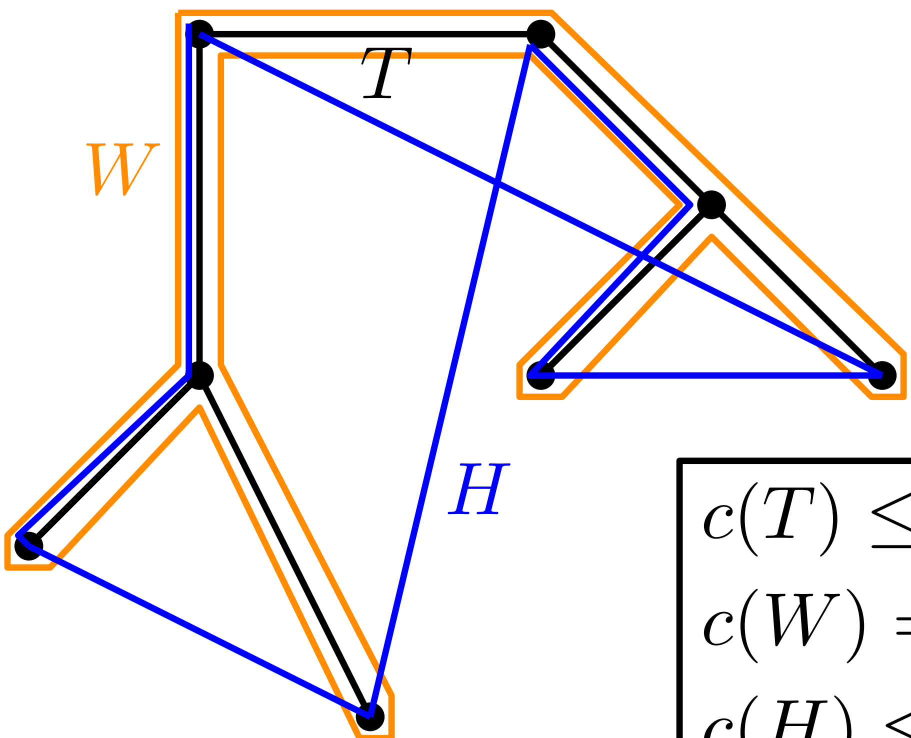
Proof: Poly-time?

Let H^* be an opt. sol.

$$c(T) \leq c(H^*)$$
$$c(W) = 2c(T)$$
$$c(H) \leq c(W)$$

Theorem

APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H



Thm.: APPROX-TSP is a poly-time 2-approx. alg.

Proof: Poly-time?

Let H^* be an opt. sol.

$$\boxed{\begin{array}{l} c(T) \le c(H^*) \\ c(W) = 2c(T) \\ c(H) \le c(W) \end{array}} \implies c(H) \le 2c(H^*)$$

Reflection and methodology

How can we prove $c(H)/c(H^*) \leq 2$ when we don't know H^*?

Answer: By proving $c(H) \leq 2c(T)$ and $c(T) \leq c(H^*)$.

Reflection and methodology

How can we prove $c(H)/c(H^*) \leq 2$ when we don't know H^*?

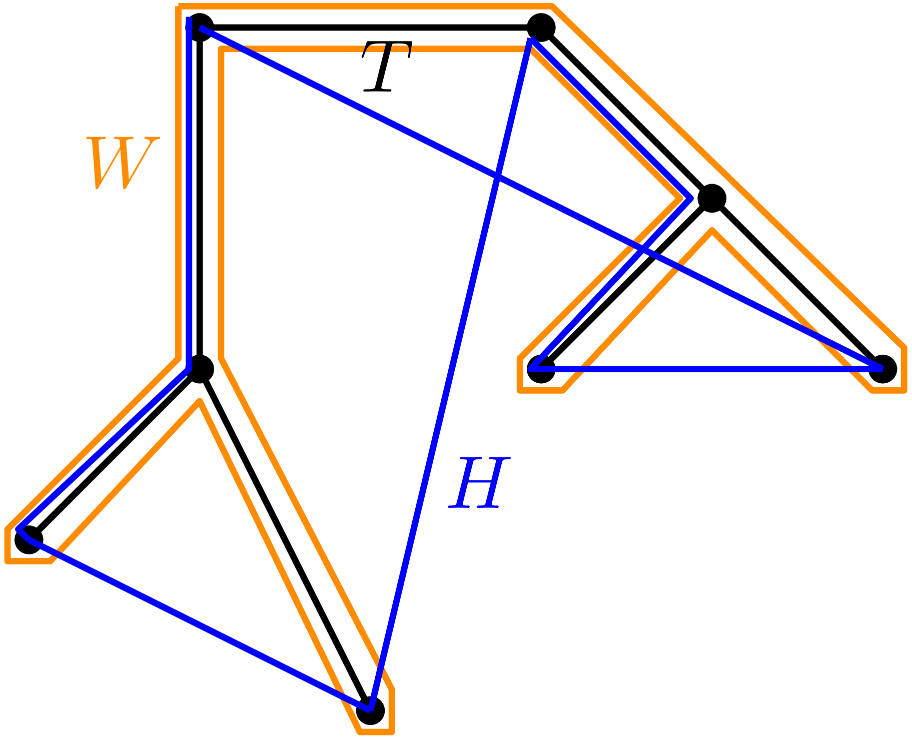
Answer: By proving $c(H) \leq 2c(T)$ and $c(T) \leq c(H^*)$.

General technique: Find a parameter $\Box$ such that $C \leq \rho \cdot \Box$ and $\Box \leq C^*$.

For TSP: $\Box = c(T)$ and $\rho = 2$.

Question

Try to guess: Is there an approximation algorithm with a better approximation ratio?



History

1976: Christofides, Serdyukov, 1.5-apx algorithm
It's simple! See, e.g., Wikipedia. No improvement for decades

2021: Karlin, Klein, Gharan, $(1.5 - \varepsilon)$-apx algorithm for some $\varepsilon > 10^{-36}$

Quanta magazine

Computer Scientists Break Traveling Salesperson Record

24

After 44 years, there's finally a better way to find approximate solutions to the notoriously difficult traveling salesperson problem.

Set Cover

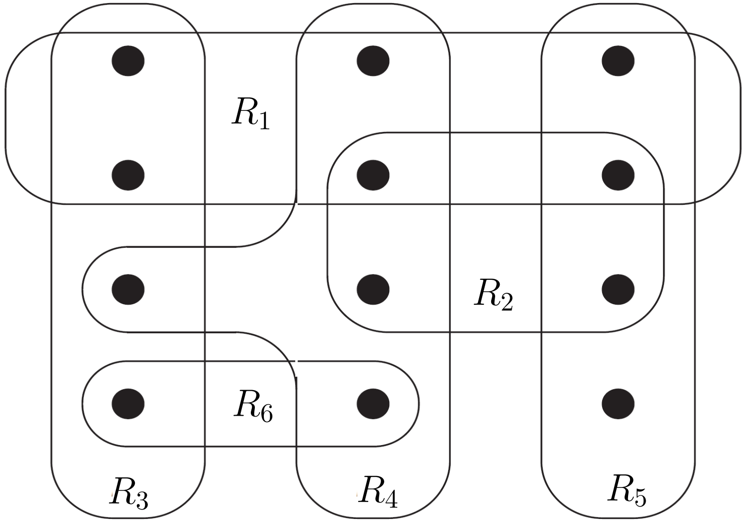
Input: Pair $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq \mathcal{P}(X)$ is a family of subsets of X.

Set Cover

Input: Pair $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq \mathcal{P}(X)$ is a family of subsets of X.
Goal: Find $\mathcal{C} \subseteq \mathcal{F}$ covering X, i.e., $\bigcup_{S \in \mathcal{C}} S = X$, with $|\mathcal{C}|$ minimum.

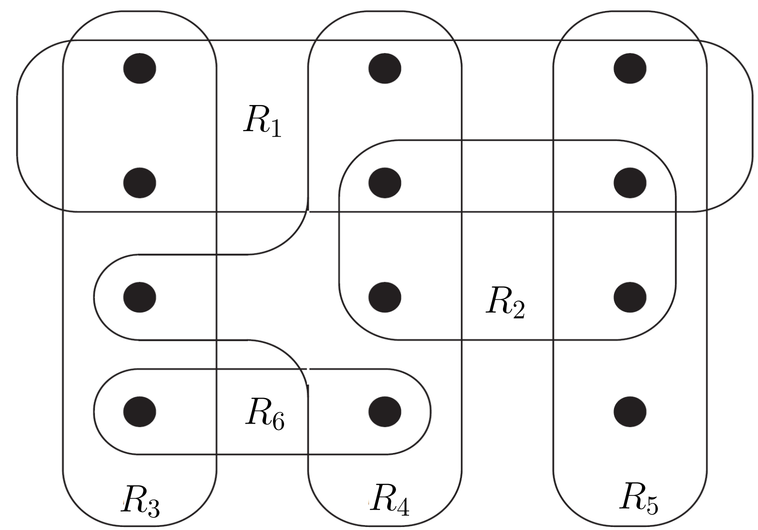
Set Cover

Input: Pair $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq \mathcal{P}(X)$ is a family of subsets of X.
Goal: Find $\mathcal{C} \subseteq \mathcal{F}$ covering X, i.e., $\bigcup_{S \in \mathcal{C}} S = X$, with $|\mathcal{C}|$ minimum.

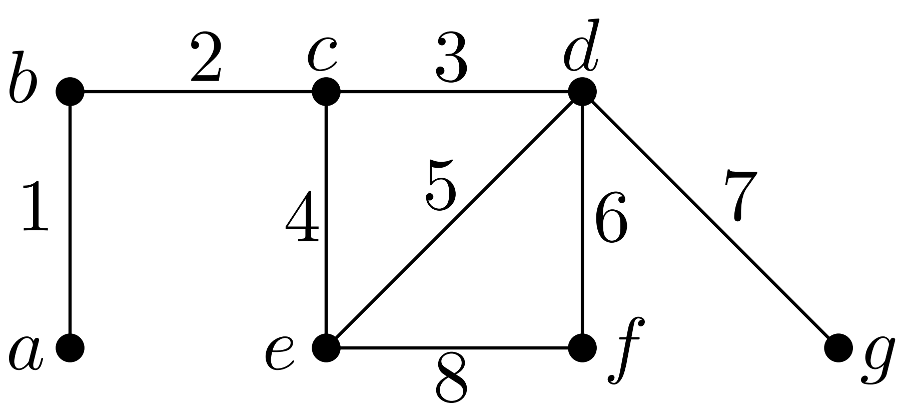


Set Cover

Input: Pair $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq \mathcal{P}(X)$ is a family of subsets of X.
Goal: Find $\mathcal{C} \subseteq \mathcal{F}$ covering X, i.e., $\bigcup_{S \in \mathcal{C}} S = X$, with $|\mathcal{C}|$ minimum.

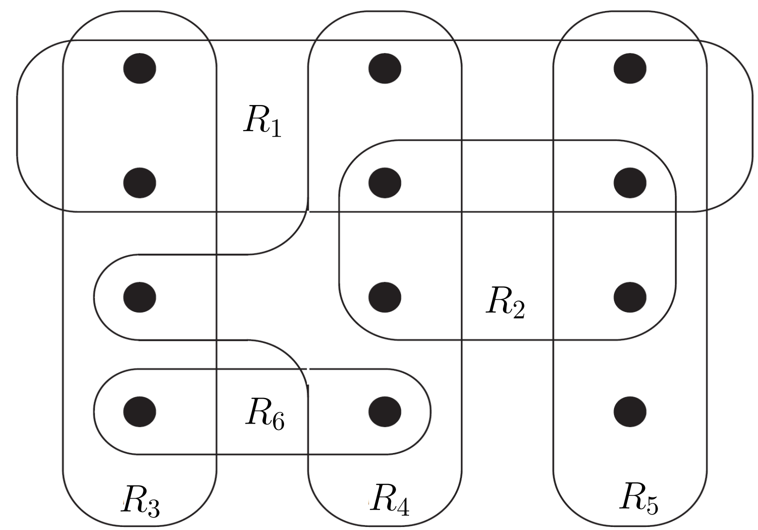


Exercise: Show that vertex cover is a special case.



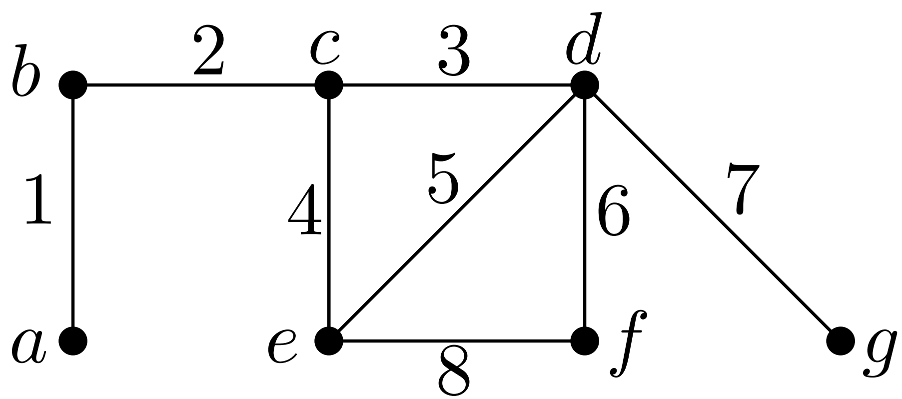
Set Cover

Input: Pair $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq \mathcal{P}(X)$ is a family of subsets of X.
Goal: Find $\mathcal{C} \subseteq \mathcal{F}$ covering X, i.e., $\bigcup_{S \in \mathcal{C}} S = X$, with $|\mathcal{C}|$ minimum.



Exercise: Show that vertex cover is a special case.

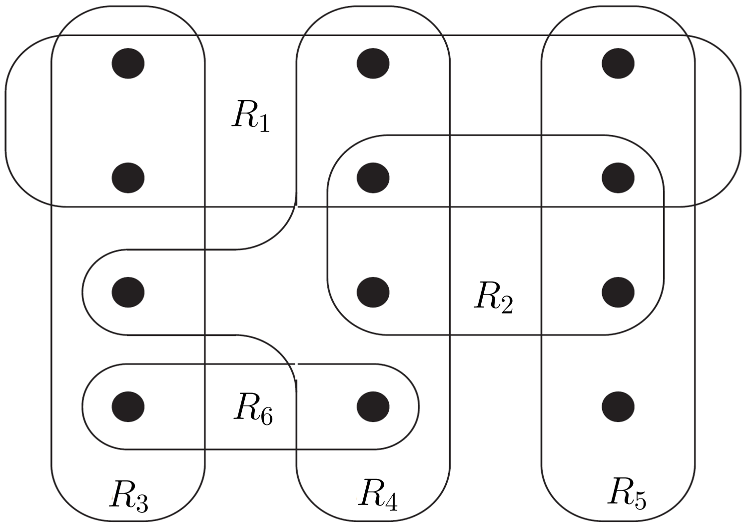
$X := \{1, 2, \ldots, 8\}$
$\mathcal{F} := \{\{1\}, \{1, 2\}, \{2, 3, 4\}, \{3, 5, 6\}, \{4, 5, 8\}, \{6, 8\}, \{7\}\}$



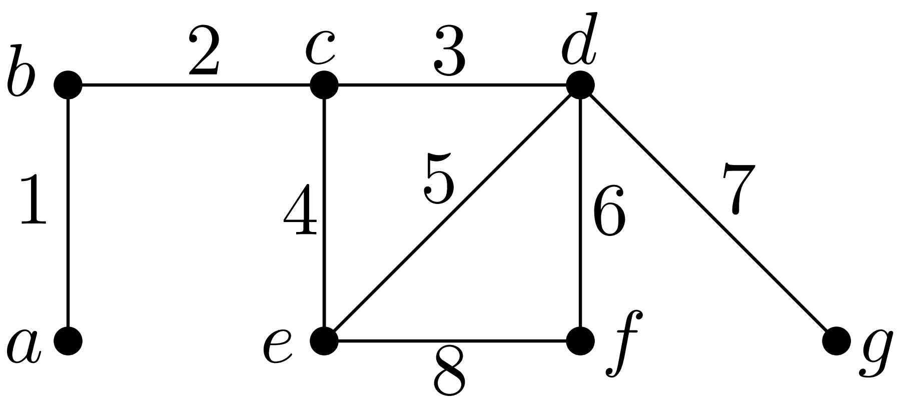
Set Cover

Input: Pair $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq \mathcal{P}(X)$ is a family of subsets of X.
Goal: Find $\mathcal{C} \subseteq \mathcal{F}$ covering X, i.e., $\bigcup_{S \in \mathcal{C}} S = X$, with $|\mathcal{C}|$ minimum.

Exercise: Show that vertex cover is a special case.



$$X := \{1, 2, \ldots, 8\}$$
$$\mathcal{F} := \{\{1\}, \{1, 2\}, \{2, 3, 4\}, \{3, 5, 6\}, \{4, 5, 8\}, \{6, 8\}, \{7\}\}$$



$$X := E$$
$$\mathcal{F} := \{E(v) \mid v \in V\}$$
$$E(v) := \{uv \in E \mid u \in V\}$$

Greedy Algorithm

GREEDY-SET-COVER($X, \mathcal{F}$)
- $i := 0$
- while $X \setminus S_{<i+1} \neq \emptyset$
  - $i := i + 1$
  - Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
- Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

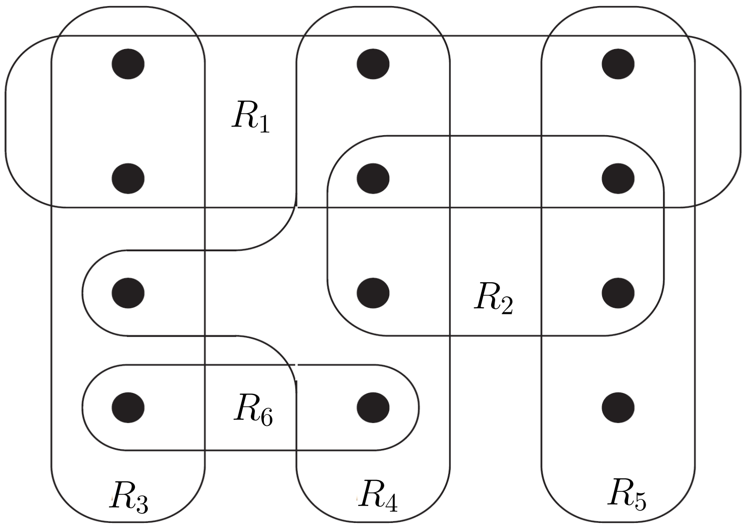
Here, $S_{<i} := \bigcup_{j=1}^{i-1} S_j$.

Greedy Algorithm

GREEDY-SET-COVER$(X, \mathcal{F})$
$i := 0$
while $X \setminus S_{<i+1} \neq \emptyset$
$i := i + 1$
Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

Here, $S_{<i} := \bigcup_{j=1}^{i-1} S_j$.

Exercise: Run the algorithm on this instance.



Greedy Algorithm

GREEDY-SET-COVER($X, \mathcal{F}$)

$i := 0$

while $X \setminus S_{<i+1} \neq \emptyset$

$i := i + 1$

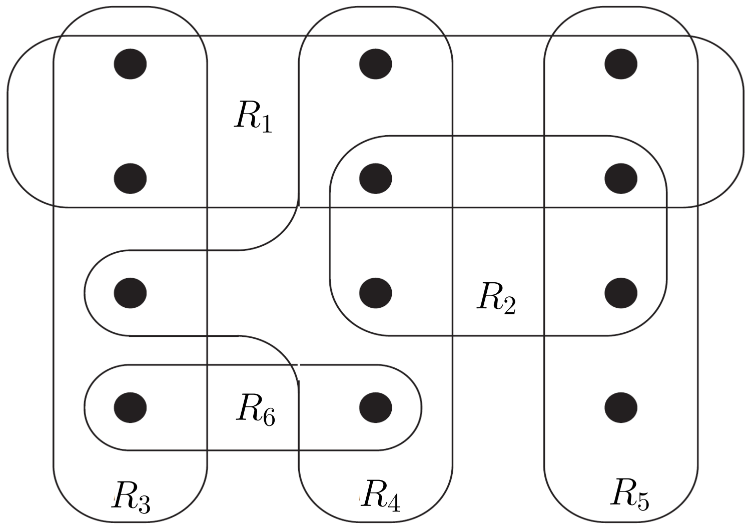
Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$

Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

Here, $S_{<i} := \bigcup_{j=1}^{i-1} S_j$.

Exercise: Run the algorithm on this instance.

$S_1 := R_1$



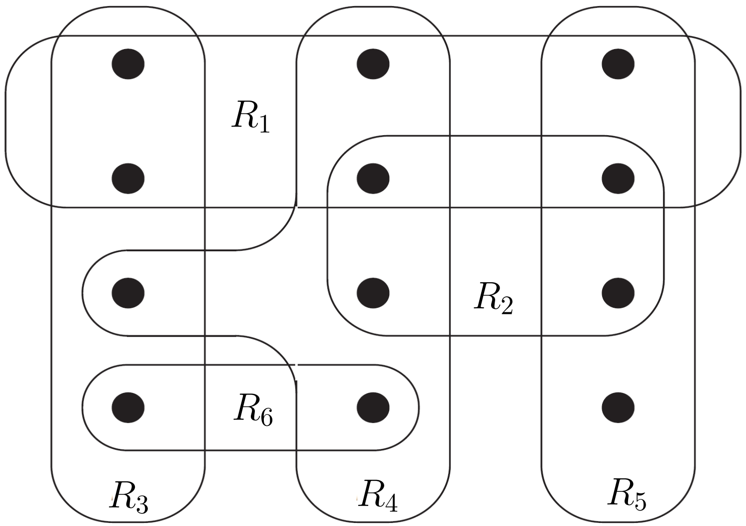
Greedy Algorithm

GREEDY-SET-COVER$(X, \mathcal{F})$
  $i := 0$
  while $X \setminus S_{<i+1} \neq \emptyset$
    $i := i + 1$
    Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
  Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

Here, $S_{<i} := \bigcup_{j=1}^{i-1} S_j$.

Exercise: Run the algorithm on this instance.

$S_1 := R_1$
$S_2 := R_4$



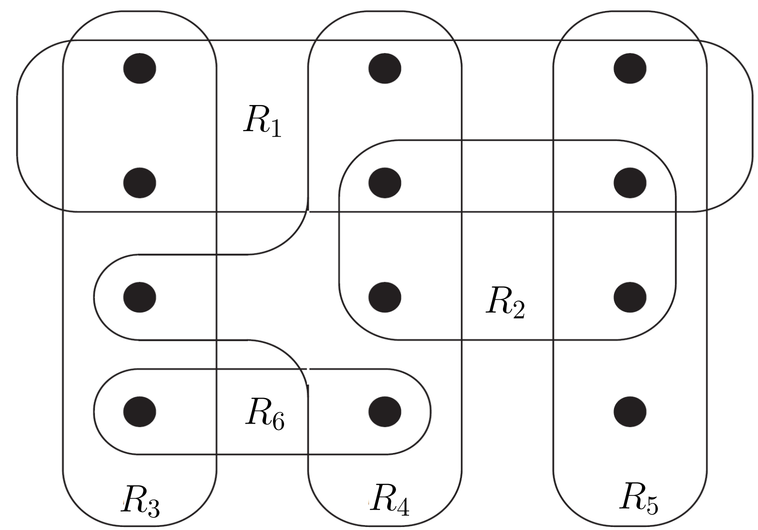
Greedy Algorithm

GREEDY-SET-COVER($X, \mathcal{F}$)
    $i := 0$
    while $X \setminus S_{<i+1} \neq \emptyset$
        $i := i + 1$
        Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
    Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

Here, $S_{<i} := \bigcup_{j=1}^{i-1} S_j$.

Exercise: Run the algorithm on this instance.

$S_1 := R_1$
$S_2 := R_4$
$S_3 := R_5$



Greedy Algorithm

GREEDY-SET-COVER$(X, \mathcal{F})$

$i := 0$

while $X \setminus S_{<i+1} \neq \emptyset$

$i := i + 1$

Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$

Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

Here, $S_{<i} := \bigcup_{j=1}^{i-1} S_j$.

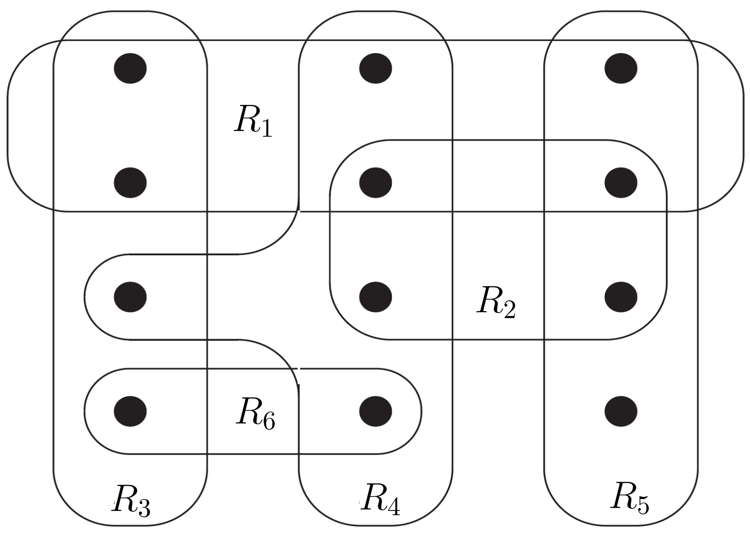
Exercise: Run the algorithm on this instance.

$S_1 := R_1$

$S_2 := R_4$

$S_3 := R_5$

$S_4 := R_3$ or $S_4 := R_6$



Theorem

Thm.: For opt. sol. $\mathcal{C}^*$, we have

$$|\mathcal{C}| \leq H_{|X|} \cdot |\mathcal{C}^*|,$$

where

$$H_n := \sum_{i=1}^{n} 1/i \leq \ln n + 1.$$

```
GREEDY-SET-COVER(X, F)
   i := 0
   while X \ S_{<i+1} ≠ ∅
      i := i + 1
      Pick S_i ∈ F with max |S_i \ S_{<i}|
   Return C := {S_1, ..., S_i}
```

Hence, GREEDY-SET-COVER is a $O(\log n)$-approx. alg.

Theorem

$\boxed{\textbf{Thm.: } |\mathcal{C}| \leq H_{|X|} \cdot |\mathcal{C}^*|.}$

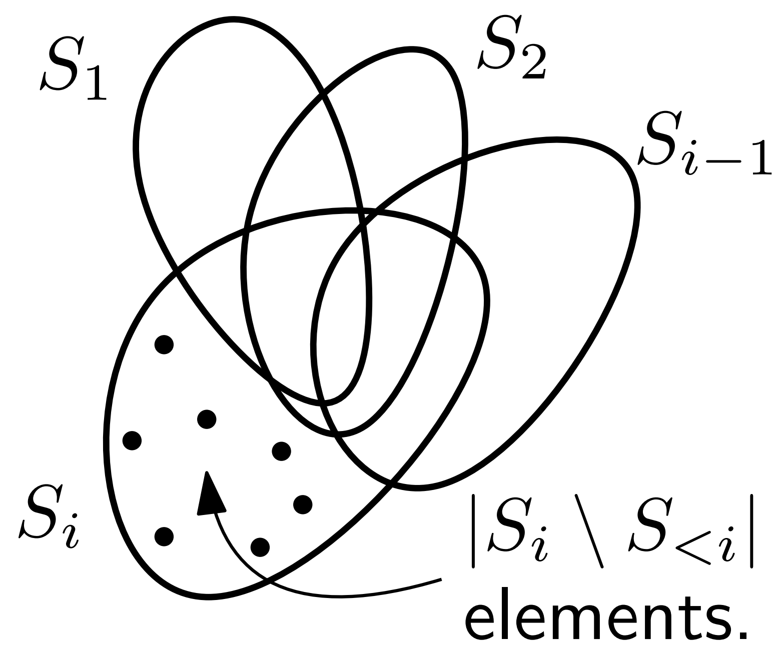
GREEDY-SET-COVER($X, \mathcal{F}$)
  $i := 0$
  while $X \setminus S_{<i+1} \neq \emptyset$
    $i := i + 1$
    Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
  Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

Theorem

$\boxed{\textbf{Thm.: } |\mathcal{C}| \leq H_{|X|} \cdot |\mathcal{C}^*|.}$

For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.
For $Y \subset X$, define $c(Y) := \sum_{x \in Y} c_x$.

```
GREEDY-SET-COVER(X, F)
  i := 0
  while X \ S_{<i+1} ≠ ∅
    i := i + 1
    Pick S_i ∈ F with max |S_i \ S_{<i}|
  Return C := {S_1, ..., S_i}
```



Theorem

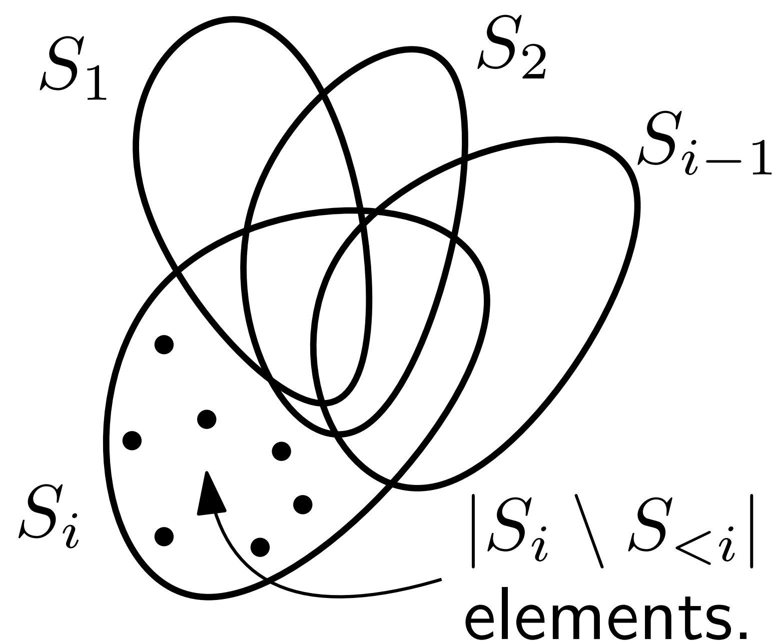
Thm.: $|\mathcal{C}| \leq H_{|X|} \cdot |\mathcal{C}^*|$.

For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.
For $Y \subset X$, define $c(Y) := \sum_{x \in Y} c_x$.

Observation:

$$c(X) = \sum_{i=1}^{|\mathcal{C}|} \sum_{x \in S_i \setminus S_{<i}} c_x = \sum_{i=1}^{|\mathcal{C}|} 1 = |\mathcal{C}|.$$

GREEDY-SET-COVER($X, \mathcal{F}$)
- $i := 0$
- while $X \setminus S_{<i+1} \neq \emptyset$
  - $i := i + 1$
  - Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
- Return $\mathcal{C} := \{S_1, \ldots, S_i\}$



Theorem

Thm.: $|\mathcal{C}| \leq H_{|X|} \cdot |\mathcal{C}^*|$.

For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.

For $Y \subset X$, define $c(Y) := \sum_{x \in Y} c_x$.

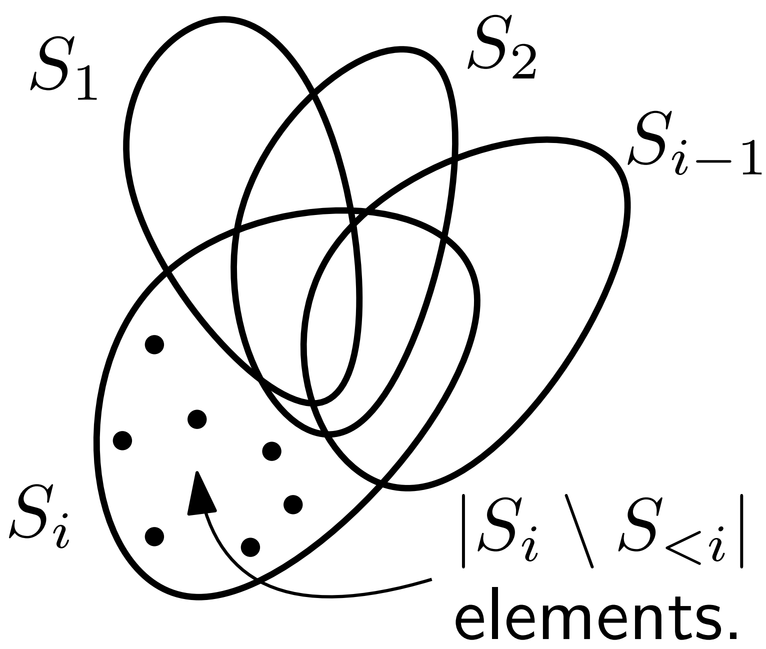
Observation:

$$c(X) = \sum_{i=1}^{|\mathcal{C}|} \sum_{x \in S_i \setminus S_{<i}} c_x = \sum_{i=1}^{|\mathcal{C}|} 1 = |\mathcal{C}|.$$

Lemma: For all $S \in \mathcal{F}$:

$$c(S) \leq \sum_{i=1}^{|S|} \frac{1}{i} = H_{|S|}.$$

GREEDY-SET-COVER$(X, \mathcal{F})$
- $i := 0$
- while $X \setminus S_{<i+1} \neq \emptyset$
  - $i := i + 1$
  - Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
- Return $\mathcal{C} := \{S_1, \ldots, S_i\}$



Theorem

Thm.: $|\mathcal{C}| \leq H_{|X|} \cdot |\mathcal{C}^*|$.

GREEDY-SET-COVER($X, \mathcal{F}$)
  $i := 0$
  while $X \setminus S_{<i+1} \neq \emptyset$
    $i := i + 1$
    Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
  Return $\mathcal{C} := \{S_1, \dots, S_i\}$

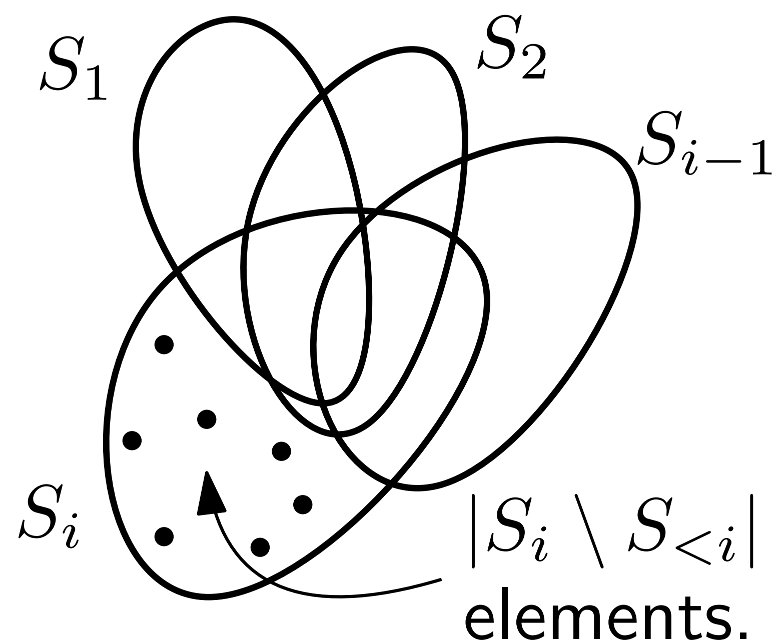
For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.
For $Y \subset X$, define $c(Y) := \sum_{x \in Y} c_x$.

Observation:

$$c(X) = \sum_{i=1}^{|\mathcal{C}|} \sum_{x \in S_i \setminus S_{<i}} c_x = \sum_{i=1}^{|\mathcal{C}|} 1 = |\mathcal{C}|.$$

Lemma: For all $S \in \mathcal{F}$:

$$c(S) \leq \sum_{i=1}^{|S|} \frac{1}{i} = H_{|S|}.$$



Proof of Thm.:

$$|\mathcal{C}| = c(X) \leq \sum_{S \in \mathcal{C}^*} c(S) \leq \sum_{S \in \mathcal{C}^*} H_{|S|} \leq \sum_{S \in \mathcal{C}^*} H_{|X|} = |\mathcal{C}^*| \cdot H_{|X|}.$$

Lemma: Idea and Example

Lemma: For all $S \in \mathcal{F}$:

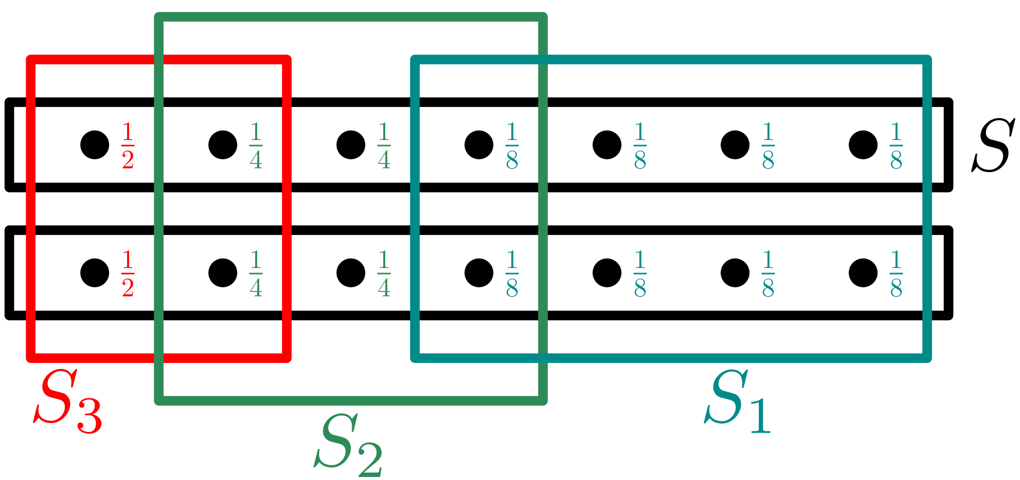
$$c(S) \le \sum_{i=1}^{|S|} \frac{1}{i} = H_{|S|}.$$

GREEDY-SET-COVER($X, \mathcal{F}$)
  $i := 0$
  while $X \setminus S_{<i+1} \neq \emptyset$
    $i := i + 1$
    Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
  Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.
For $Y \subset X$, define $c(Y) := \sum_{x \in Y} c_x$.

Idea: 1st element in S to be covered has $c_x \le \frac{1}{|S|}$, 2nd has $c_x \le \frac{1}{|S|-1}$, ...

Example:



$$\begin{aligned} c(S) &= \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} \\ &\le 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} \\ &= H_{|S|}. \end{aligned}$$

Proof of Lemma

Lemma: For all $S \in \mathcal{F}$:

$$c(S) \leq \sum_{i=1}^{|S|} \frac{1}{i} = H_{|S|}.$$

GREEDY-SET-COVER$(X, \mathcal{F})$
    $i := 0$
    while $X \setminus S_{<i+1} \neq \emptyset$
        $i := i + 1$
        Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
    Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.
For $Y \subset X$, define $c(Y) := \sum_{x \in Y} c_x$.

Proof: Let $S = \{x_k, x_{k-1}, \ldots, x_1\}$, where x_k covered first, then x_{k-1}, etc. (break ties arbitrarily).

Proof of Lemma

Lemma: For all $S \in \mathcal{F}$:

$$c(S) \le \sum_{i=1}^{|S|} \frac{1}{i} = H_{|S|}.$$

```
GREEDY-SET-COVER(X, F)
  i := 0
  while X \ S_{<i+1} ≠ ∅
    i := i + 1
    Pick S_i ∈ F with max |S_i \ S_{<i}|
  Return C := {S_1, ..., S_i}
```

For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.
For $Y \subset X$, define $c(Y) := \sum_{x \in Y} c_x$.

Proof: Let $S = \{x_k, x_{k-1}, \ldots, x_1\}$, where x_k covered first, then x_{k-1}, etc. (break ties arbitrarily).

x_j covered first by $S_i \implies |S \setminus S_{<i}| \ge j$
(since $S \setminus S_{<i}$ contains $x_j, x_{j-1}, \ldots, x_1$)

Proof of Lemma

Lemma: For all $S \in \mathcal{F}$:

$$c(S) \le \sum_{i=1}^{|S|} \frac{1}{i} = H_{|S|}.$$

GREEDY-SET-COVER$(X, \mathcal{F})$
  $i := 0$
  while $X \setminus S_{<i+1} \neq \emptyset$
    $i := i + 1$
    Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
  Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.
For $Y \subset X$, define $c(Y) := \sum_{x \in Y} c_x$.

Proof: Let $S = \{x_k, x_{k-1}, \ldots, x_1\}$, where x_k covered first, then x_{k-1}, etc. (break ties arbitrarily).

x_j covered first by $S_i \implies |S \setminus S_{<i}| \ge j$
(since $S \setminus S_{<i}$ contains $x_j, x_{j-1}, \ldots, x_1$)

$$|S_i \setminus S_{<i}| \ge |S \setminus S_{<i}| \ge j \implies c_{x_j} = \frac{1}{|S_i \setminus S_{<i}|} \le \frac{1}{j}.$$

by greedy choice of S_i

Proof of Lemma

Lemma: For all $S \in \mathcal{F}$:

$$c(S) \leq \sum_{i=1}^{|S|} \frac{1}{i} = H_{|S|}.$$

GREEDY-SET-COVER($X, \mathcal{F}$)
  $i := 0$
  while $X \setminus S_{<i+1} \neq \emptyset$
    $i := i + 1$
    Pick $S_i \in \mathcal{F}$ with max $|S_i \setminus S_{<i}|$
  Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.
For $Y \subset X$, define $c(Y) := \sum_{x \in Y} c_x$.

Proof: Let $S = \{x_k, x_{k-1}, \ldots, x_1\}$, where x_k covered first, then x_{k-1}, etc. (break ties arbitrarily).

x_j covered first by $S_i \implies |S \setminus S_{<i}| \geq j$
(since $S \setminus S_{<i}$ contains $x_j, x_{j-1}, \ldots, x_1$)

$$|S_i \setminus S_{<i}| \geq |S \setminus S_{<i}| \geq j \implies c_{x_j} = \frac{1}{|S_i \setminus S_{<i}|} \leq \frac{1}{j}.$$

by greedy choice of S_i

$$c(S) = c_{x_1} + c_{x_2} + \ldots + c_{x_k} \leq 1 + \frac{1}{2} + \ldots + \frac{1}{k} = H_{|S|}$$

Using greedy algorithm for vertex cover

GREEDY-VERTEX-COVER(G)
    $C := \emptyset$
    while $E \neq \emptyset$
        Choose $v \in V$ of maximum degree
        $C := C \cup \{u\}$
        Remove edges incident to u from E
    return C

Using greedy algorithm for vertex cover

```
GREEDY-VERTEX-COVER(G)
  C := ∅
  while E ≠ ∅
    Choose v ∈ V of maximum degree
    C := C ∪ {u}
    Remove edges incident to u from E
  return C
```

Exercise: Find graph G where GREEDY-VERTEX-COVER does not produce optimal solution.

Using greedy algorithm for vertex cover

```
GREEDY-VERTEX-COVER(G)
    C := ∅
    while E ≠ ∅
        Choose v ∈ V of maximum degree
        C := C ∪ {u}
        Remove edges incident to u from E
    return C
```

Exercise: Find graph G where GREEDY-VERTEX-COVER does not produce optimal solution.

The algorithm only gives a $\Theta(\log |E|)$-approximation.