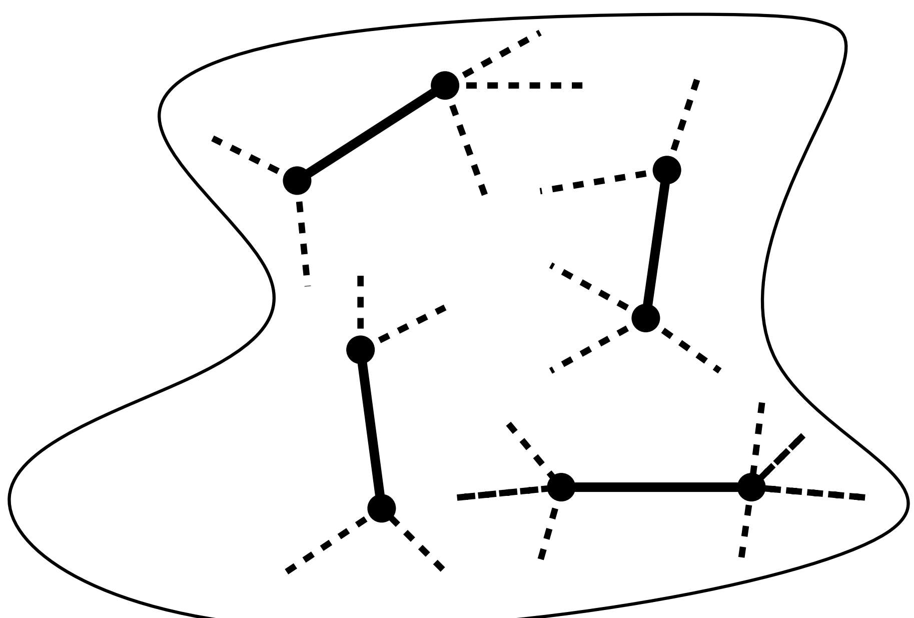
Introduction to Approximation Algorithms, part I

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,

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

$C^* :=$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

至少为1

$$\frac{\text{our solution}}{\text{optimal solution}} \geq 1$$

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

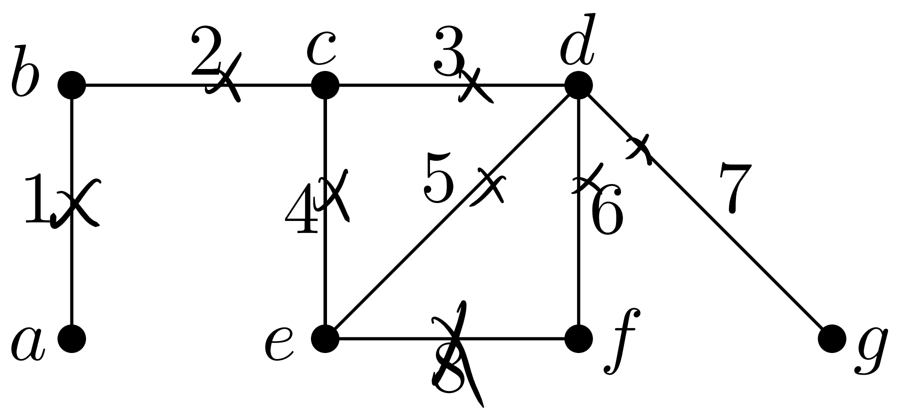
Def.: Let $G = (V, E)$ be a graph. A set $V' \subseteq V$ of vertices is a *vertex cover* if for all $uv \in E$, we have $u \in V'$ or $v \in V'$.

NP-hard!

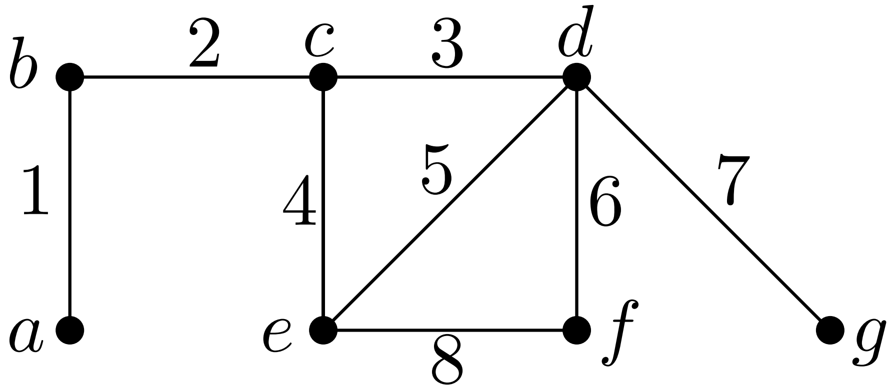
```
APPROX-VERTEX-COVER(G)
   C := ∅
   while E(G) ≠ ∅
      choose uv ∈ E(G)
      C := C ∪ {u, v}
      remove all edges incident on u or v from E(G)
   return C
```

Exercise:

$C = \{a, b, c, d, e, f\}$



Implementation



Adjacency lists:

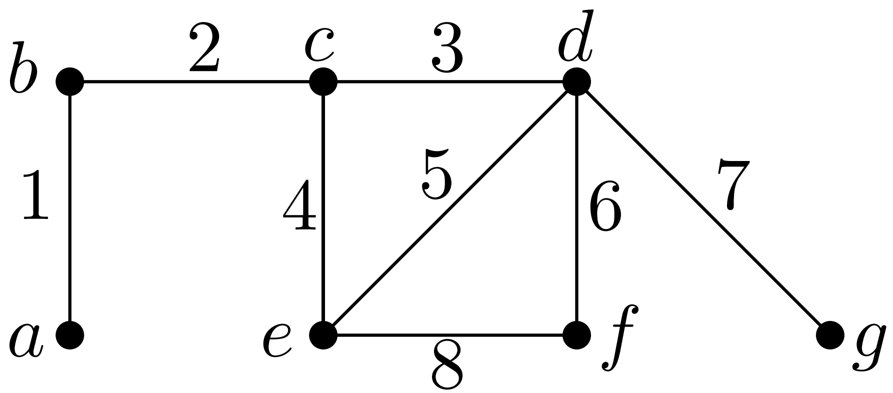
$L[a] = \{b\}$

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

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

$\vdots$

Implementation



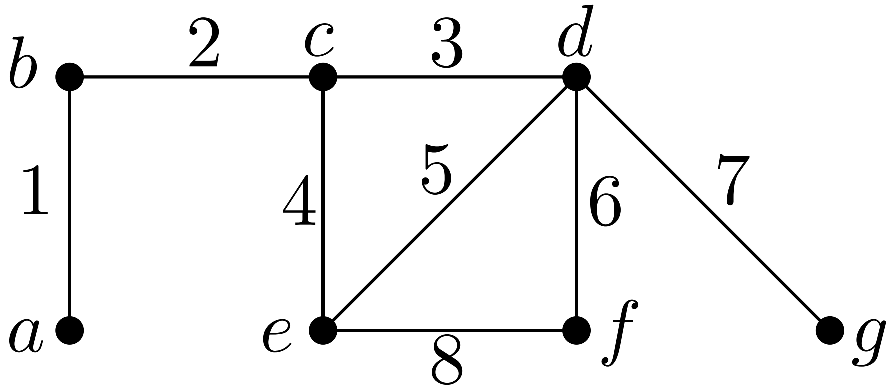
Adjacency lists:

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

Array of edges

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

Implementation



Adjacency lists:

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

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

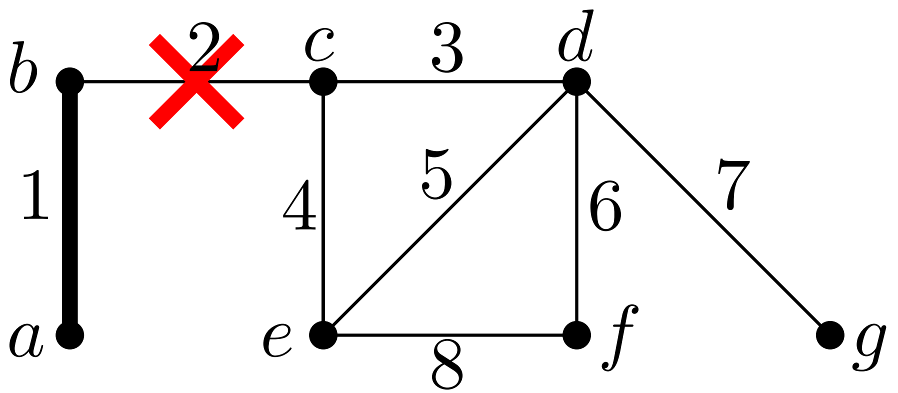
$L[c] = \{b, d, e\}$ $\quad E[c] = [2, 3, 4]$

$\vdots \qquad \vdots$

Array of edges

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

Implementation



Adjacency lists:

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

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

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

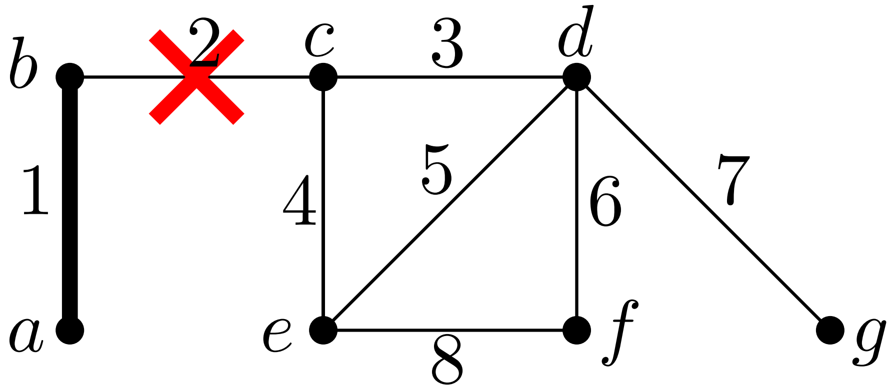
$\vdots$ $\vdots$

Array of edges

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

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

Implementation



Adjacency lists:

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

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

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

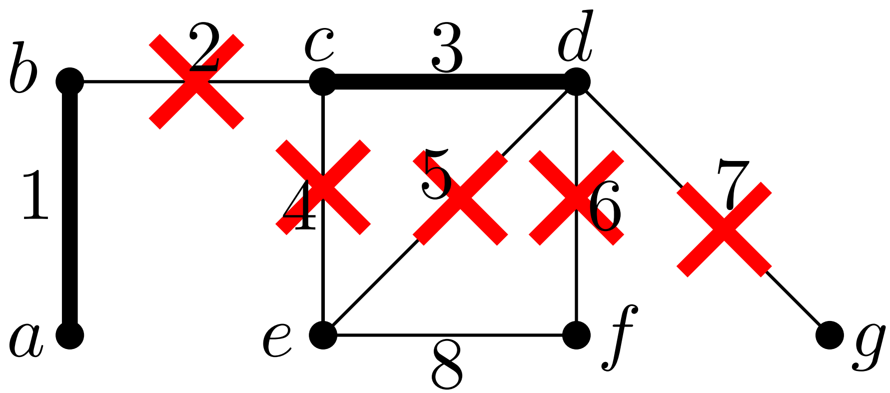
$\vdots \qquad \vdots$

Array of edges

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

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

Implementation



Adjacency lists:

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

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

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

$\vdots \qquad \vdots$

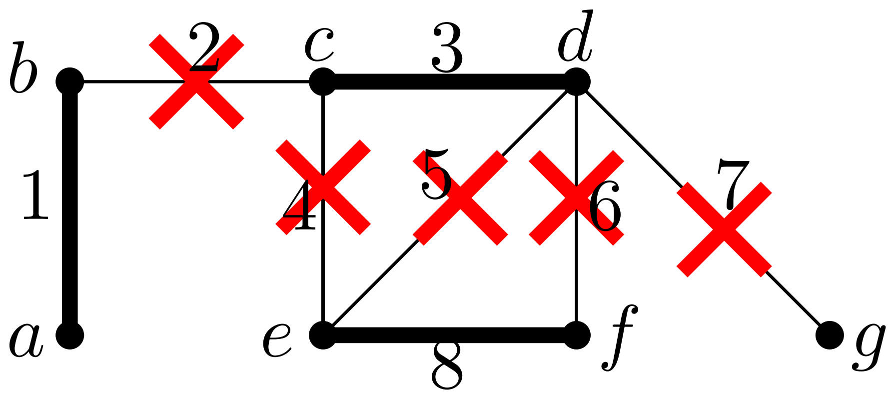
Array of edges

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

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

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

Implementation



Adjacency lists:

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

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

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

$\vdots \qquad \vdots$

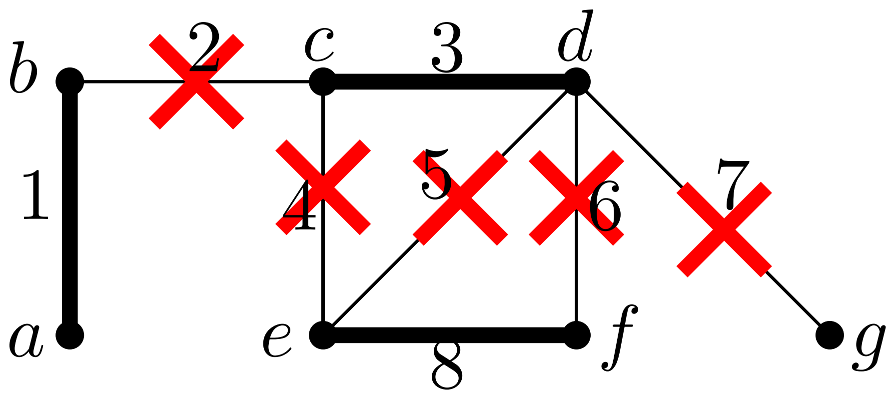
Array of edges

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

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

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

Implementation



Adjacency lists:

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

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

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

$\vdots \qquad \vdots$

Array of edges

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

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

the size of graph

Theorem

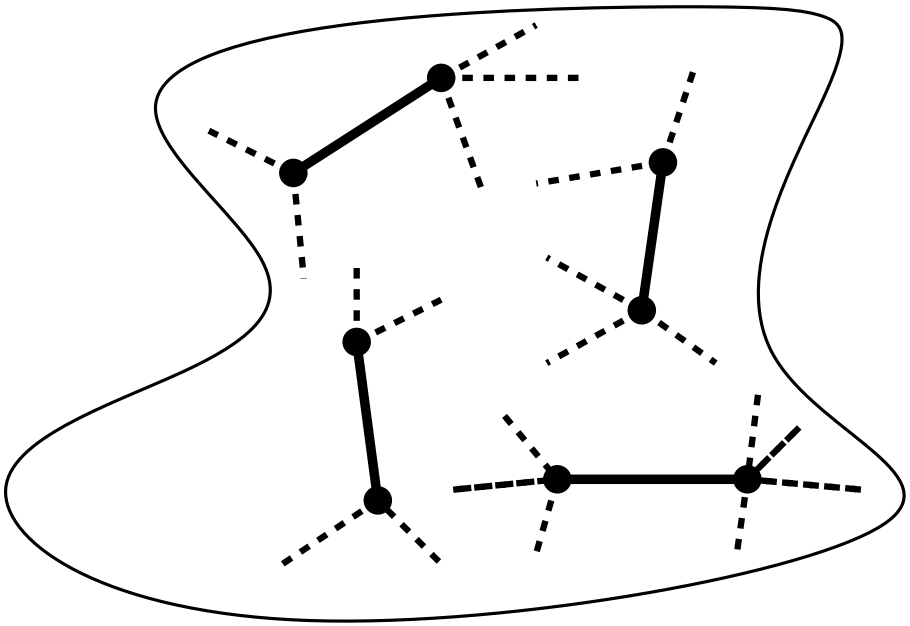
Thm.: APPROX-VERTEX-COVER is a 2-approximation algorithm.

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.

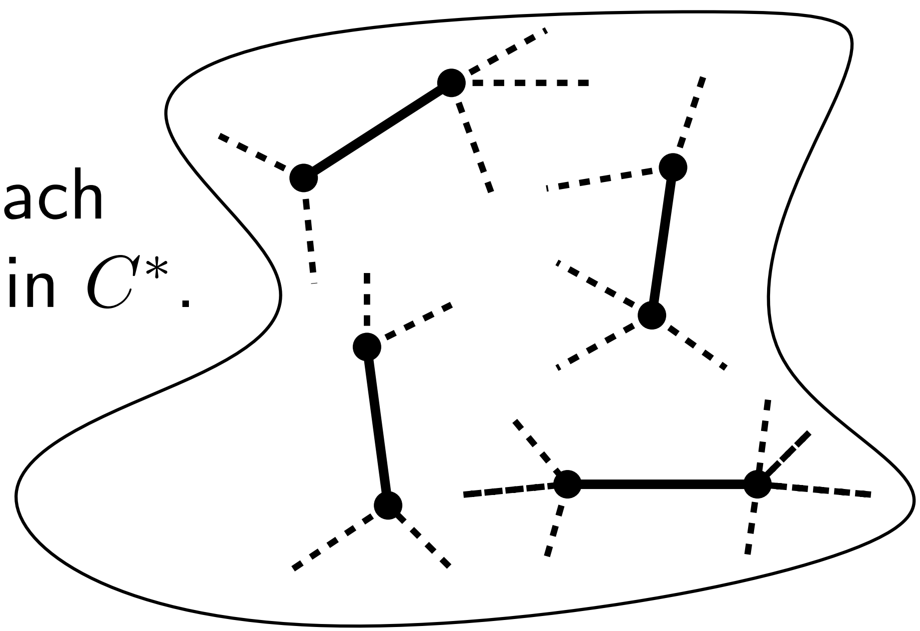


Theorem

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.

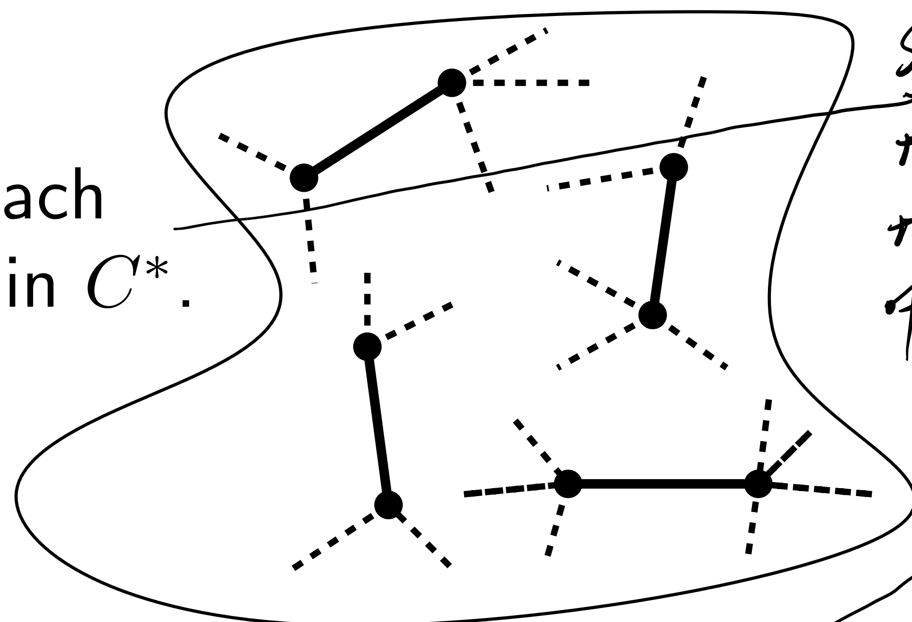
An endpoint of each $uv \in A$ must be in C^*.



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.



Since it is a cover, then it hits all the edges. So one of the endpoints in each of these are in the C^*.

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

Hence,

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

The edges, they don't share an endpoint, so ~~therefore~~, so we have $\geq$

in A

in C^*, we have a vertex for each edge in A.

可能 C^* 会选 2 个点 e

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^*?

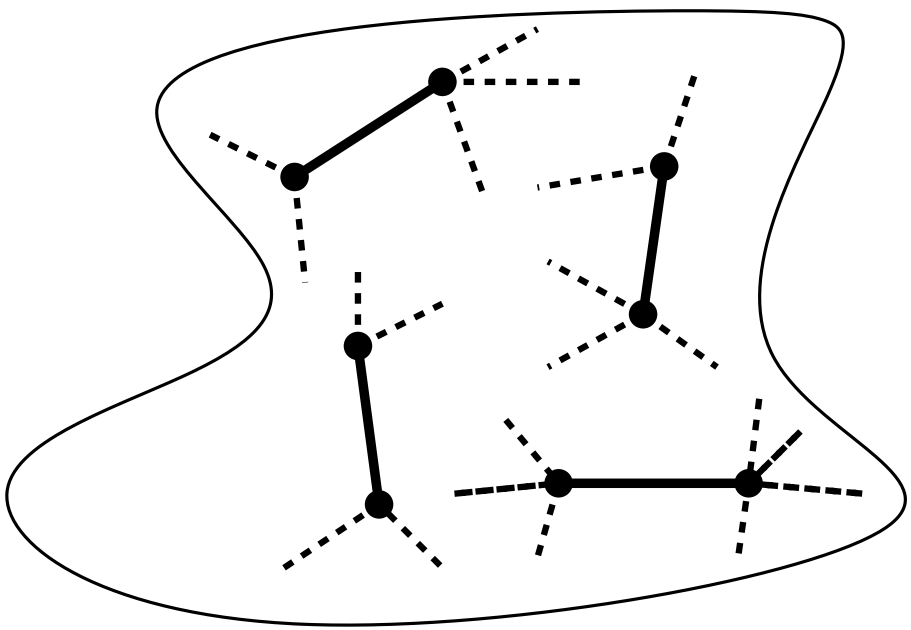
Answer: By proving $C \leq 2|A|$ and $|A| \leq C^*$.

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

For vertex cover: $\Box = |A|$ 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)

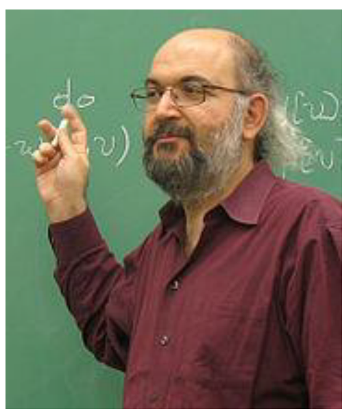
Karp

Turing Award

History

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

Gavril



Yannakakis

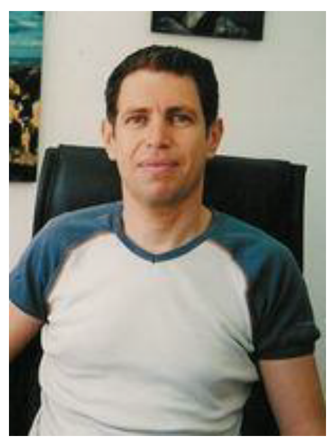
...

History

Assuming P$\neq$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.

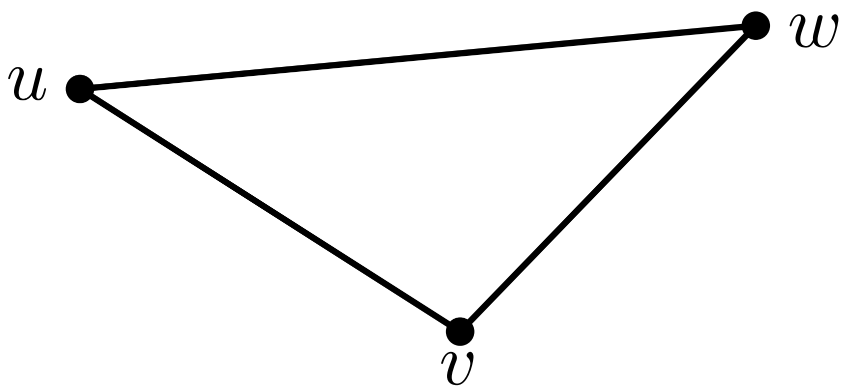
Traveling Salesperson

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

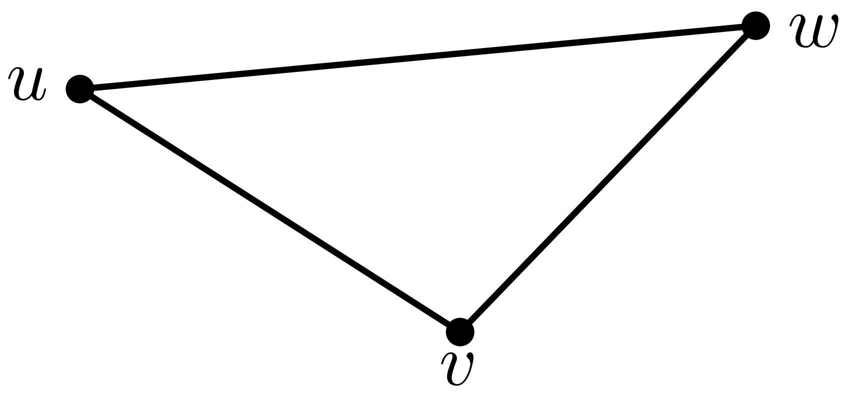
Given a complete undirected graph $G = (V, E)$.

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!

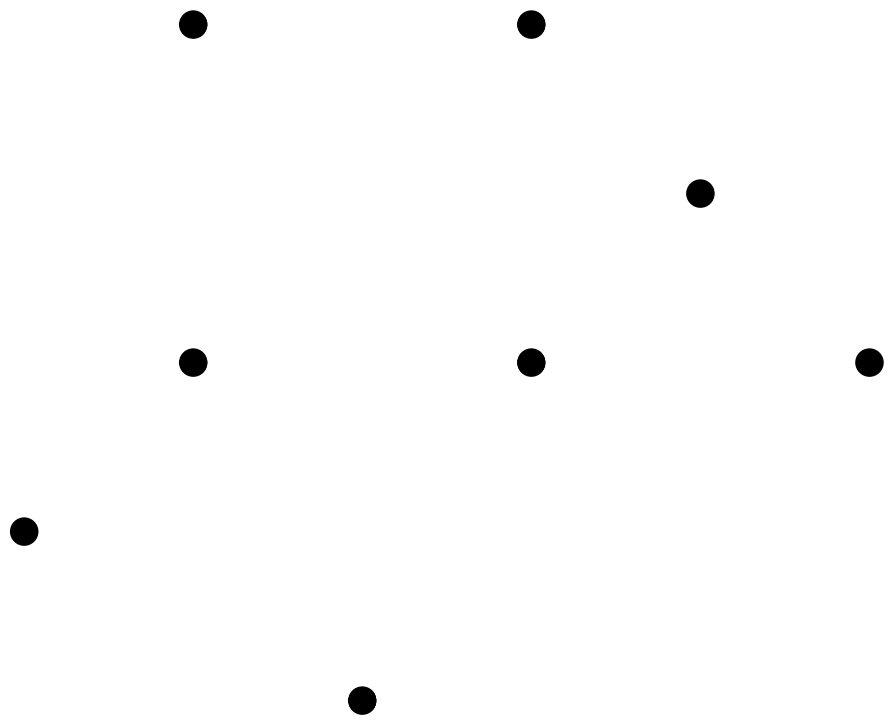


Algorithm

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

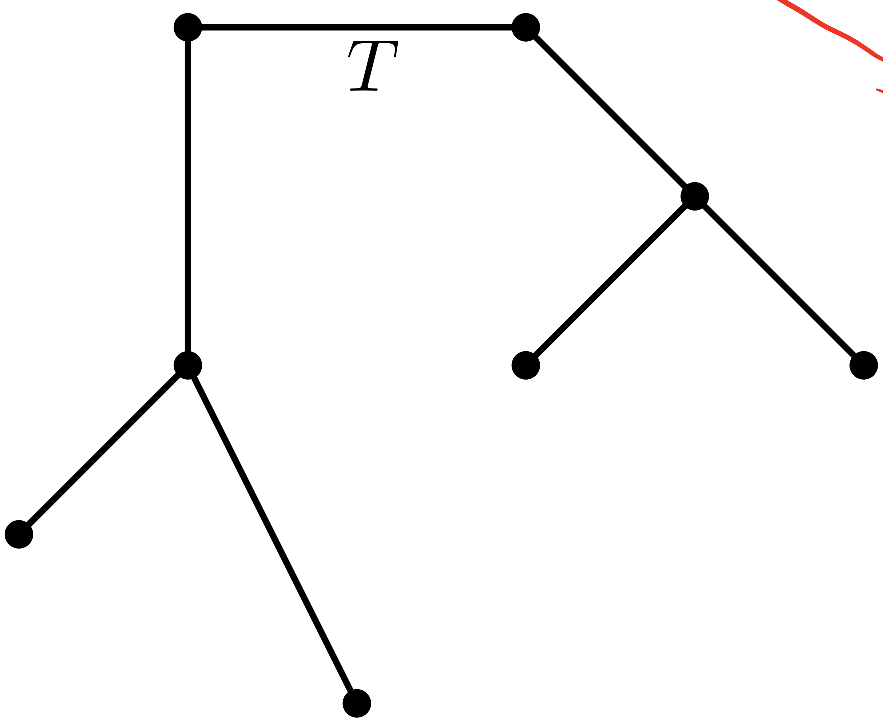
APPROX-TSP(G, c)
    Find MST T
    Make Euler tour W using each edge of T twice
    Shortcut W to H by skipping duplicates
    Return H



Algorithm

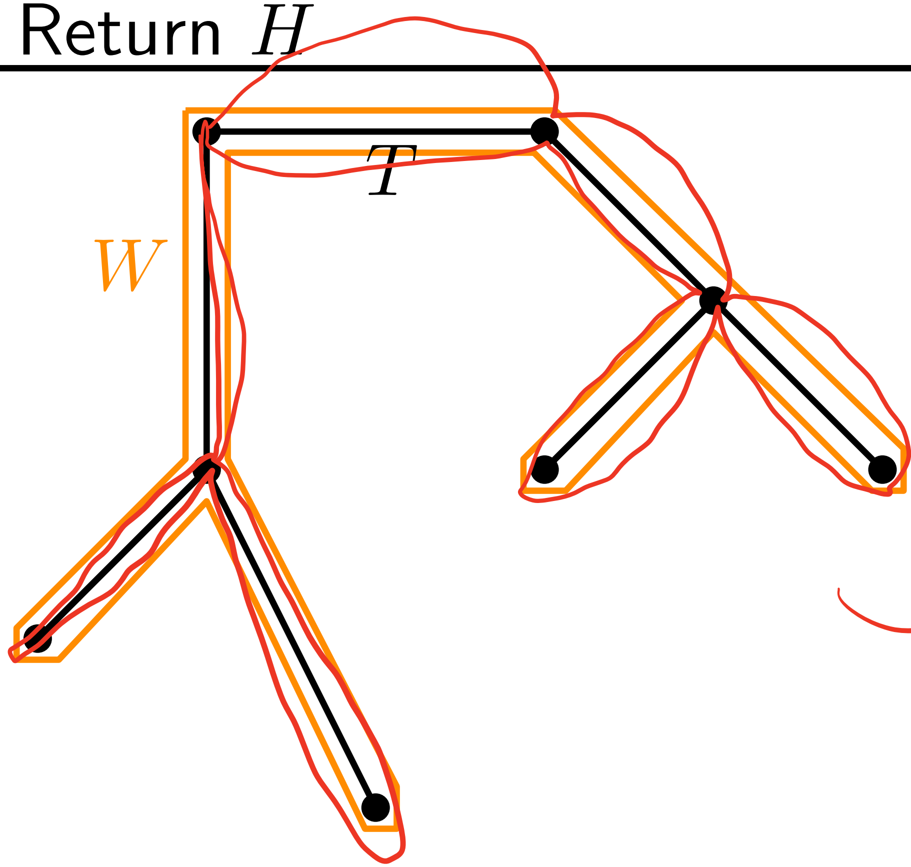
APPROX-TSP(G, c)
Find MST T
Make Euler tour W using each edge of T twice
Shortcut W to H by skipping duplicates
Return H

Minimum Spanning Tree



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

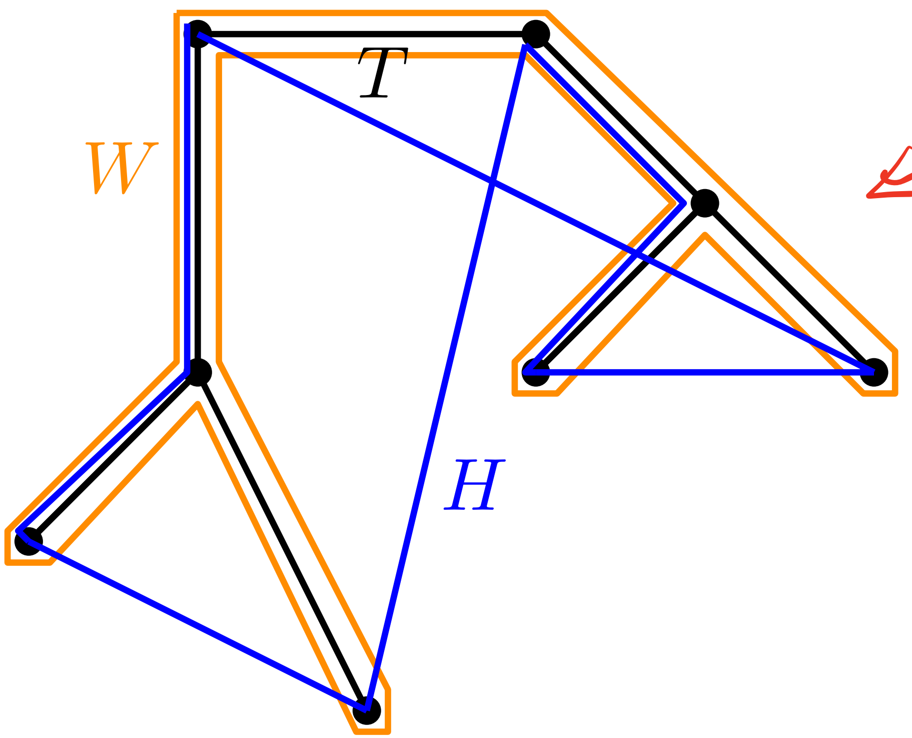


Remove

Repeated things

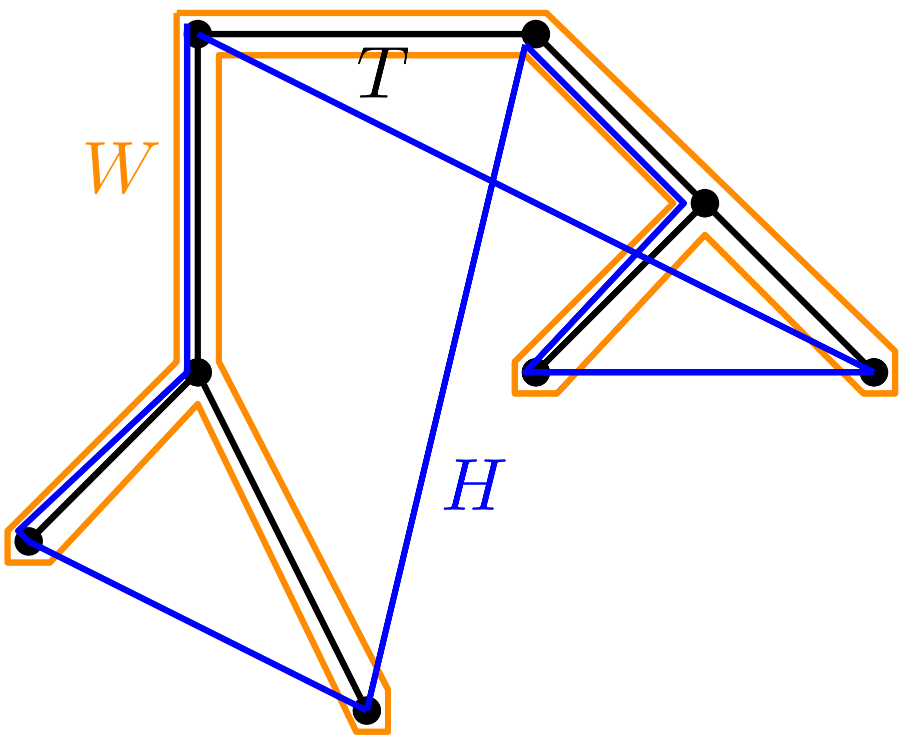
Algorithm

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

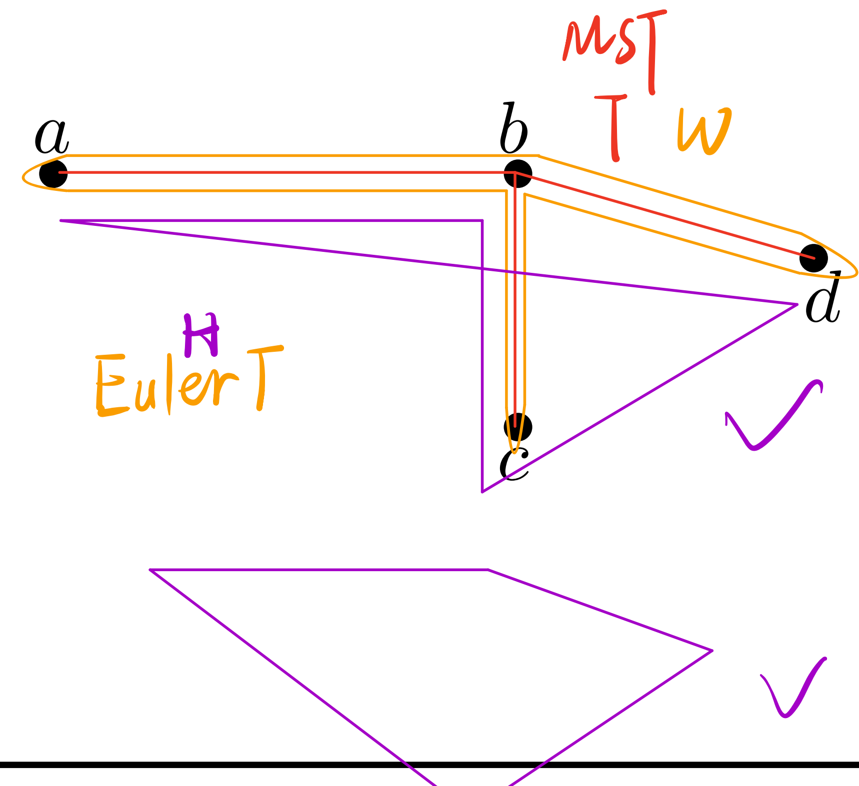


Algorithm

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

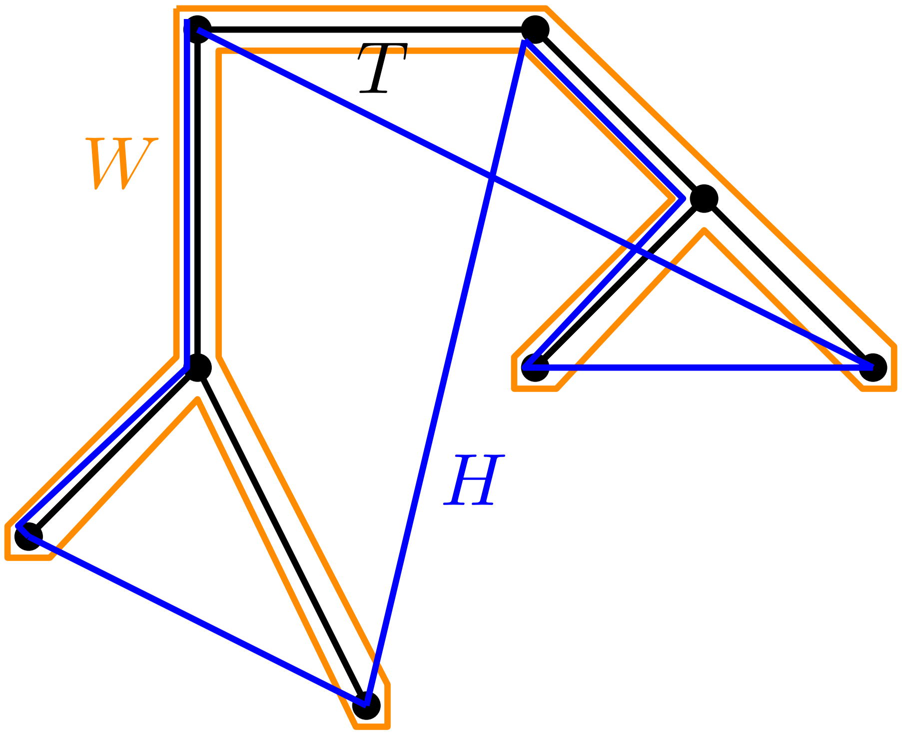


Exercise: Run the algorithm on this instance.



Theorem

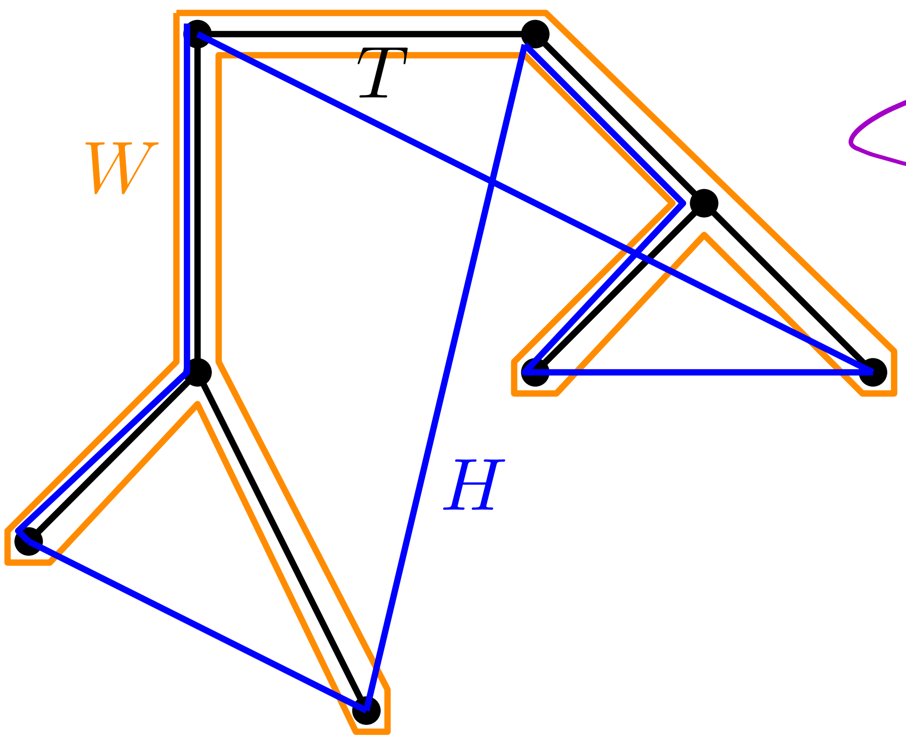
APPROX-TSP(G, c)
  Find MST T
  Make Euler tour W using each edge of T twice
  Shortcut W to H by skipping duplicates
  Return H



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



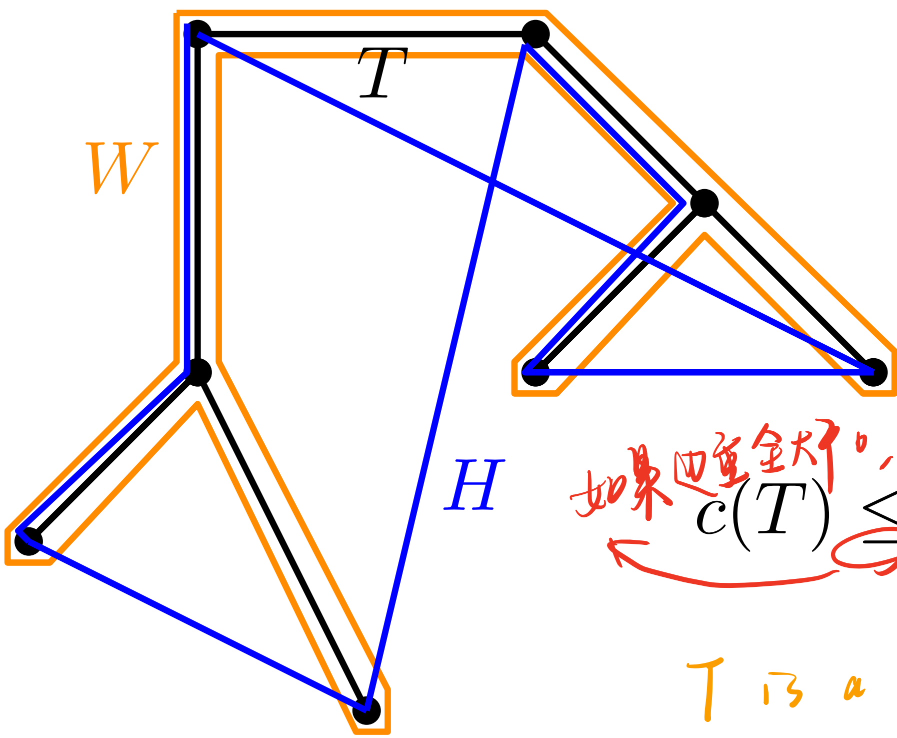
Thm.: APPROX-TSP is a poly-time 2-approx. alg.

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.

如果边权全大于0，那么不可能

$c(T) \leq c(H^*)$ → The cost of our tree is at most the cost of the optimal TSP.

T is a set of edges with smallest total cost that connect all vertices.

H^*: the smallest weight cycle, a way to connect all the vertices.

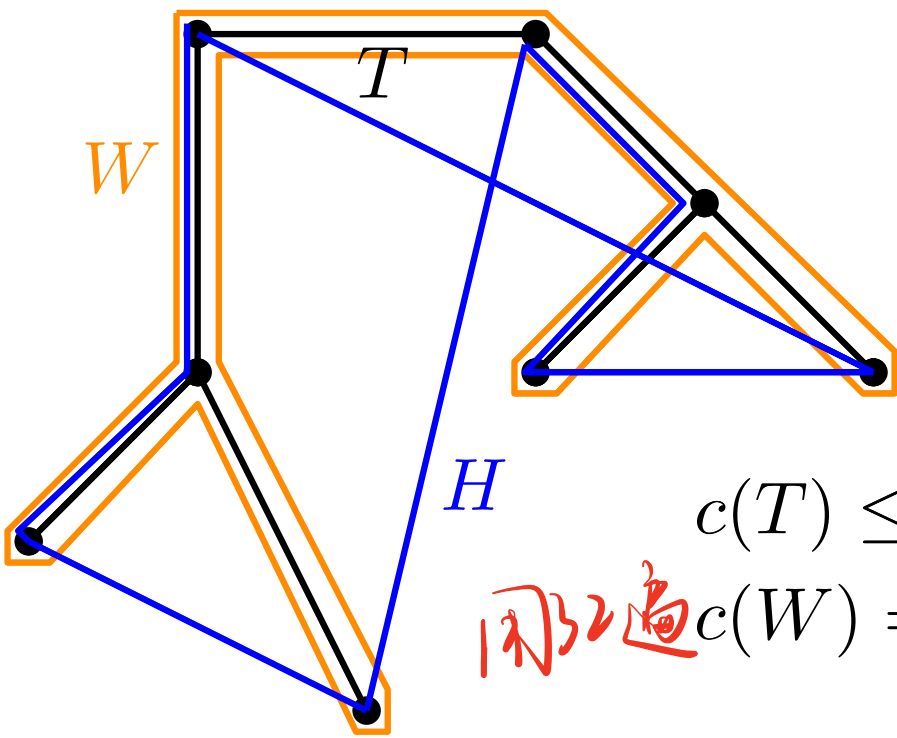
连接所有顶点，总成本最小的一组边.

a way

如果你拿走H*里的一条边，剩下的也是一个树，但那个树比MST贵。

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?

we can compute the minimum spanning tree with Primus Algorithms which is very efficient and then we can make the euler that just need to repeat these edges while travesing the tree. When we compute W, we just need to keep track of which vertices have we been on before and then skip those. ⇒ Polynomial Time.

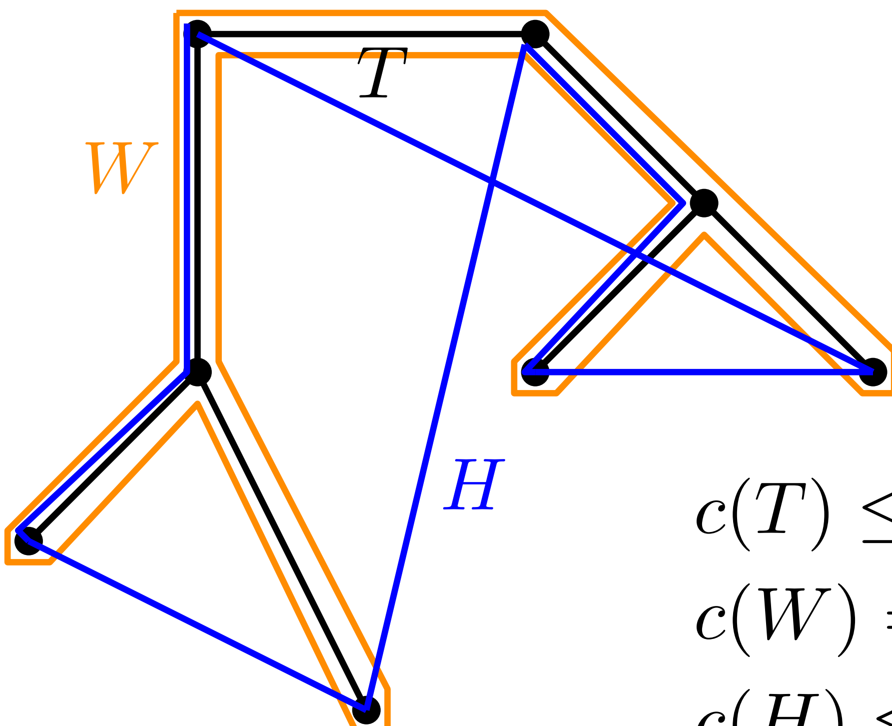
Let H^* be an opt. sol.

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

用了2遍 $c(W) = 2c(T)$

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.

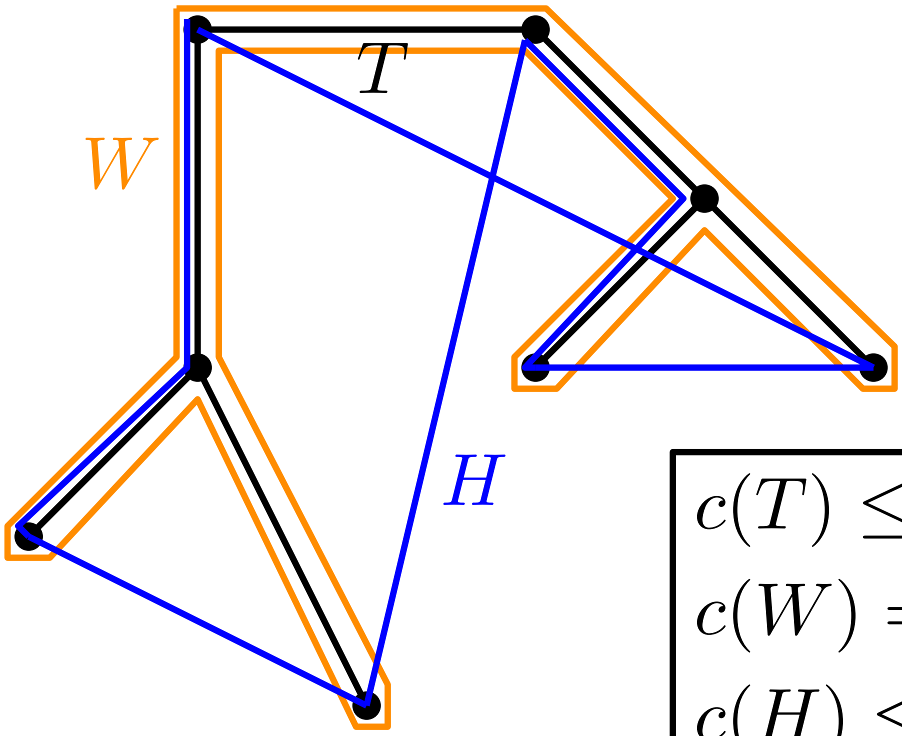
$c(T) \leq c(H^*)$

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

$c(H) \leq c(W)$ → 三角不等式

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.

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

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 $c(H)/c(H^*) \le 2$ when we don't know H^*?

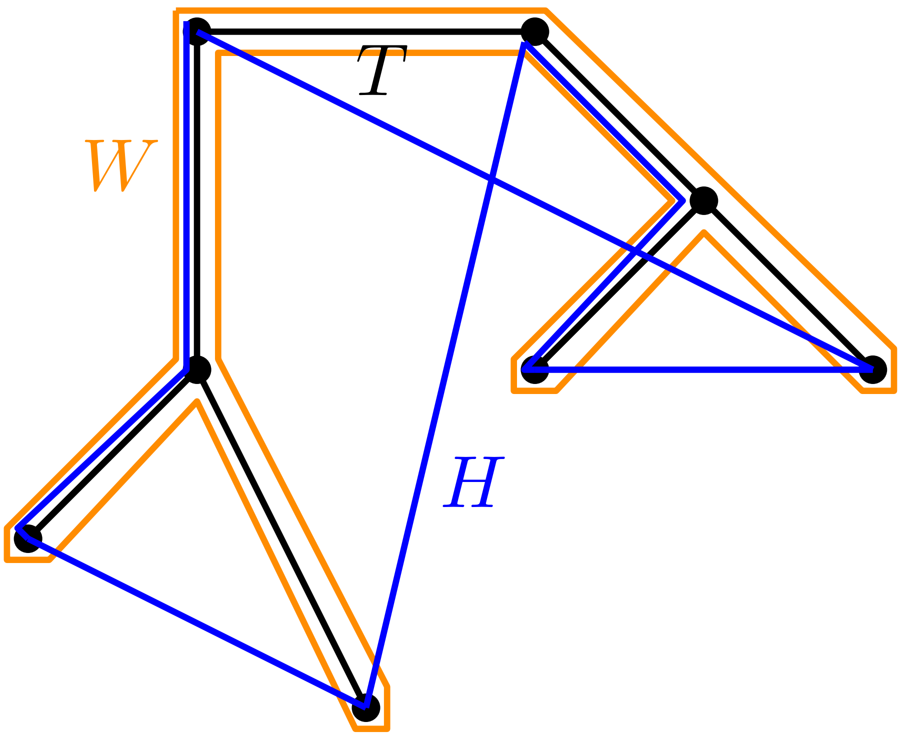
Answer: By proving $c(H) \le 2c(T)$ and $c(T) \le c(H^*)$.

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

For TSP: $\square = c(T)$ 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.

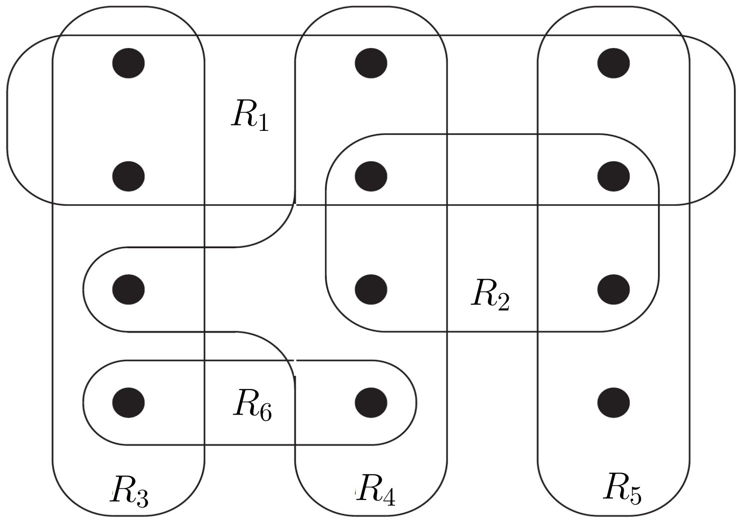
Set Cover

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

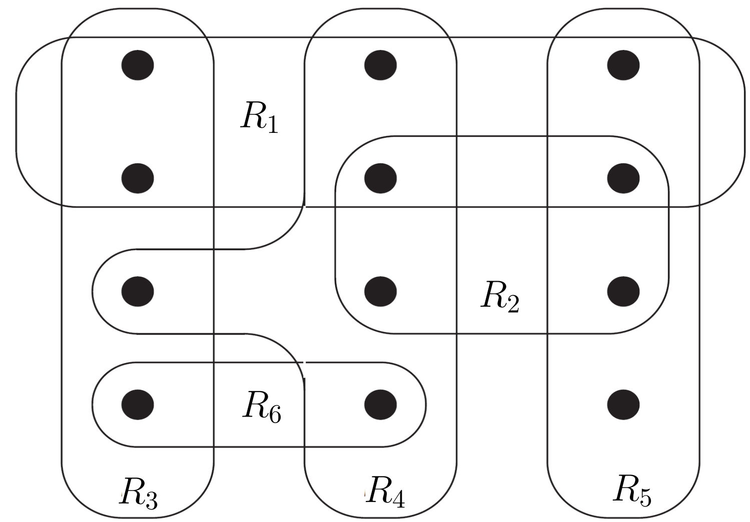
Input: Pair $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq \mathcal{P}(X)$ is a family of subsets of X.

Goal: Find $\mathcal{C} \subseteq \mathcal{F}$ covering X, i.e., $\bigcup_{S \in \mathcal{C}} S = X$, with $|\mathcal{C}|$ minimum.

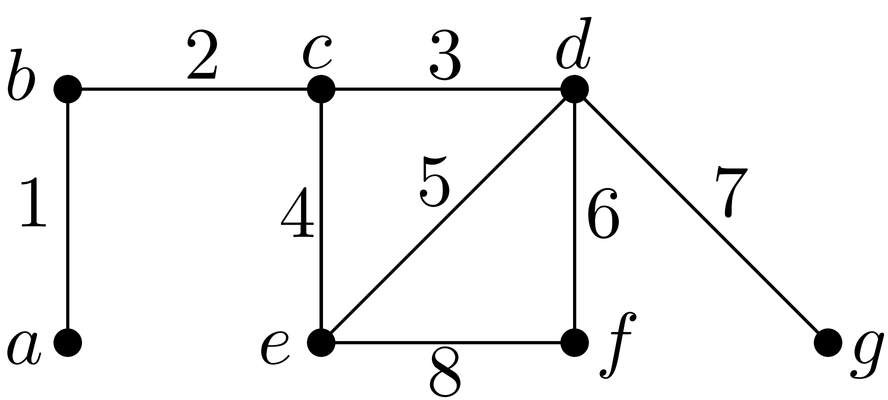


Set Cover

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.

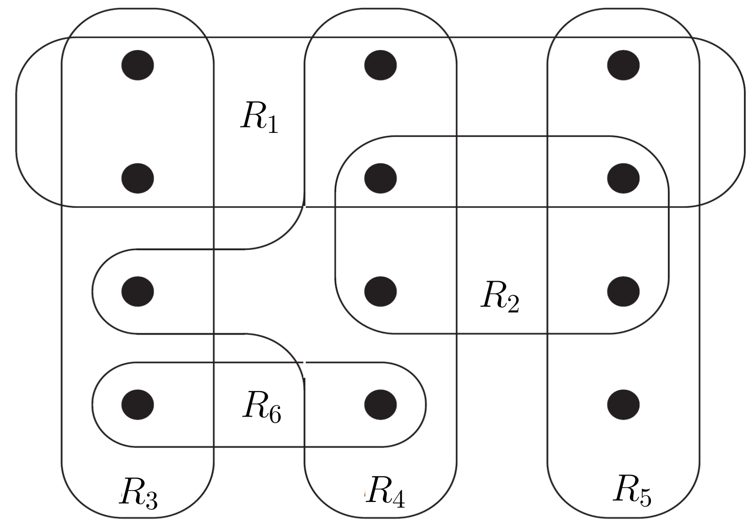


Exercise: Show that vertex cover is a special case.



Set Cover

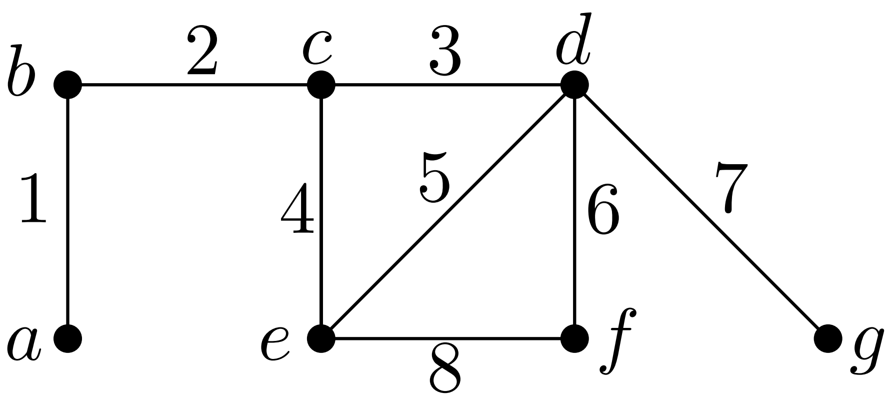
Input: Pair $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq \mathcal{P}(X)$ is a family of subsets of X.
Goal: Find $\mathcal{C} \subseteq \mathcal{F}$ covering X, i.e., $\bigcup_{S \in \mathcal{C}} S = X$, with $|\mathcal{C}|$ minimum.



Exercise: Show that vertex cover is a special case.

$X := \{1, 2, \ldots, 8\}$ = E

$\mathcal{F} := \{\overset{a}{\{1\}}, \overset{b}{\{1, 2\}}, \overset{c}{\{2, 3, 4\}}$, 点相连的边
$\overset{d}{\{3, 5, 6\}}, \overset{e}{\{4, 5, 8\}}, \overset{f}{\{6, 8\}}, \overset{g}{\{7\}}\}$

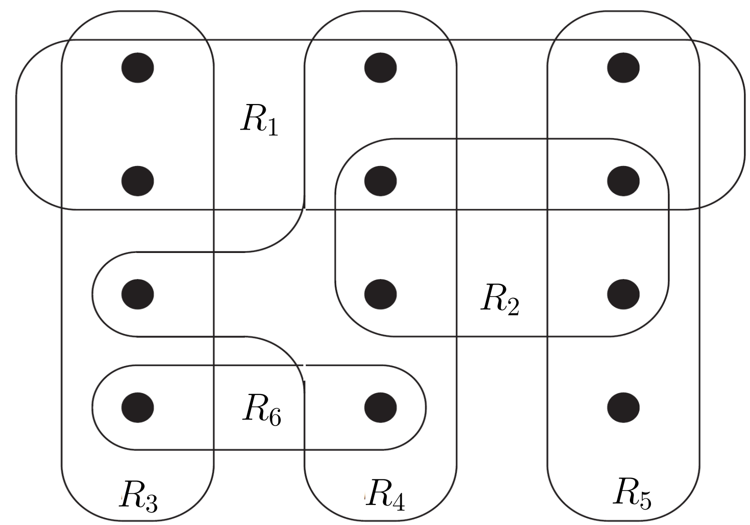


Set Cover

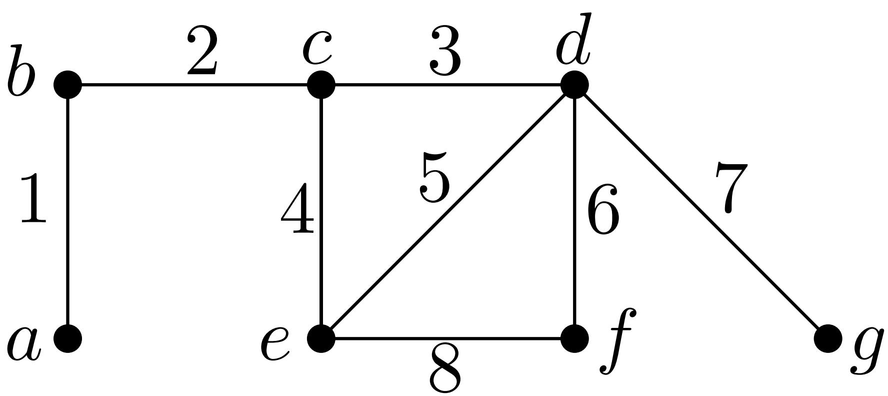
Input: Pair $(X, \mathcal{F})$, where X is a finite set and $\mathcal{F} \subseteq \mathcal{P}(X)$ is a family of subsets of X.

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.



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



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

Set cover is more general

Greedy Algorithm

贪心算法

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

$i := 0$

while $X \setminus S_{<i+1} \neq \emptyset$ → we have not covered all the elements

$\quad i := i + 1$

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

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

alredy covered.

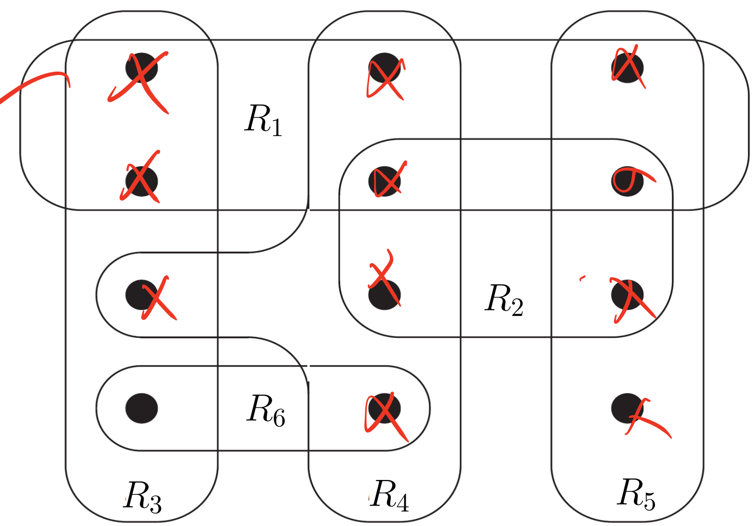
Here, $S_{<i} := \bigcup_{j=1}^{i-1} S_j$.

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.



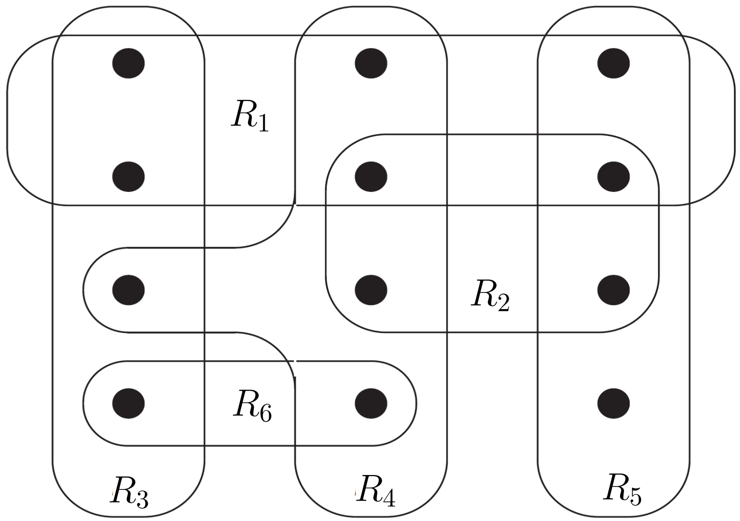
S_1, S_2, S_3, S_4
$\Downarrow$ $\Downarrow$ $\Downarrow$ $\Downarrow$
R_1 R_4 R_5 R_6或R_3

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 $|S_i \setminus S_{<i}|$

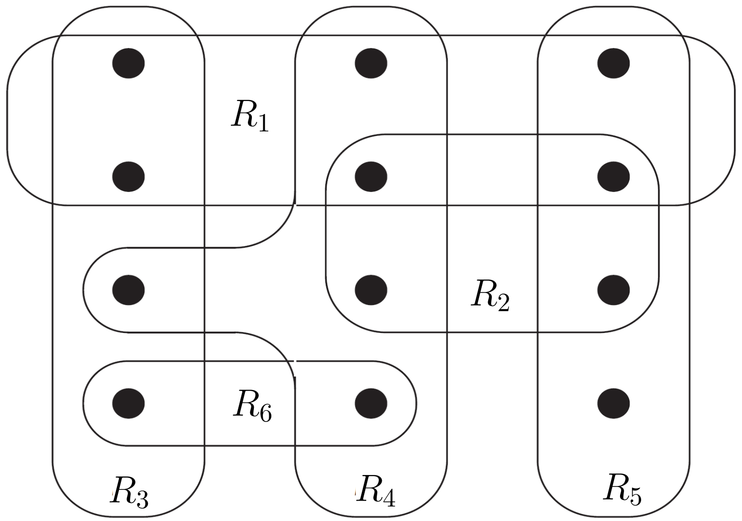
Return $\mathcal{C} := \{S_1, \ldots, S_i\}$

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

Exercise: Run the algorithm on this instance.

$S_1 := R_1$

$S_2 := R_4$



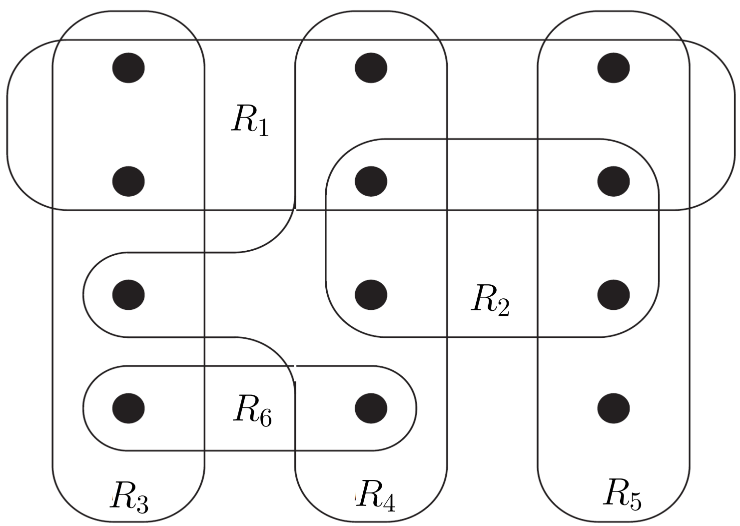
Greedy Algorithm

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.

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



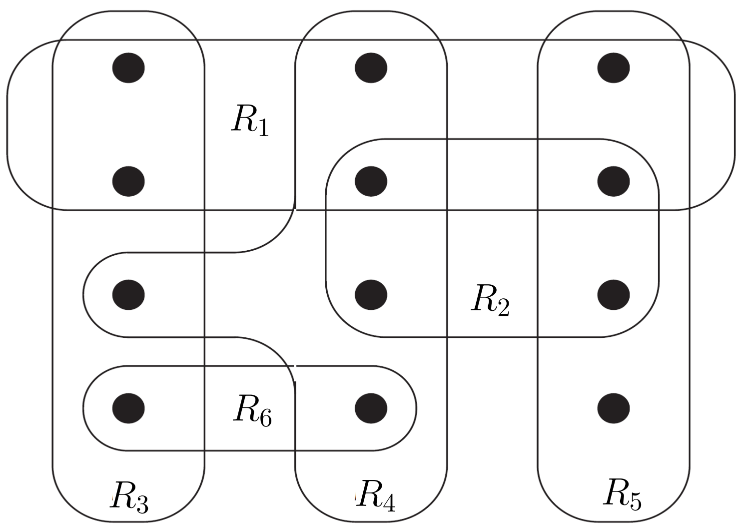
Greedy Algorithm

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

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

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

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

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

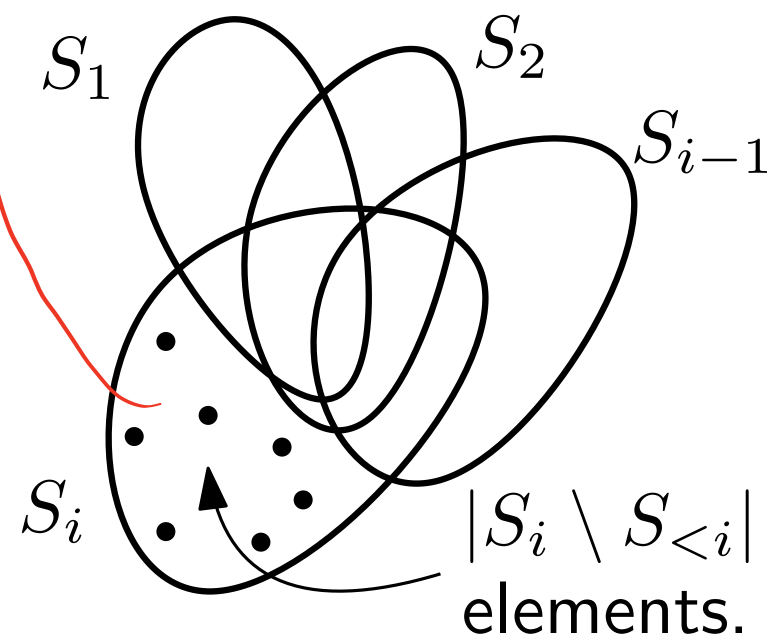
Theorem

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

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($X, \mathcal{F}$)
  $i := 0$
  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 $\mathcal{C} := \{S_1, \dots, S_i\}$



Theorem

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

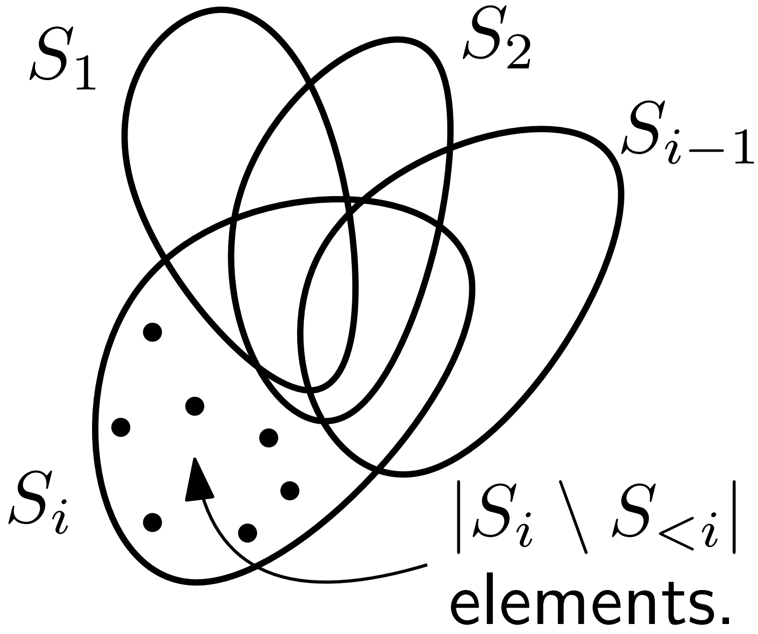
For $x \in S_i \setminus S_{<i}$, define $c_x := \frac{1}{|S_i \setminus S_{<i}|}$.

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

```
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 $c(Y) := \sum_{x \in Y} c_x$.

改写

Observation:

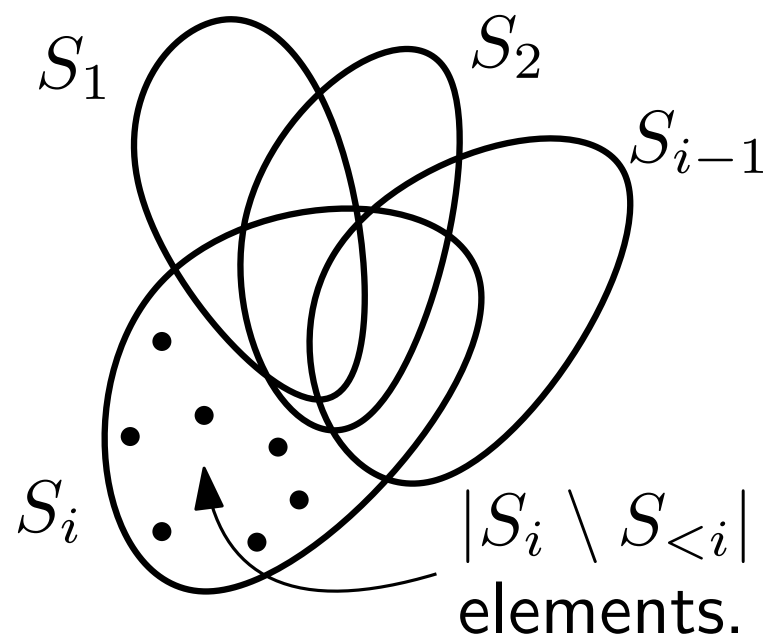
when the algorithm stops

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

total cost

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

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



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

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

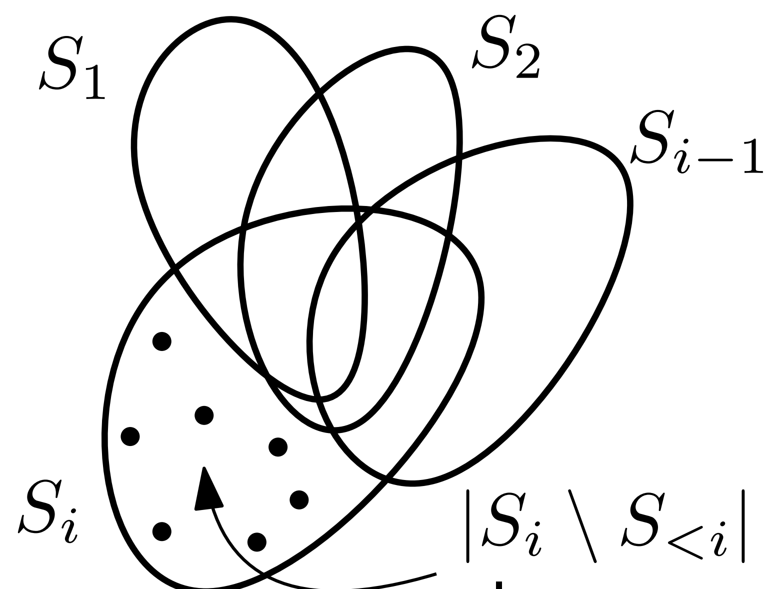
Observation:

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

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

arbitrary

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



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 $S \in \mathcal{F}$:

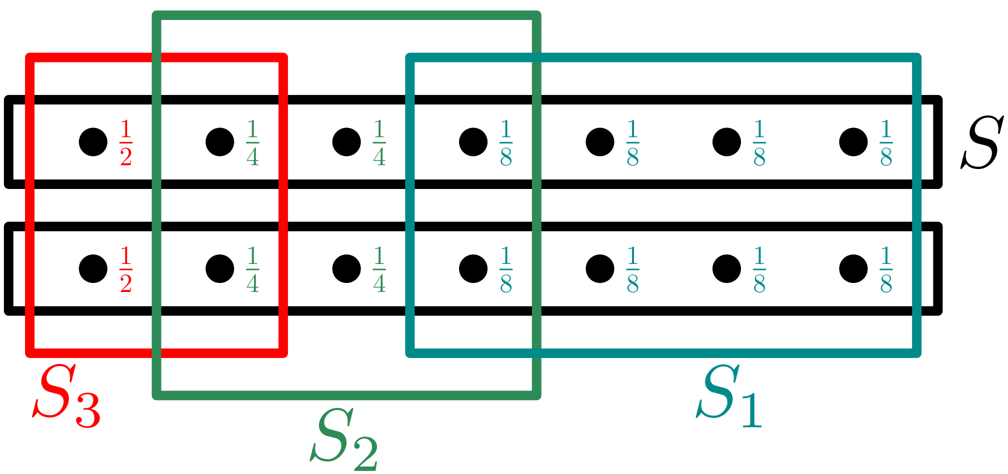
$$c(S) \le \sum_{i=1}^{|S|} \frac{1}{i} = H_{|S|}.$$

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 $S \in \mathcal{F}$:

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

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 $x_j, x_{j-1}, \ldots, x_1$)

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

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

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

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

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

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