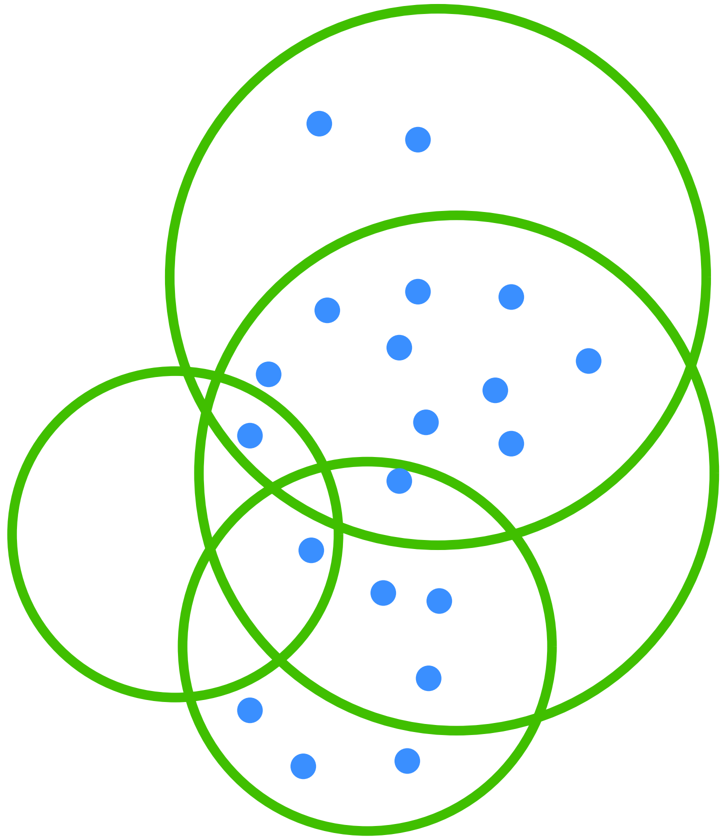
Geometric Set Cover

sampling with reweighting

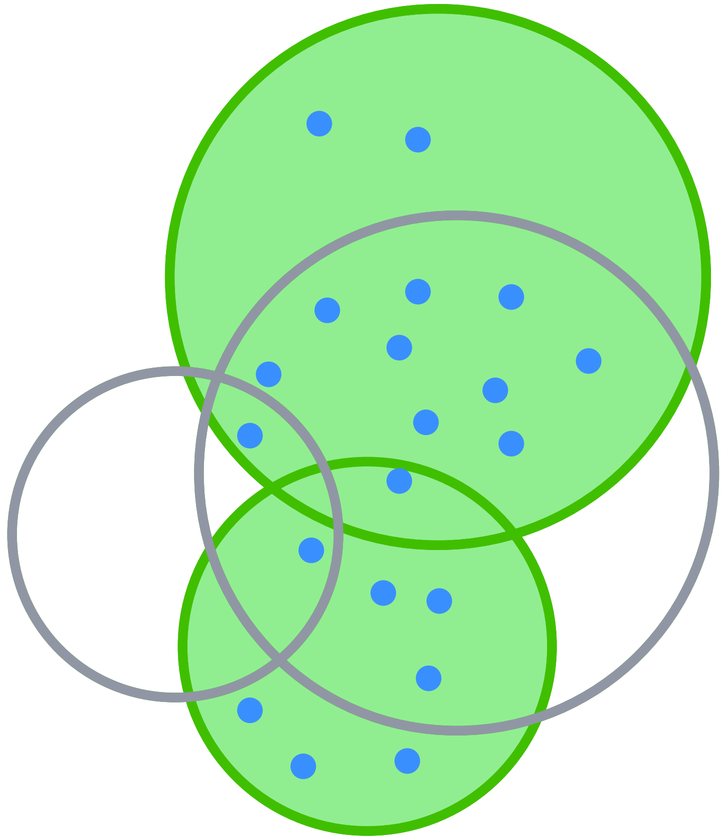
Example: Covering points with disk

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering the points.



Example: Covering points with disk

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering the points.

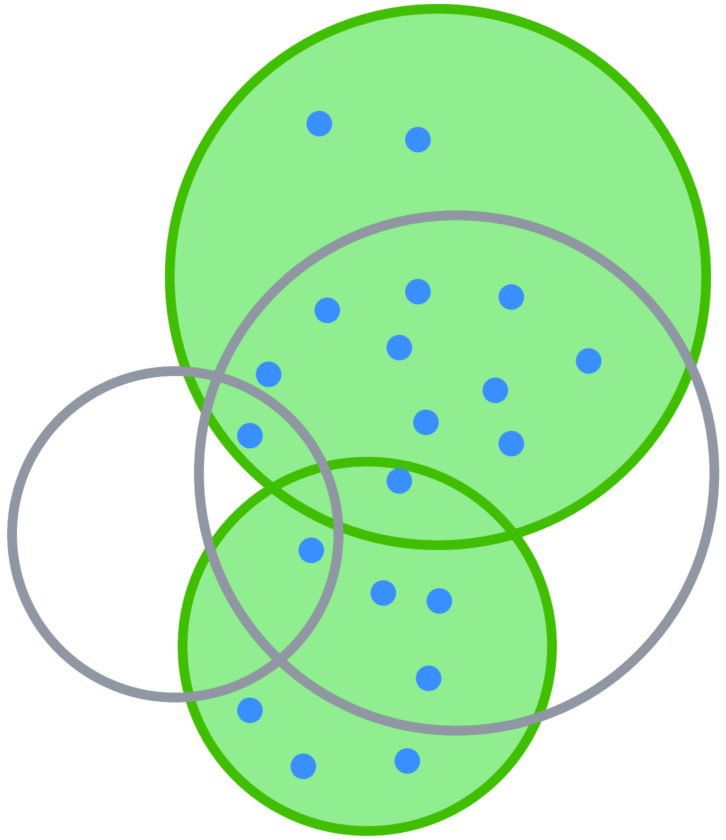


Example: Covering points with disk

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering the points.

range space:

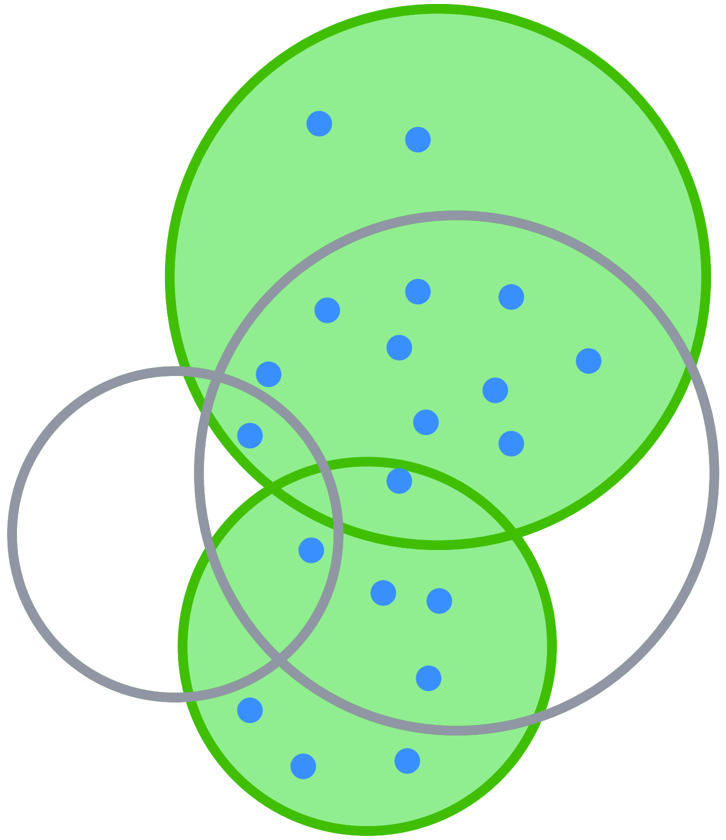
?



Example: Covering points with disk

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering the points.

range space:
$(P, \mathcal{D}_{|P})$

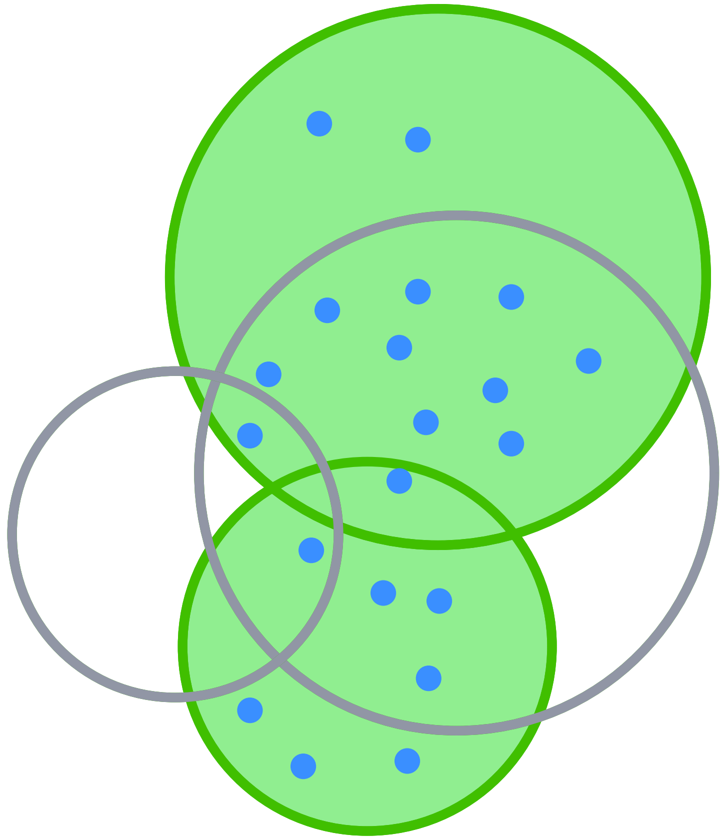


Example: Covering points with disk

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering the points.

range space:
$(P, \mathcal{D}_{|P})$

where $\mathcal{D}_{|P} := \{P \cap d \mid d \in \mathcal{D}\}$



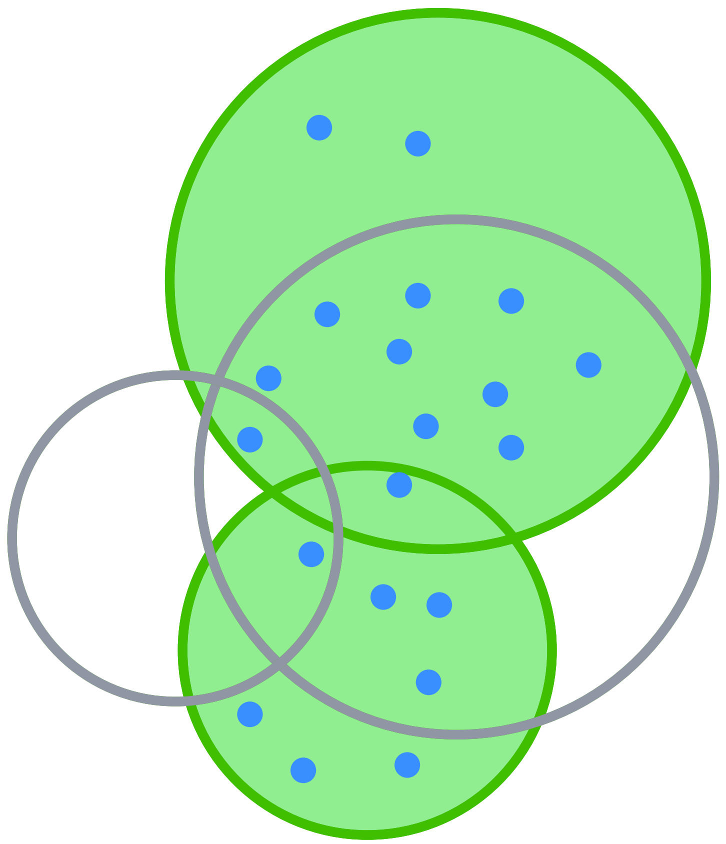
Example: Covering points with disk

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering the points.

range space:
$(P, \mathcal{D}_{|P})$

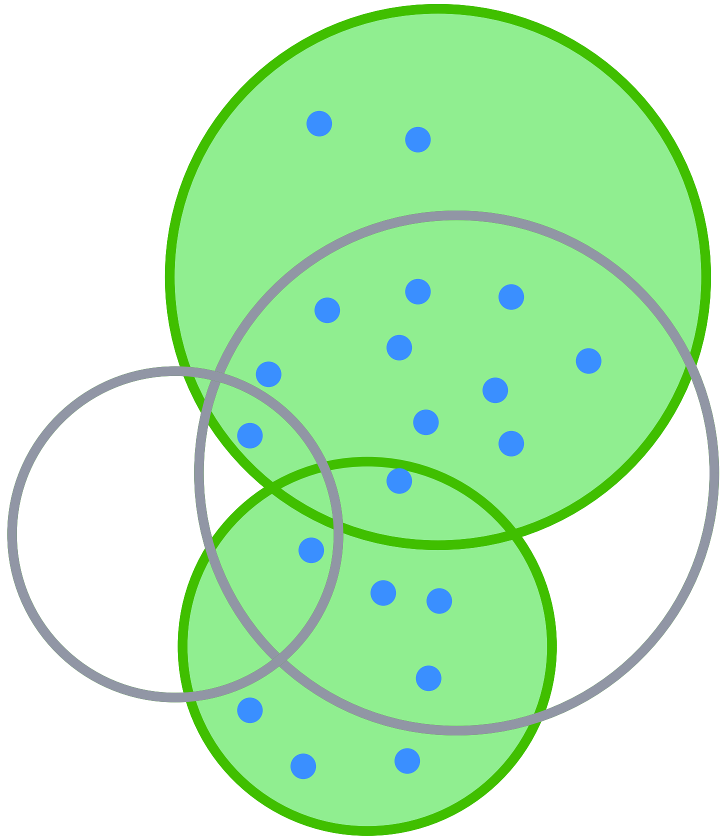
where $\mathcal{D}_{|P} := \{P \cap d \mid d \in \mathcal{D}\}$

minimum set cover:
smallest $D \subset \mathcal{D}_{|P}$ such that $\bigcup_{d \in D} d = P$



Example: Covering points with disk

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering the points.



range space:
$(P, \mathcal{D}_{|P})$

where $\mathcal{D}_{|P} := \{P \cap d \mid d \in \mathcal{D}\}$

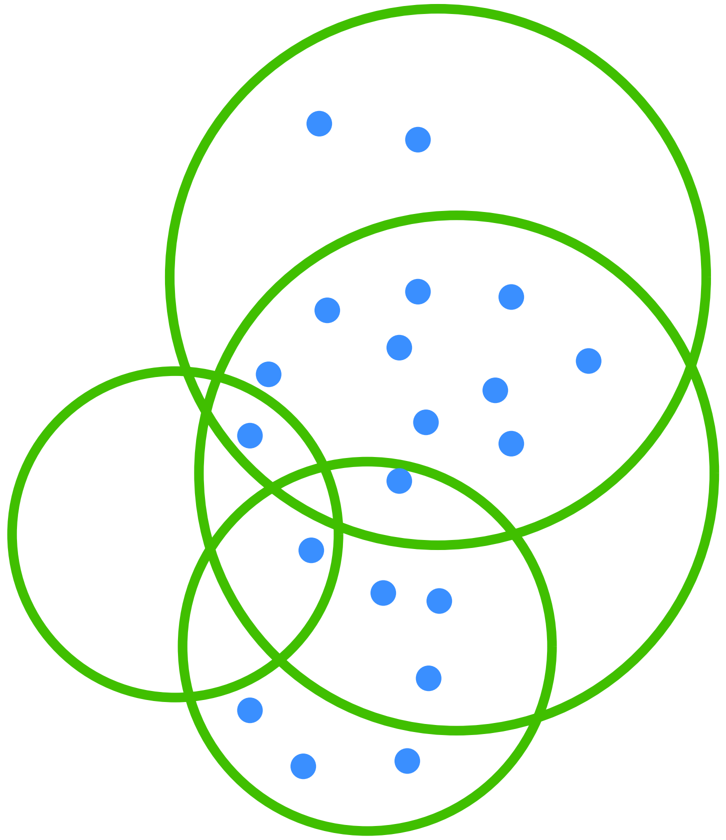
minimum set cover:
smallest $D \subset \mathcal{D}_{|P}$ such that $\bigcup_{d \in D} d = P$

Question: What do you know about the set cover problem?

Greedy Approximation

Greedy Algorithm:

While $\exists$ uncovered points, select range that contains the most uncovered points



Greedy Approximation

Greedy Algorithm:

While $\exists$ uncovered points, select range that contains the most uncovered points

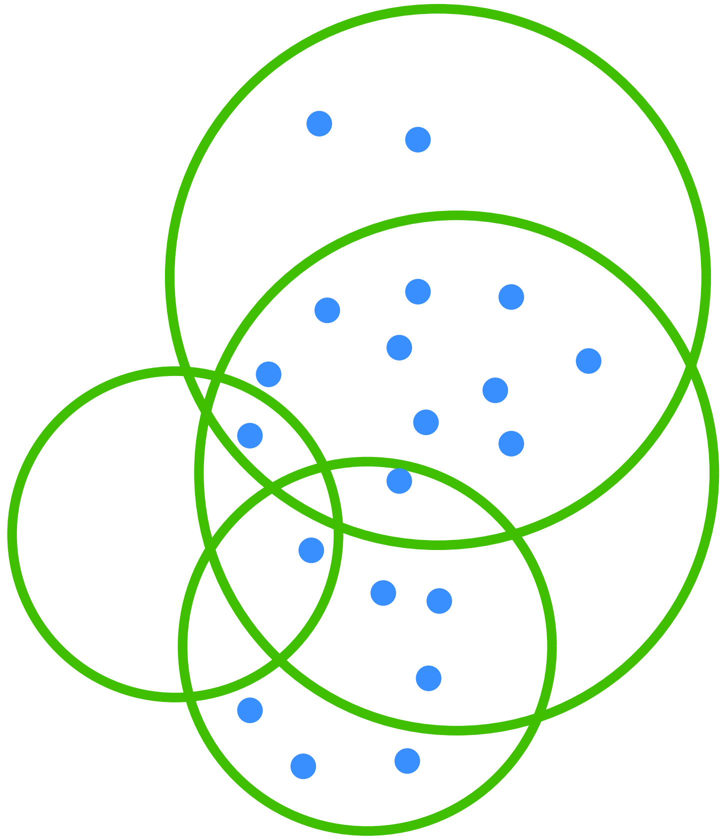
Quiz

What is the size of the greedy solution?

A 2

B 3

C 4



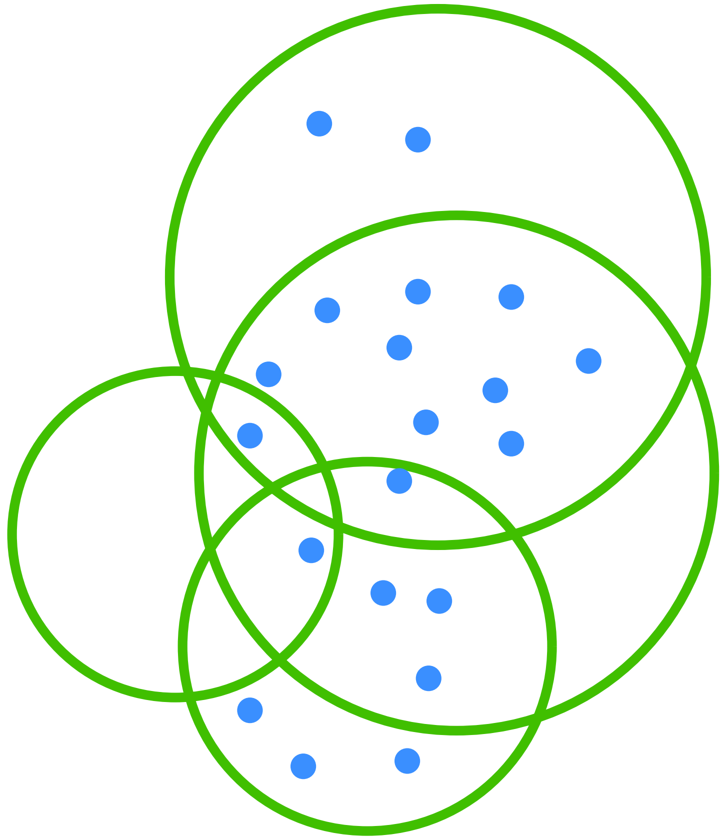
Greedy Approximation

Greedy Algorithm:

While ∃ uncovered points, select range that contains the most uncovered points

Quiz

What is the size of the greedy solution?



A 2

$\boxed{\text{B} \quad 3}$

C 4

Greedy Approximation

Greedy Algorithm:

While $\exists$ uncovered points, select range that contains the most uncovered points

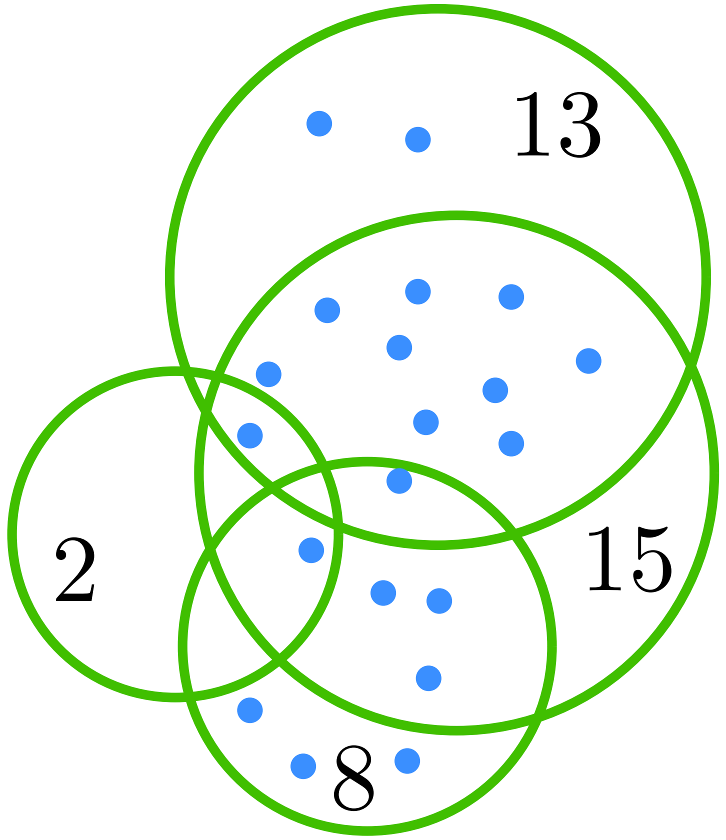
Quiz

What is the size of the greedy solution?

A 2

B 3

C 4



Greedy Approximation

Greedy Algorithm:

While $\exists$ uncovered points, select range that contains the most uncovered points

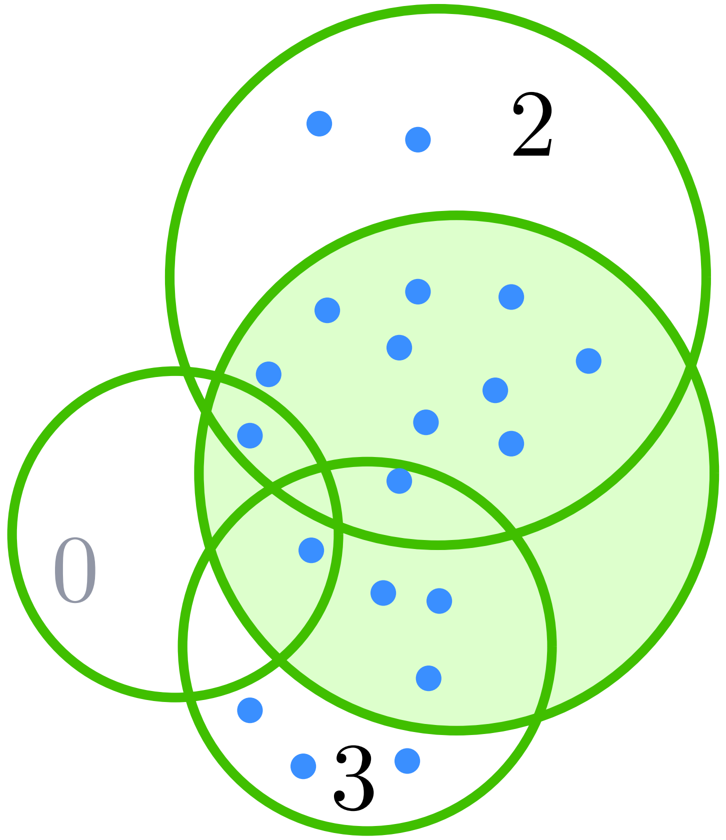
Quiz

What is the size of the greedy solution?

A 2

$\boxed{\text{B} \quad 3}$

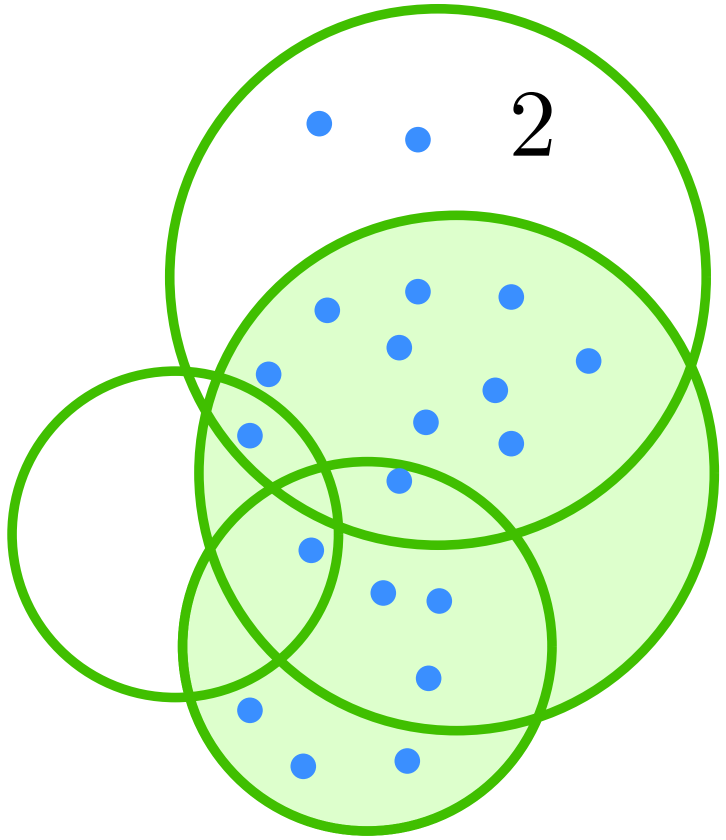
C 4



Greedy Approximation

Greedy Algorithm:

While $\exists$ uncovered points, select range that contains the most uncovered points



Quiz

What is the size of the greedy solution?

A 2

B 3

C 4

Greedy Approximation

Greedy Algorithm:

While ∃ uncovered points, select range that contains the most uncovered points

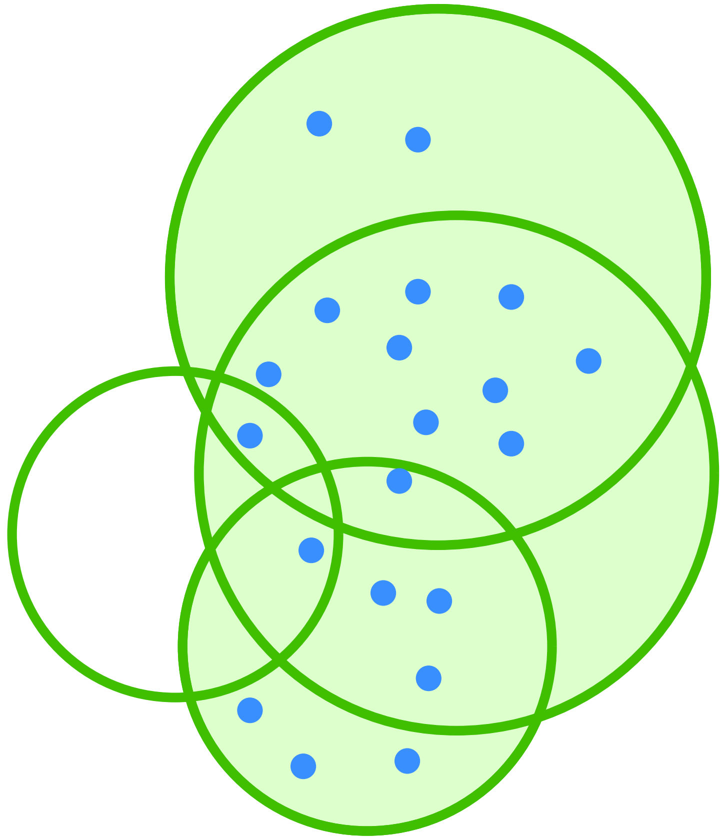
Quiz

What is the size of the greedy solution?

A 2

B 3

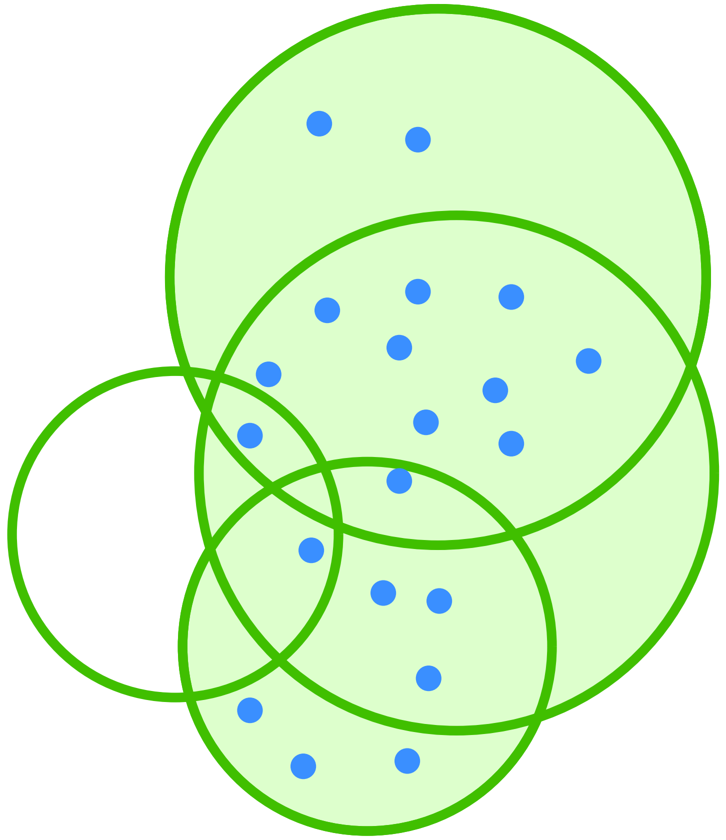
C 4



Greedy Approximation

size of greedy solution: $O(k \log n)$,

n: number of points, k: size of optimal solution

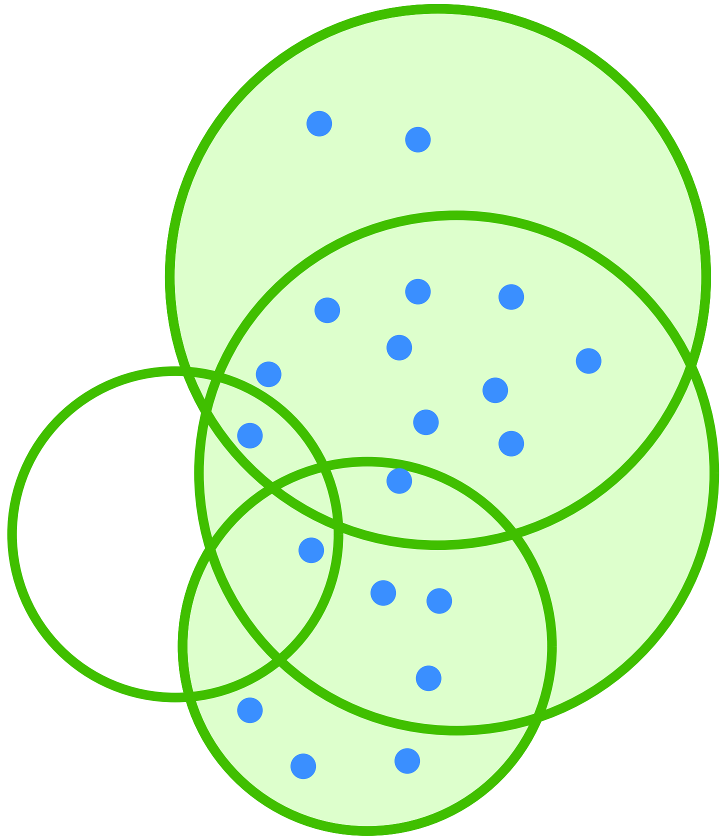


Greedy Approximation

size of greedy solution: $O(k \log n)$,

n: number of points, k: size of optimal solution

proof sketch



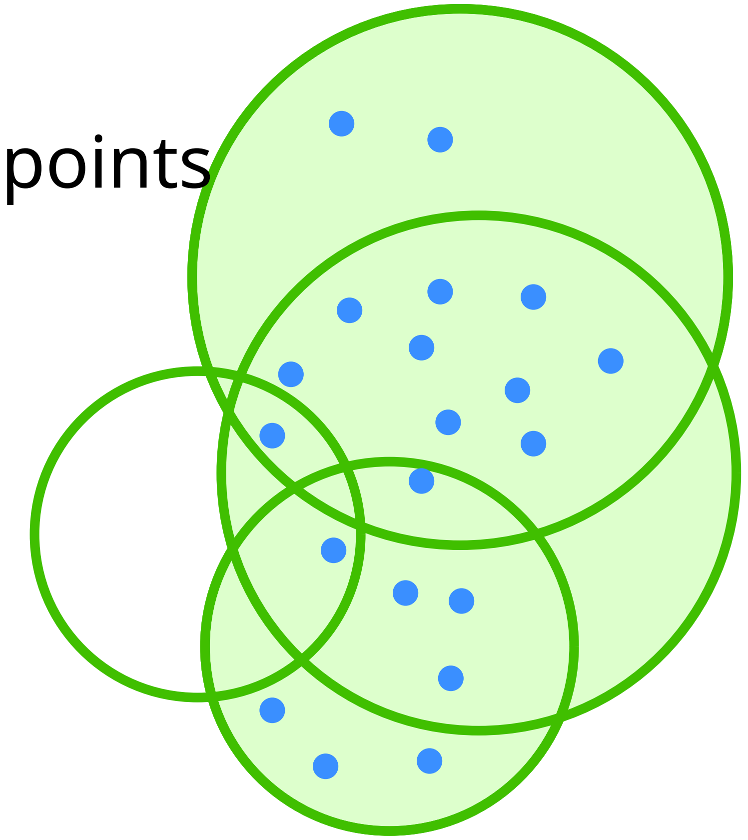
Greedy Approximation

size of greedy solution: $O(k \log n)$,

n: number of points, k: size of optimal solution

proof sketch

pigeonhole principle: $\exists$ range in optimal solution with $\geq n/k$ points



Greedy Approximation

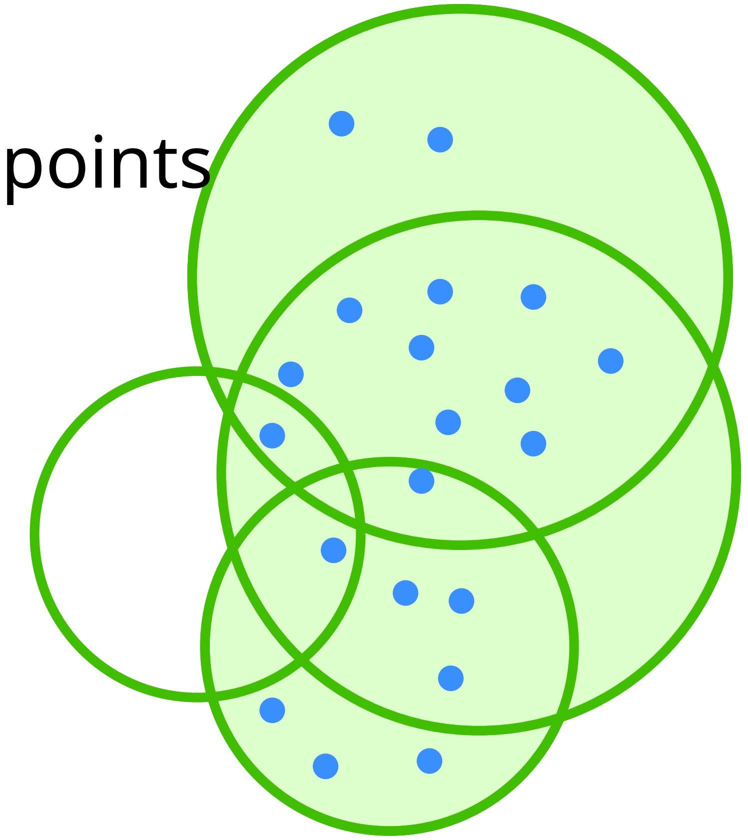
size of greedy solution: $O(k \log n)$,

n: number of points, k: size of optimal solution

proof sketch

pigeonhole principle: $\exists$ range in optimal solution with $\geq n/k$ points

$\Rightarrow$ first range in greedy solution contains $\geq n/k$ points



Greedy Approximation

size of greedy solution: $O(k \log n)$,

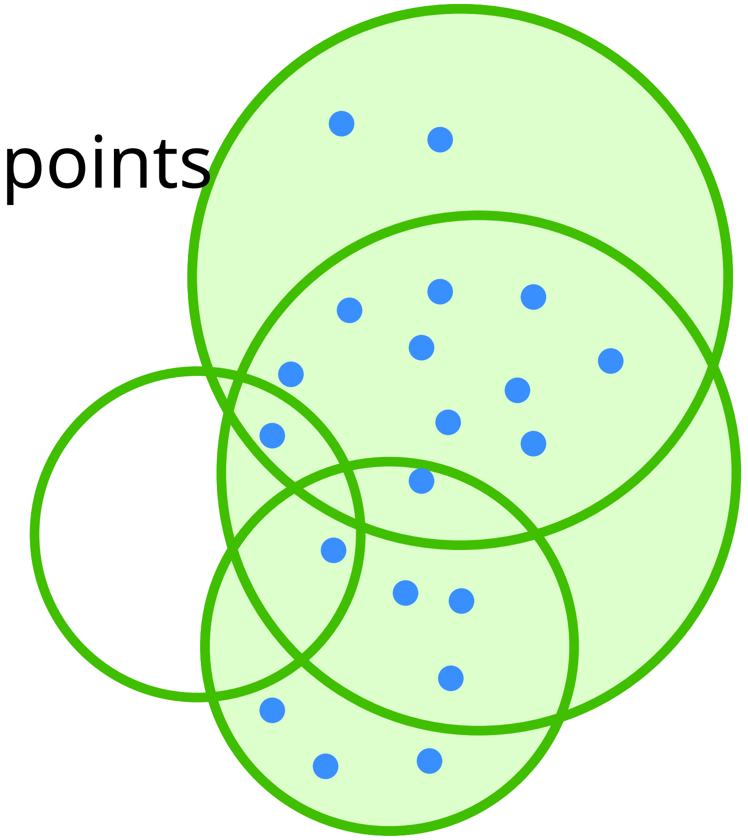
n: number of points, k: size of optimal solution

proof sketch

pigeonhole principle: $\exists$ range in optimal solution with $\geq n/k$ points

$\Rightarrow$ first range in greedy solution contains $\geq n/k$ points

$\Rightarrow \leq n(1 - \frac{1}{k})$ points remain uncovered



Greedy Approximation

size of greedy solution: $O(k \log n)$,

n: number of points, k: size of optimal solution

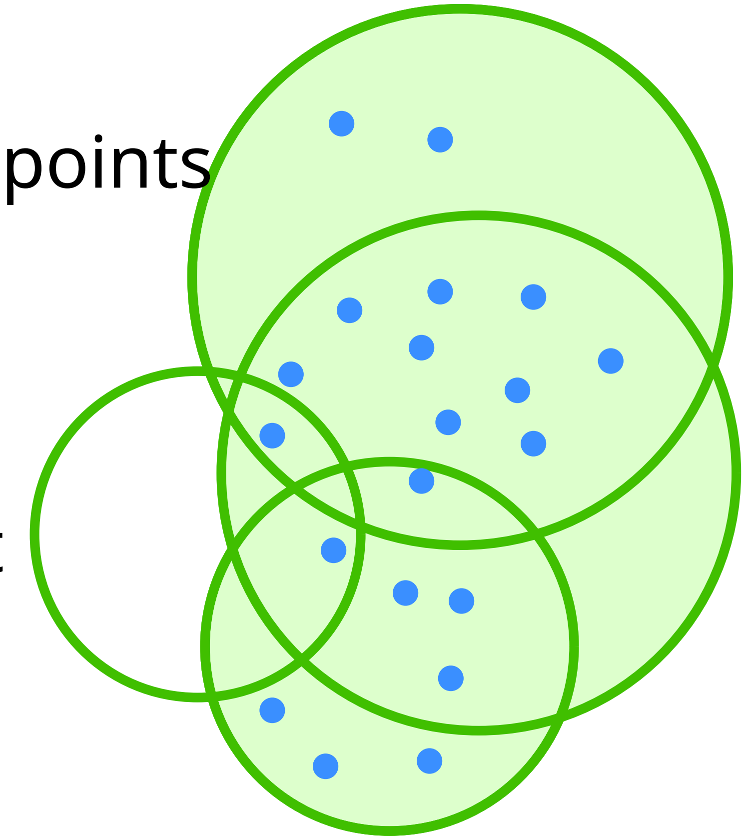
proof sketch

pigeonhole principle: $\exists$ range in optimal solution with $\geq n/k$ points

$\Rightarrow$ first range in greedy solution contains $\geq n/k$ points

$\Rightarrow \leq n(1 - \frac{1}{k})$ points remain uncovered

iterate argument: after i steps greedy algorithm covers all but $\leq n(1 - \frac{1}{k})^i$



Greedy Approximation

size of greedy solution: $O(k \log n)$,

n: number of points, k: size of optimal solution

proof sketch

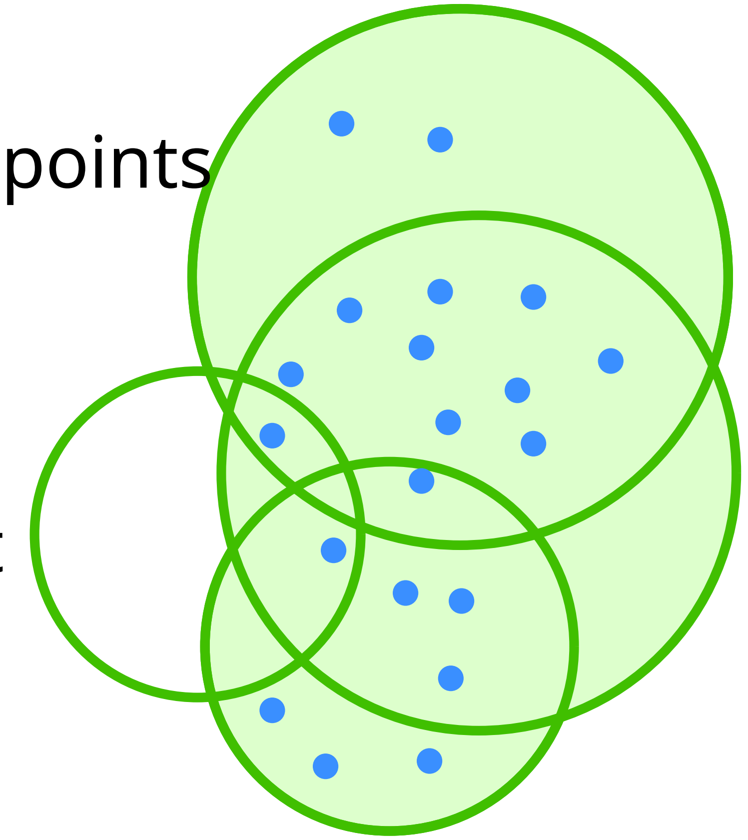
pigeonhole principle: $\exists$ range in optimal solution with $\geq n/k$ points

$\Rightarrow$ first range in greedy solution contains $\geq n/k$ points

$\Rightarrow \leq n(1 - \frac{1}{k})$ points remain uncovered

iterate argument: after i steps greedy algorithm covers all but $\leq n(1 - \frac{1}{k})^i$

all points covered when $n(1 - \frac{1}{k})^i < 1$

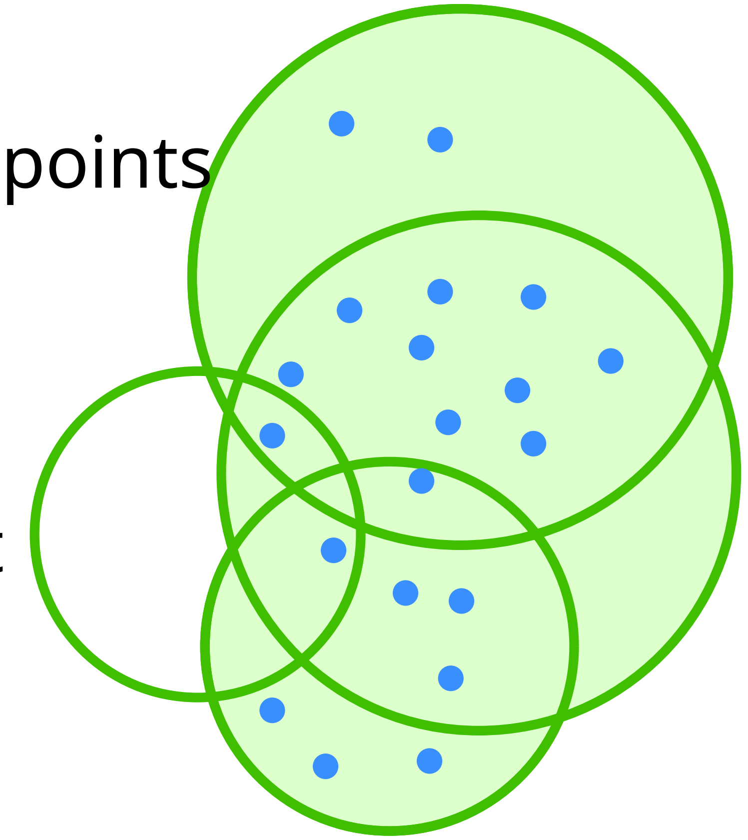


Greedy Approximation

size of greedy solution: $O(k \log n)$,

n: number of points, k: size of optimal solution

proof sketch



pigeonhole principle: $\exists$ range in optimal solution with $\geq n/k$ points

$\Rightarrow$ first range in greedy solution contains $\geq n/k$ points

$\Rightarrow \leq n(1-\frac{1}{k})$ points remain uncovered

iterate argument: after i steps greedy algorithm covers all but $\leq n(1-\frac{1}{k})^i$

all points covered when $n(1-\frac{1}{k})^i < 1$

note: $\left(1-\frac{1}{k}\right)^k < 1/e$

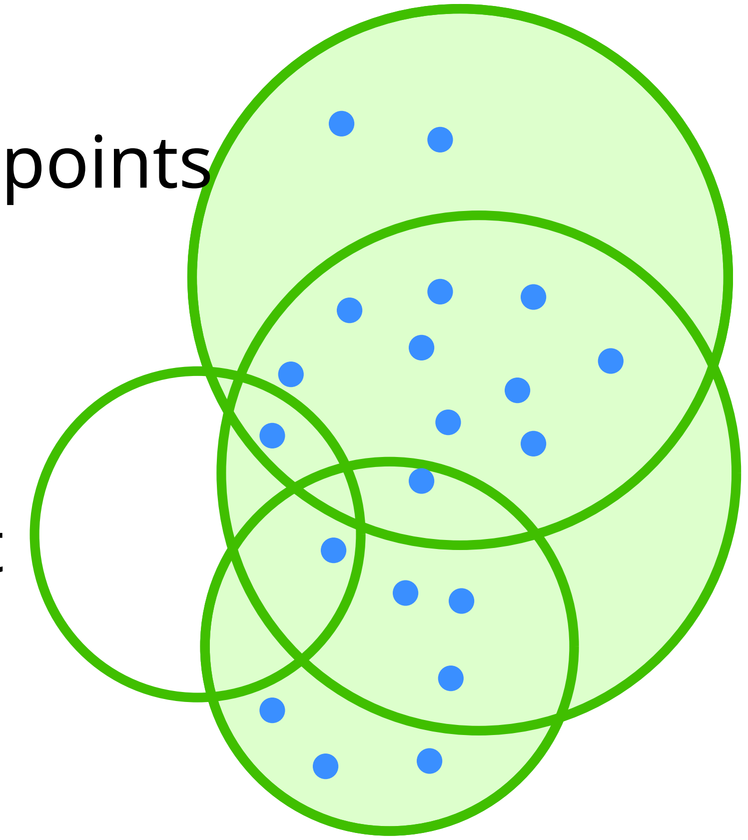
Greedy Approximation

size of greedy solution: $O(k \log n)$,

n: number of points, k: size of optimal solution

proof sketch

pigeonhole principle: $\exists$ range in optimal solution with $\geq n/k$ points



$\Rightarrow$ first range in greedy solution contains $\geq n/k$ points

$\Rightarrow \leq n(1 - \frac{1}{k})$ points remain uncovered

iterate argument: after i steps greedy algorithm covers all but $\leq n(1 - \frac{1}{k})^i$

all points covered when $n(1 - \frac{1}{k})^i < 1$

note: $\left(1 - \frac{1}{k}\right)^k < 1/e$

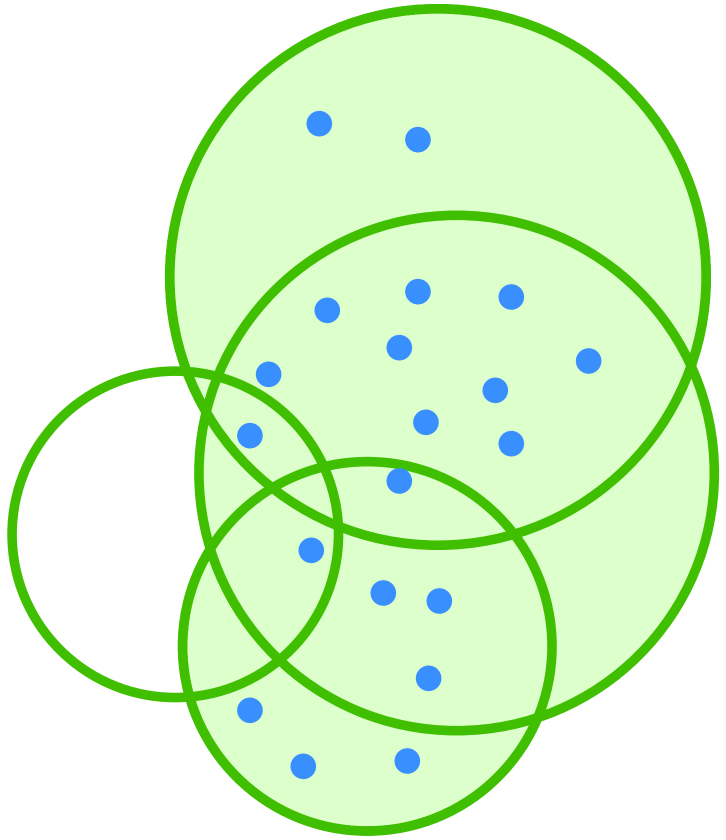
$\Rightarrow$ all points covered for $i = k \ln n = O(k \log n)$

Greedy Approximation

size of greedy solution: $O(k \log n)$,

n: number of points, k: size of optimal solution

Can we do better?

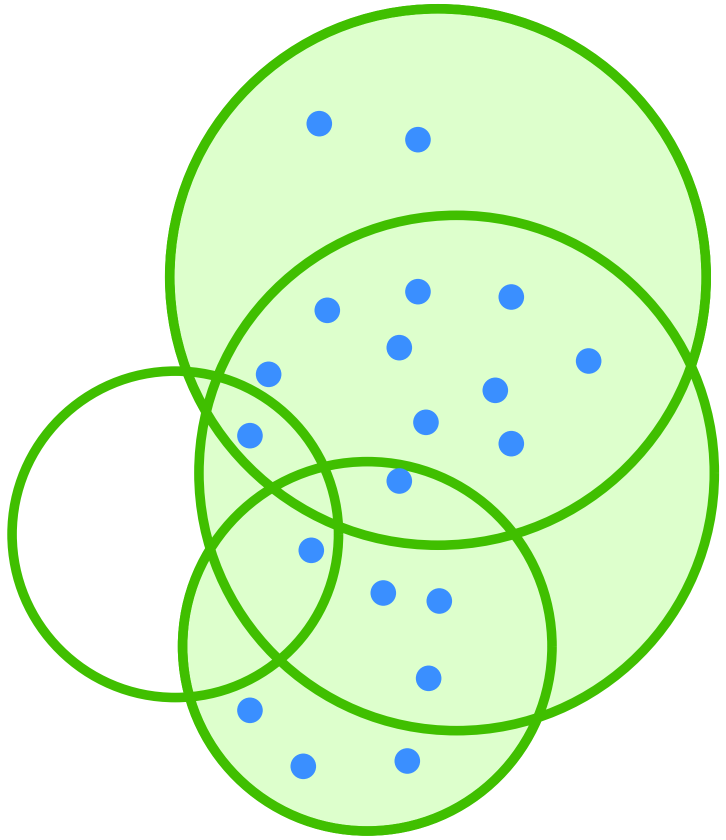


Greedy Approximation

size of greedy solution: $O(k \log n)$,
n: number of points, k: size of optimal solution

Can we do better?

no, for general set systems: no polynomial-time approximation algorithm with approximation factor better that $O(\log n)$ (if $P \neq NP$)



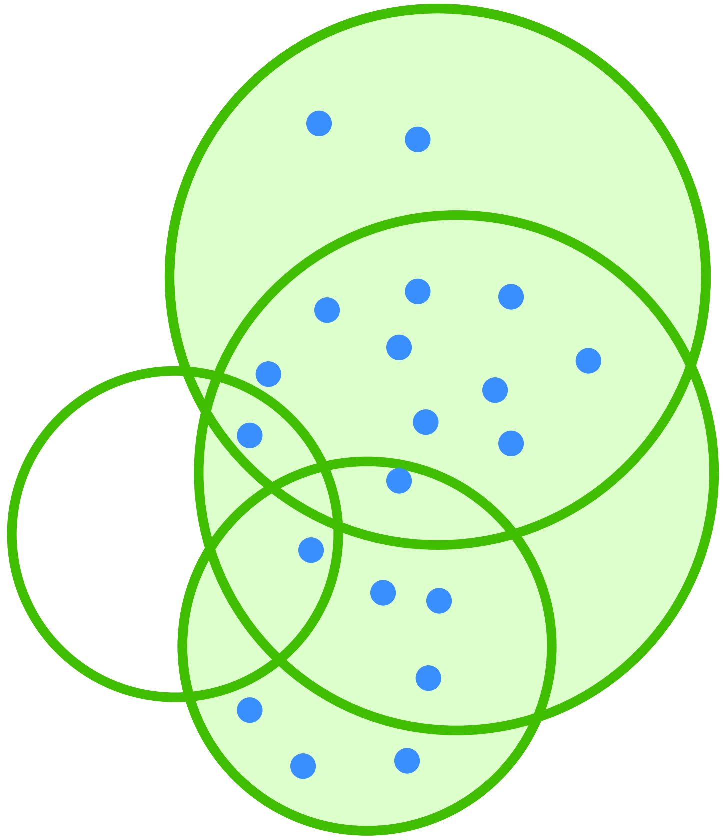
Greedy Approximation

size of greedy solution: $O(k \log n)$,
n: number of points, k: size of optimal solution

Can we do better?

no, for general set systems: no polynomial-time approximation algorithm with approximation factor better that $O(\log n)$ (if $P \neq NP$)

yes, for geometric range spaces



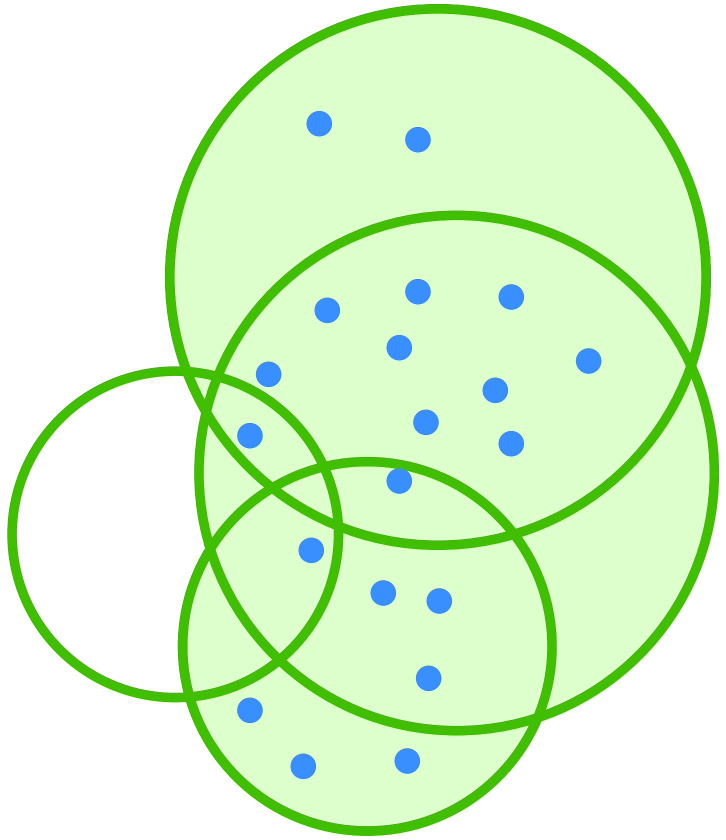
Greedy Approximation

size of greedy solution: $O(k \log n)$,
n: number of points, k: size of optimal solution

Can we do better?

no, for general set systems: no polynomial-time approximation algorithm with approximation factor better that $O(\log n)$ (if $P \neq NP$)

yes, for geometric range spaces
today: algorithm with solution of size $O(k \log k)$

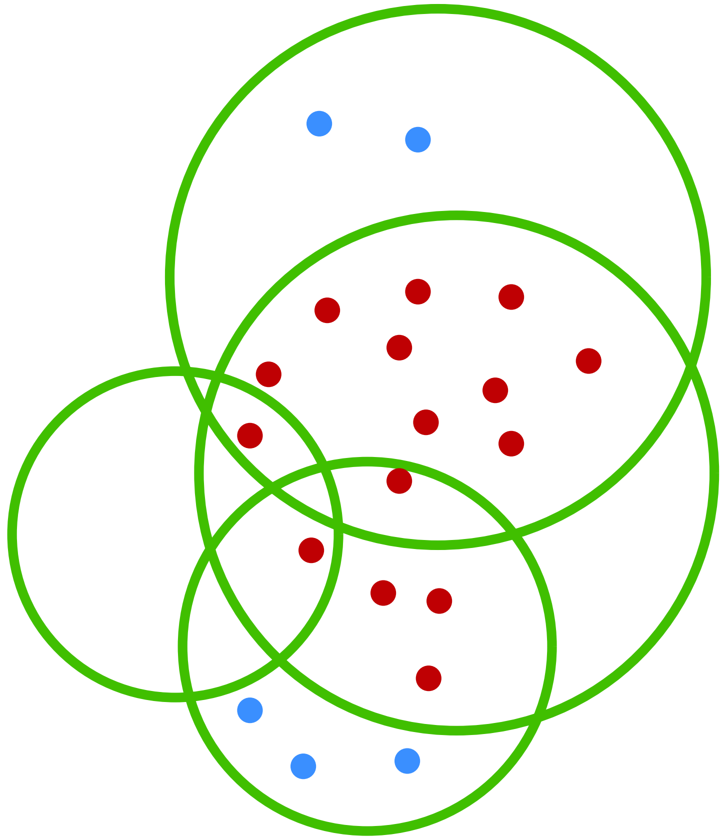


Dual range spaces

warm-up: covering points that are in many ranges

Dual range space

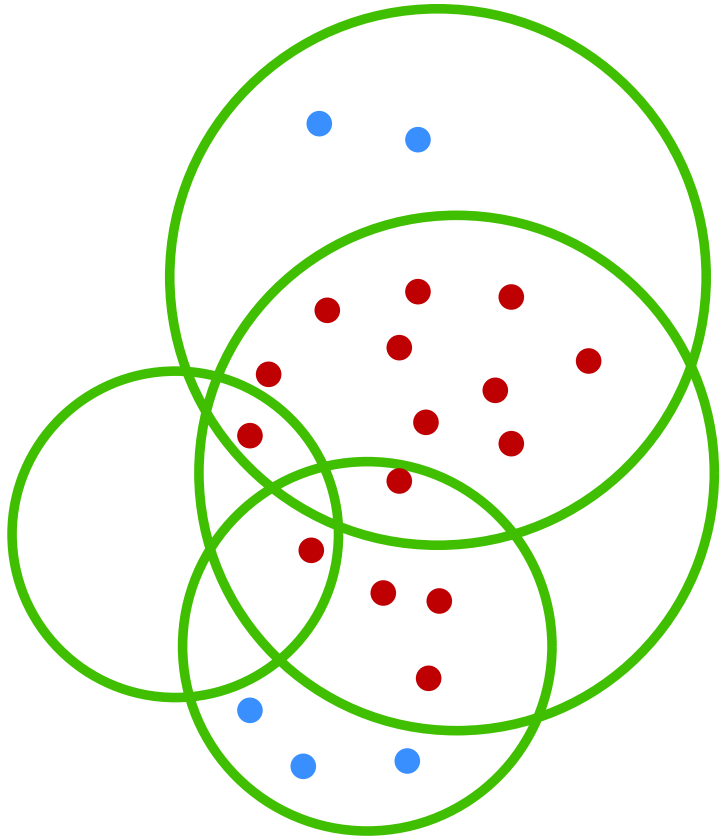
Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering
the points that are in at least $\varepsilon|\mathcal{D}|$ ranges.



Dual range space

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering
the points that are in at least $\varepsilon|\mathcal{D}|$ ranges.

Dual range space of $(X, \mathcal{R})$: $(\mathcal{R}, X^*)$,
where $X^* = \{\mathcal{R}_p \mid p \in X\}, \mathcal{R}_p = \{r \in R \mid p \in r\}$



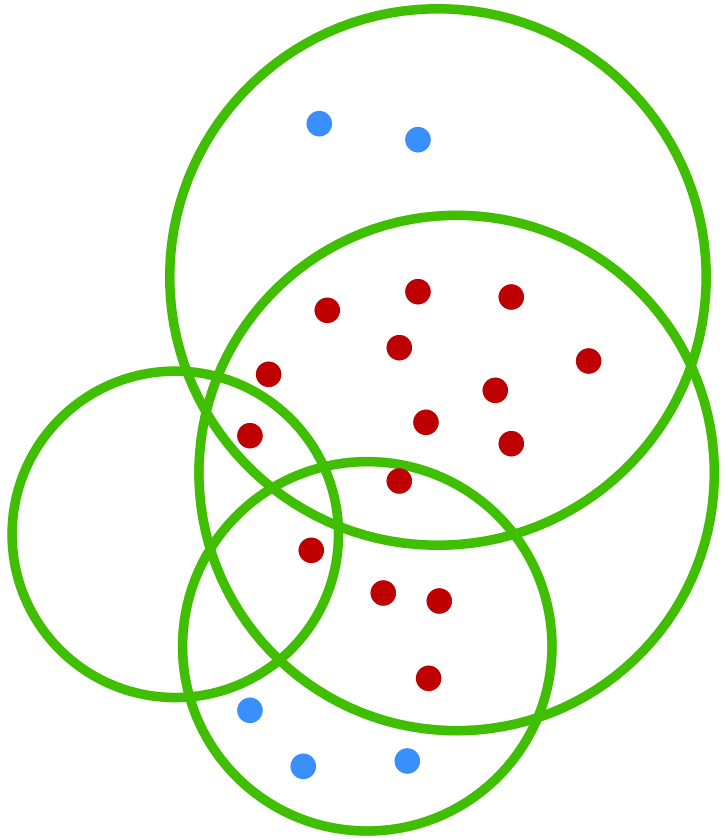
Dual range space

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering
the points that are in at least $\varepsilon|\mathcal{D}|$ ranges.

Dual range space of $(X, \mathcal{R})$: $\quad (\mathcal{R}, X^*)$,
where $X^* = \{\mathcal{R}_p \mid p \in X\}, \mathcal{R}_p = \{r \in R \mid p \in r\}$

Here: $\mathcal{R} = \mathcal{D}$,

dual ranges: set of disks with a common point

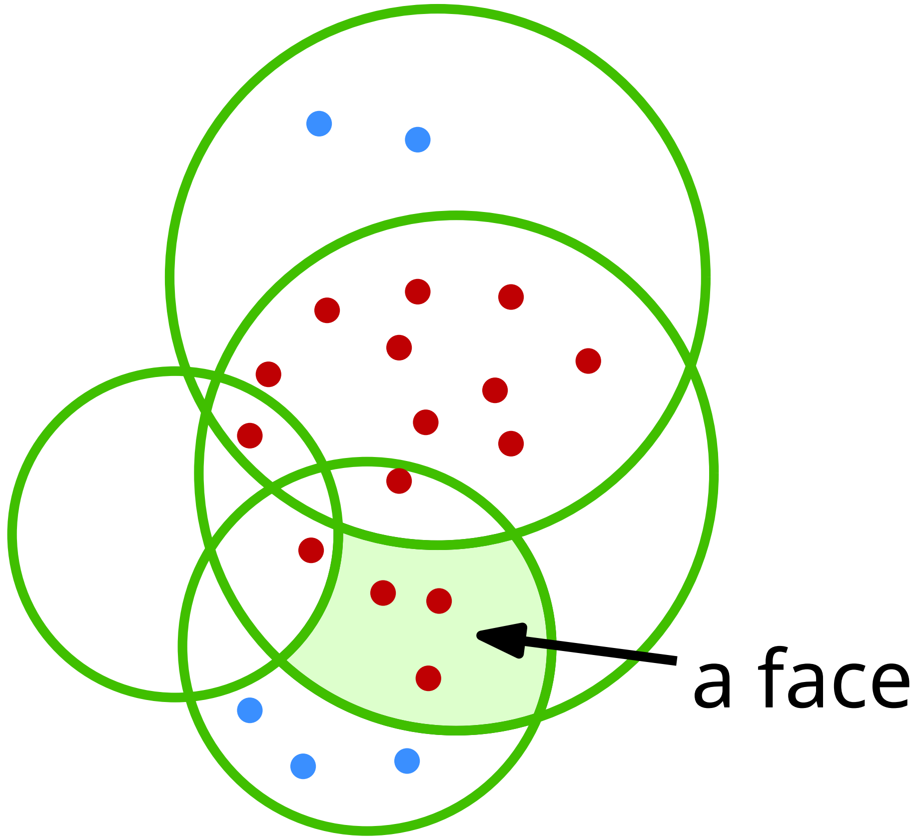


Dual range space

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering
the points that are in at least $\varepsilon|\mathcal{D}|$ ranges.

Dual range space of $(X, \mathcal{R})$: $(\mathcal{R}, X^*)$,
where $X^* = \{\mathcal{R}_p \mid p \in X\}, \mathcal{R}_p = \{r \in R \mid p \in r\}$

Here: $\mathcal{R} = \mathcal{D}$,
dual ranges: set of disks with a common point



Intuition:

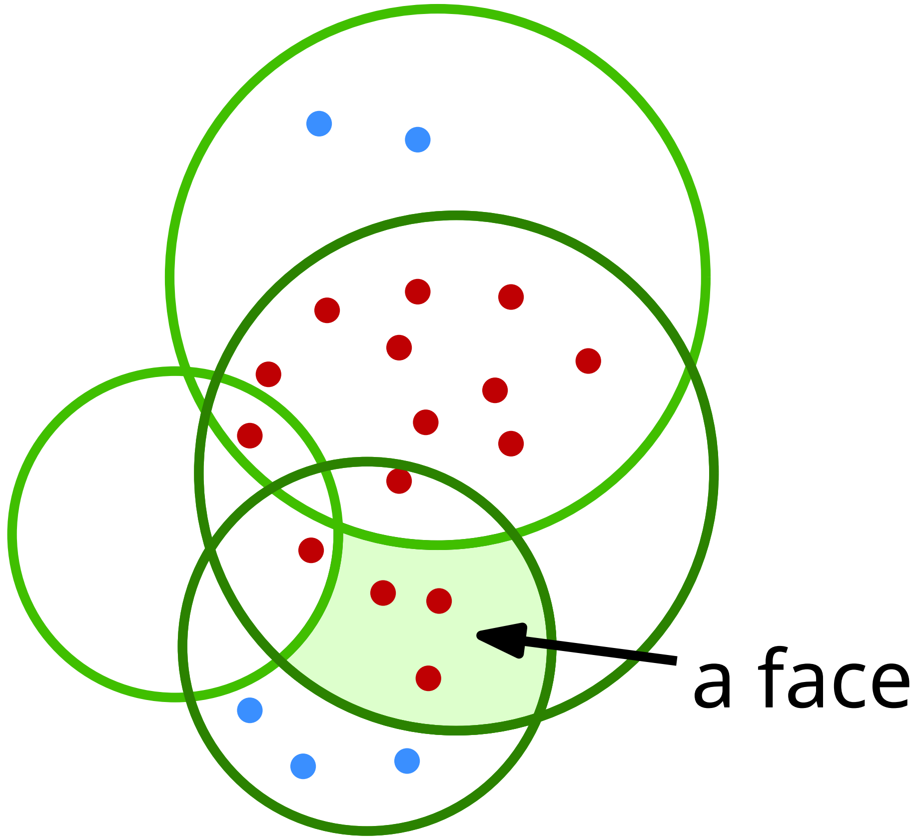
each face in the arrangement of disks corresponds to a dual range, namely to set of disks that include this face

Dual range space

Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering
the points that are in at least $\varepsilon|\mathcal{D}|$ ranges.

Dual range space of $(X, \mathcal{R})$: $(\mathcal{R}, X^*)$,
where $X^* = \{\mathcal{R}_p \mid p \in X\}, \mathcal{R}_p = \{r \in R \mid p \in r\}$

Here: $\mathcal{R} = \mathcal{D}$,
dual ranges: set of disks with a common point



Intuition:

each face in the arrangement of disks corresponds to a dual range, namely to set of disks that include this face

Dual range space

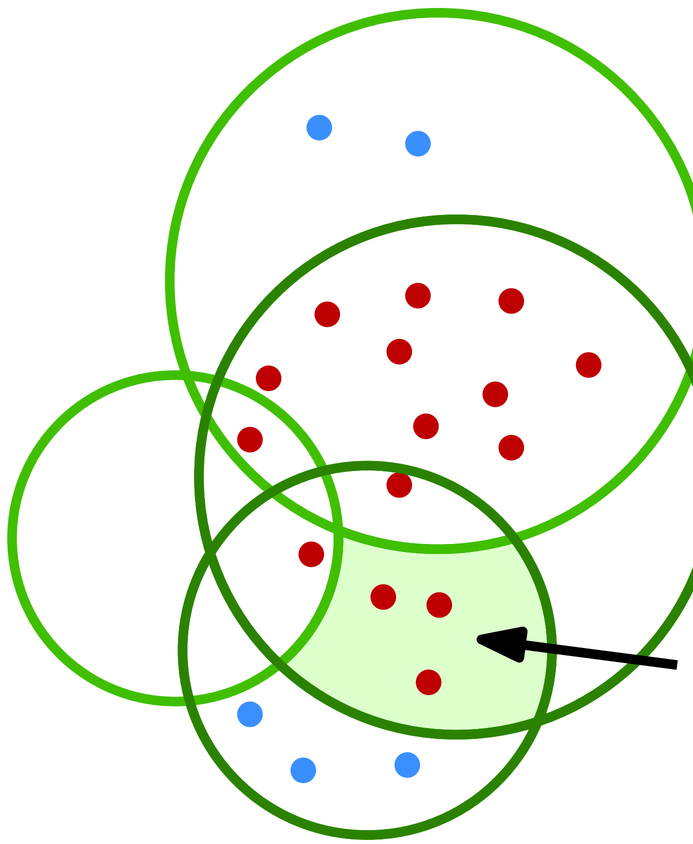
Given: a set of points P and a set of disks $\mathcal{D}$,
find a smallest set of disks covering
the points that are in at least $\varepsilon|\mathcal{D}|$ ranges.

asks for ε-net in dual range space

Dual range space of $(X, \mathcal{R})$: $(\mathcal{R}, X^*)$,
where $X^* = \{\mathcal{R}_p \mid p \in X\}, \mathcal{R}_p = \{r \in R \mid p \in r\}$

Here: $\mathcal{R} = \mathcal{D}$,

dual ranges: set of disks with a common point



a face

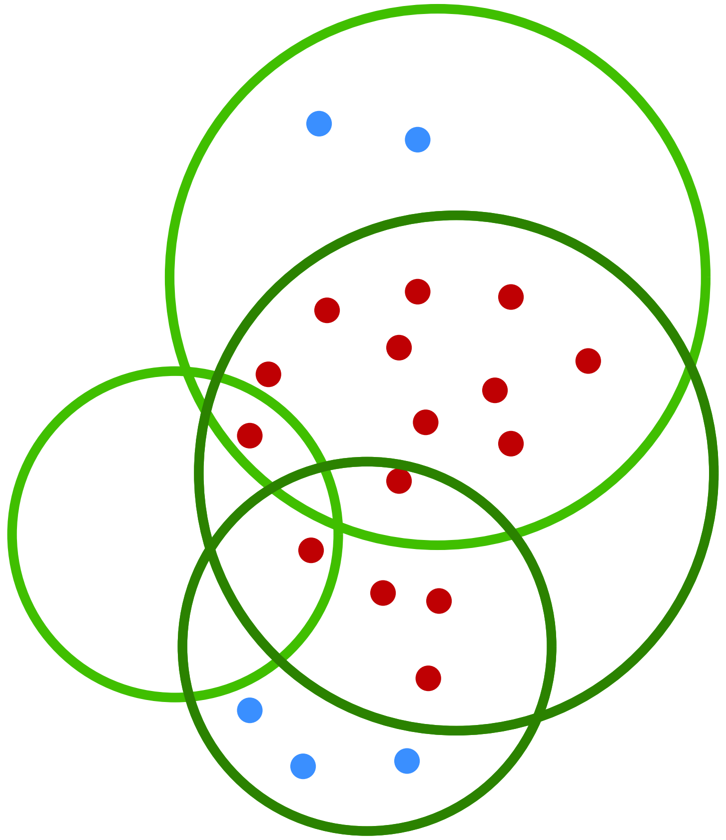
Intuition:

each face in the arrangement of disks corresponds to a dual range, namely to set of disks that include this face

Dual range space

Given: a set of points P and a set of disks $\mathcal{D}$, find a smallest set of disks covering the points that are in at least $\varepsilon|\mathcal{D}|$ ranges.

asks for ε-net in dual range space

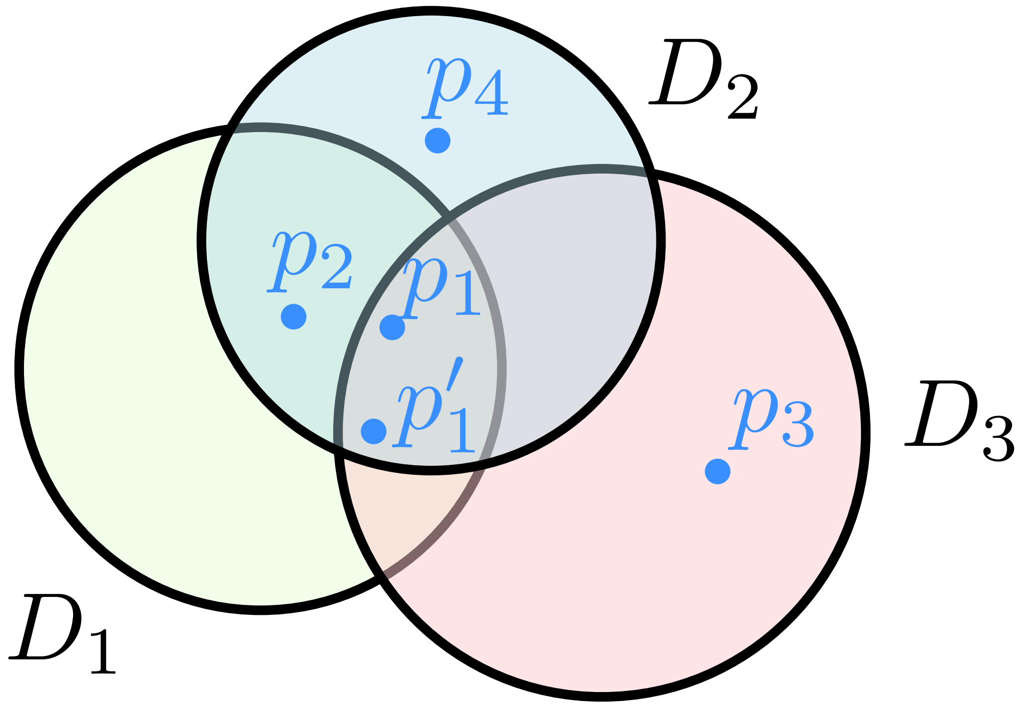


Dual range space of $(X, \mathcal{R})$: $(\mathcal{R}, X^*)$, where $X^* = \{\mathcal{R}_p \mid p \in X\}, \mathcal{R}_p = \{r \in R \mid p \in r\}$

VC-dimension of dual range space: δ^*

ε-net theorem:
A random sample of the disks of size $O\left(\frac{\delta^*}{\varepsilon} \log \frac{1}{\varepsilon}\right)$ is an ε-net with probability $\geq 1/2$

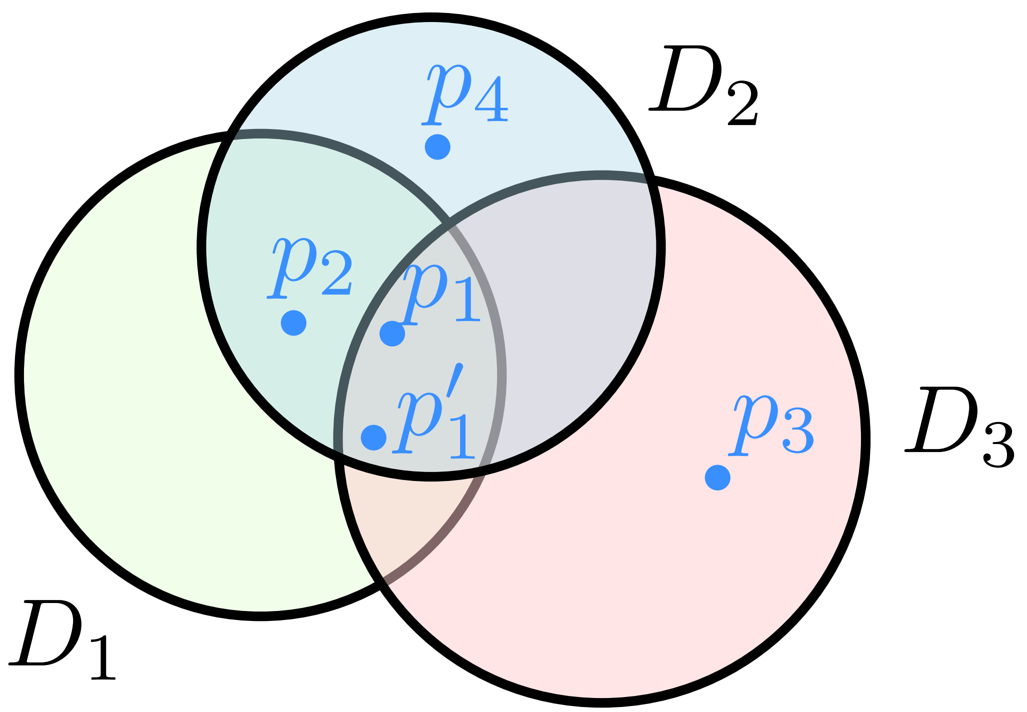
Dual range space: matrix interpretation



incidence matrix of range space

	p_1'	p_1	p_2	p_3	p_4
D_1	1	1	1	0	0
D_2	1	1	1	0	1
D_3	1	1	0	1	0

Dual range space: matrix interpretation



incidence matrix of range space

	p_1'	p_1	p_2	p_3	p_4
D_1	1	1	1	0	0
D_2	1	1	1	0	1
D_3	1	1	0	1	0

We obtain the matrix of the dual range space by transposition + removing/merging duplicates

	D_1	D_2	D_3
p_1'	1	1	1
p_1	1	1	1
p_2	1	1	0
p_3	0	0	1
p_4	0	1	0

Dual range space: VC-dimension

If a range space has VC-dim δ, then the dual VC-dim $\delta^* < 2^{\delta+1}$

. . . or in short: if δ is constant, so is δ^*.

Dual range space: VC-dimension

If a range space has VC-dim δ, then the dual VC-dim $\delta^* < 2^{\delta+1}$

Why?

Suppose $\delta^* \geq 2^{\delta'}$ and prove that $\delta \geq \delta'$, e.g., below $\delta^* = 4 = 2^2$

$$
\begin{array}{ccccccccccccccc}
 & p_1 & p_2 & \dots & & & & & & & & & & & \\
r_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
r_2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
r_3 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
r_4 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}
$$

Dual range space: VC-dimension

If a range space has VC-dim δ, then the dual VC-dim $\delta^* < 2^{\delta+1}$

Why?

Suppose $\delta^* \geq 2^{\delta'}$ and prove that $\delta \geq \delta'$, e.g., below $\delta^* = 4 = 2^2$

There are δ^* rows (= ranges) for which all 2^{δ^*} columns occur

$$
\begin{array}{cccccccccccccccc}
 & p_1 & p_2 & \dots & & & & & & & & & & & & \\
r_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
r_2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
r_3 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
r_4 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}
$$

Dual range space: VC-dimension

If a range space has VC-dim δ, then the dual VC-dim $\delta^* < 2^{\delta+1}$

Why?

Suppose $\delta^* \geq 2^{\delta'}$ and prove that $\delta \geq \delta'$, e.g., below $\delta^* = 4 = 2^2$

There are δ^* rows (= ranges) for which all 2^{δ^*} columns occur

Take columns that together count from 0 to $\delta^* - 1 = 2^{\delta'} - 1$ in binary

$$
\begin{array}{ccccccccccccccc}
 & p_1 & p_2 & \dots & & & & & & & & & & & \\
r_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
r_2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
r_3 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
r_4 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}
$$

Dual range space: VC-dimension

If a range space has VC-dim δ, then the dual VC-dim $\delta^* < 2^{\delta+1}$

Why?

Suppose $\delta^* \geq 2^{\delta'}$ and prove that $\delta \geq \delta'$, e.g., below $\delta^* = 4 = 2^2$

There are δ^* rows (= ranges) for which all 2^{δ^*} columns occur

Take columns that together count from 0 to $\delta^* - 1 = 2^{\delta'} - 1$ in binary

These are $\delta' = \log \delta^*$ shattered columns $\Rightarrow \delta \geq \delta'$

	p_1	p_2	$\dots$											
r_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
r_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1
r_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0
r_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Quiz

In the previous proof: Could we have picked two different columns?

A no

B yes, we could have picked the second column differently

C yes, we could have picked completely different columns

$$\begin{array}{ccccccccccccccc}
 & p_1 & p_2 & \dots & & & & & & & & & & & \\
r_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
r_2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
r_3 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
r_4 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}$$

Quiz

In the previous proof: Could we have picked two different columns?

A no

B yes, we could have picked the second column differently

$\boxed{\text{C} \quad \text{yes, we could have picked completely different columns}}$

$$
\begin{array}{ccccccccccccccc}
 & & & & & & & & & & & \downarrow & \text{e.g.} & \downarrow & \\
 & p_1 & p_2 & \dots & & & & & & & & & & & \\
r_1 & 0 & 0 & 0 & {\color{blue}0} & 0 & {\color{blue}0} & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
r_2 & 0 & 0 & 0 & {\color{blue}0} & 1 & {\color{blue}1} & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
r_3 & 0 & 0 & 1 & {\color{blue}1} & 0 & {\color{blue}0} & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
r_4 & 0 & 1 & 0 & {\color{blue}1} & 0 & {\color{blue}1} & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}
$$

Approximation Algorithm for Geometric Set Cover

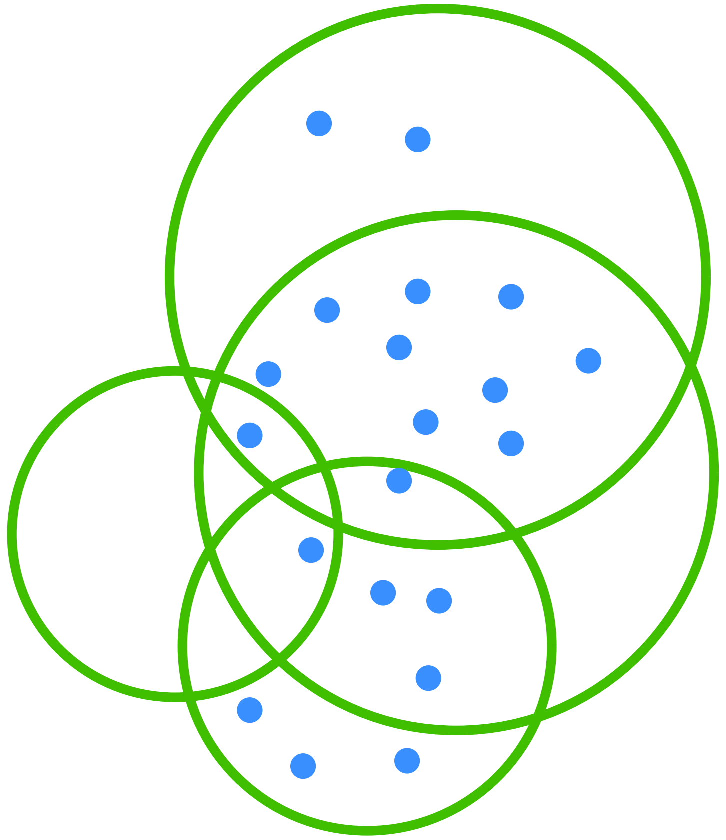
sampling ranges with reweighing

The algorithm: preparation

Given $(X, \mathcal{R})$, $n = |X|$, $m = |\mathcal{R}|$, and dual VC-dimension δ^*, the algorithm computes a set cover which uses $\mathcal{O}(\delta^* k \cdot \log(\delta^* k))$ sets where k is the number of sets used by the optimal solution.

The algorithm: preparation

Given $(X, \mathcal{R})$, $n = |X|$, $m = |\mathcal{R}|$, and dual VC-dimension δ^*, the algorithm computes a set cover which uses $\mathcal{O}(\delta^* k \cdot \log(\delta^* k))$ sets where k is the number of sets used by the optimal solution.



Algorithm assumes k is known. We run it for $k = 1, 2, 4, 8, \ldots$ until it finds a solution

Pick $\varepsilon = \frac{1}{4k}$

Assign each range r a weight $W(r)$. Initially, $W(r) = 1$

$W(\mathcal{R})$ is the total weight of all the ranges in $\mathcal{R}$

$\mathcal{R}'$ is a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$

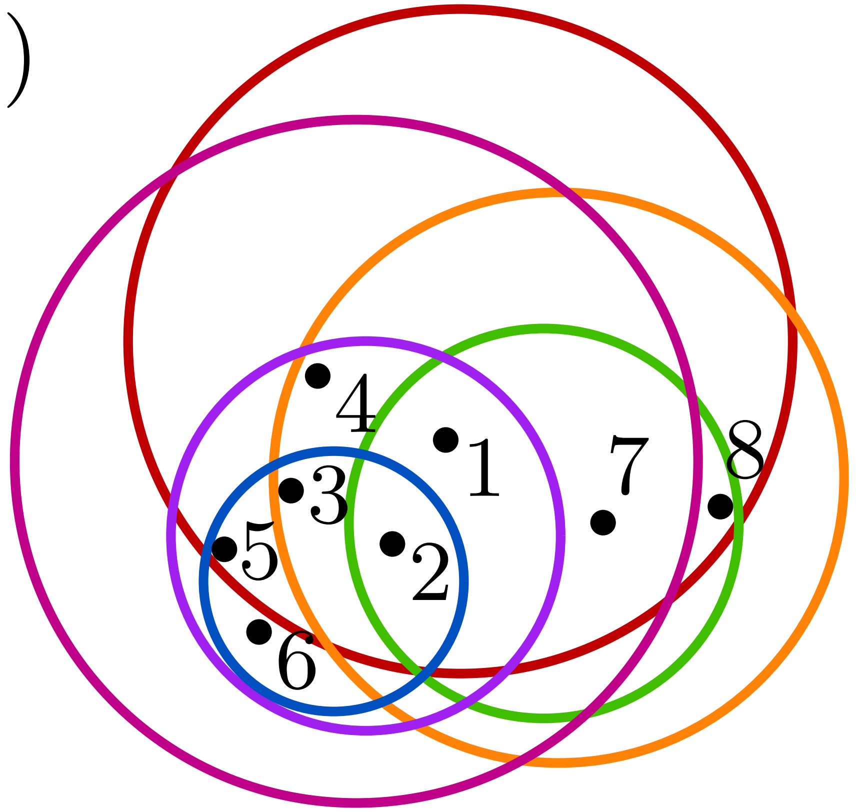
Each range $r \in \mathcal{R}$ has probability $W(r)/W(\mathcal{R})$ of being selected

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



$W(r)$

$1 \quad red = \{1, 2, 3, 4, 5, 7, 8\}$

$1 \quad green = \{1, 2, 7, 8\}$

$1 \quad orange = \{1, 2, 3, 4, 7, 8\}$

$1 \quad purple = \{1, 2, 3, 4, 5, 6\}$

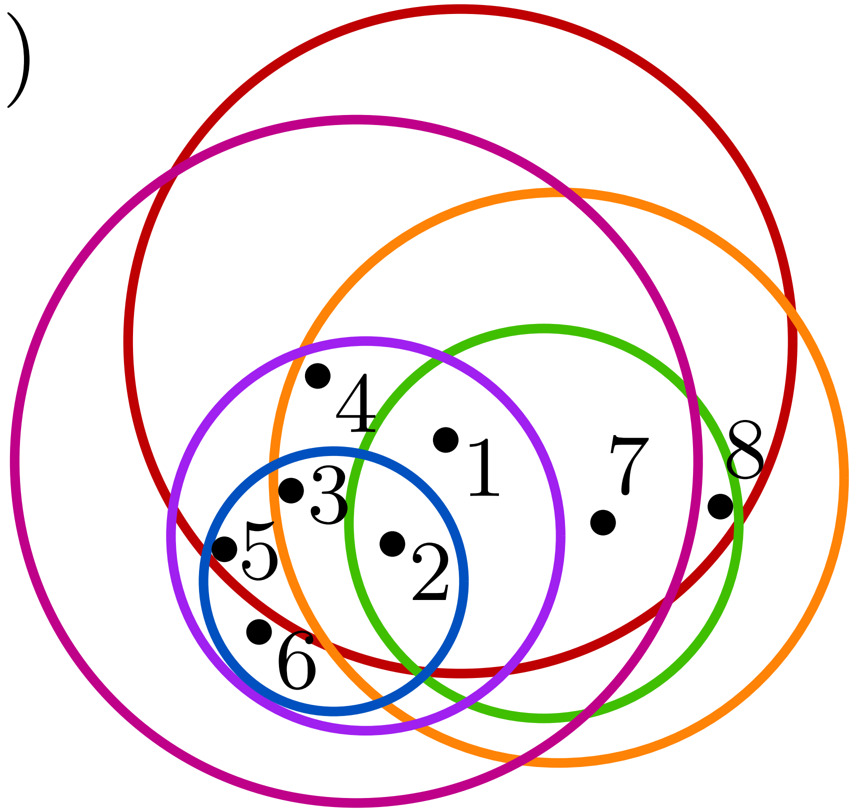
$1 \quad blue = \{2, 3, 5, 6\}$

$1 \quad pink = \{1, 2, 3, 4, 5, 6, 7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$

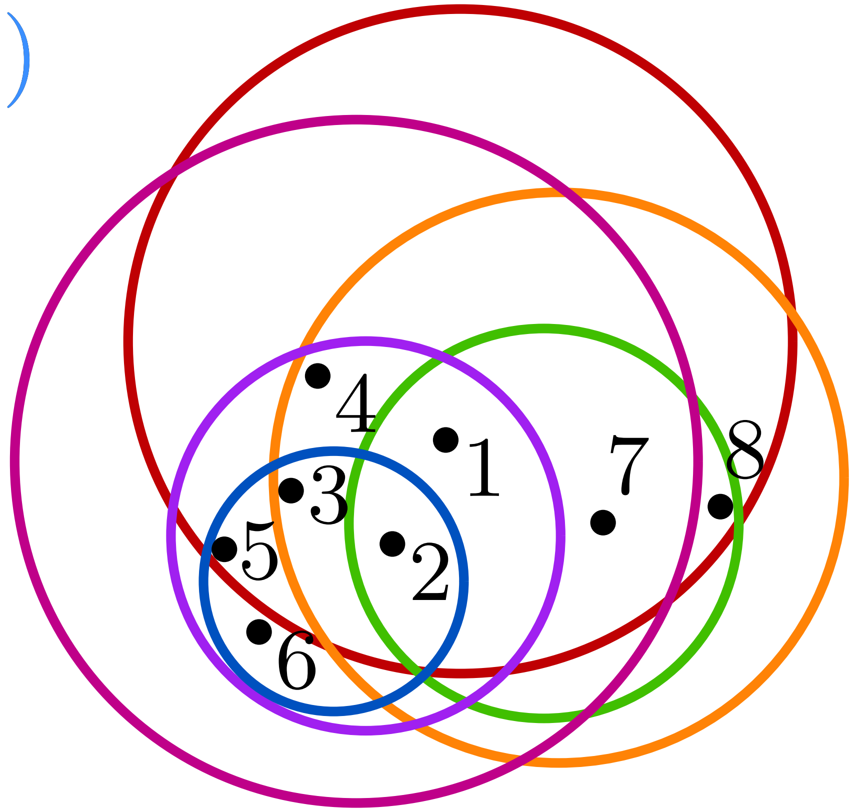
Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$



$W(r)$	
1	$red = \{1,2,3,4,5,7,8\}$
1	$green = \{1,2,7,8\}$
1	$orange = \{1,2,3,4,7,8\}$
1	$purple = \{1,2,3,4,5,6\}$
1	$blue = \{2,3,5,6\}$
1	$pink = \{1,2,3,4,5,6,7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

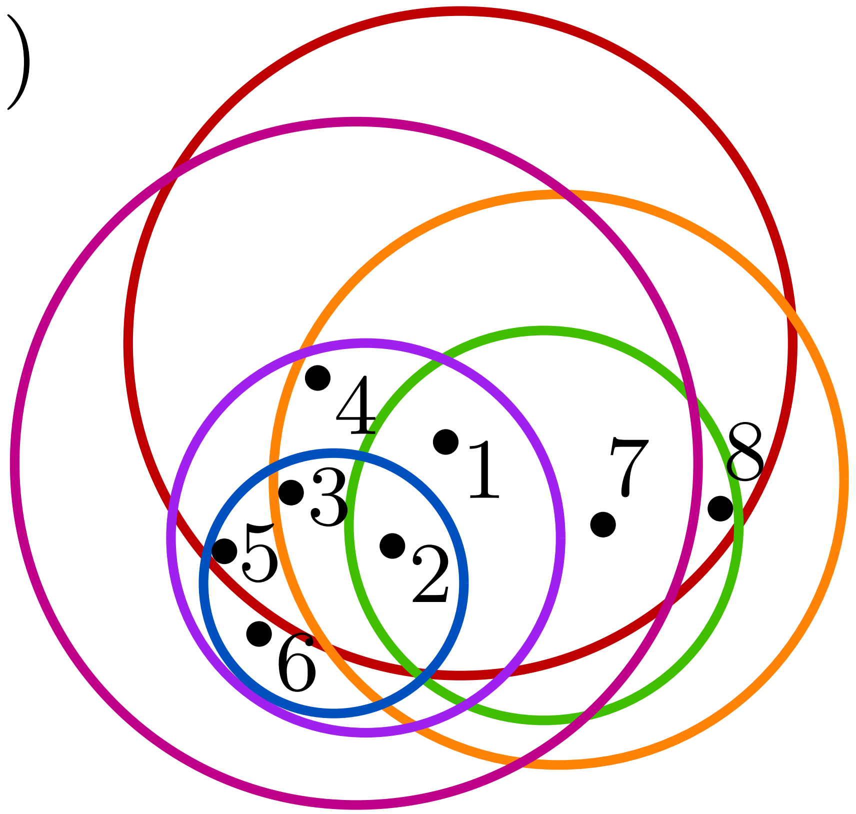
sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{red, green\}$

$W(r)$

1 $red = \{1,2,3,4,5,7,8\}$
1 $green = \{1,2,7,8\}$
1 $orange = \{1,2,3,4,7,8\}$
1 $purple = \{1,2,3,4,5,6\}$
1 $blue = \{2,3,5,6\}$
1 $pink = \{1,2,3,4,5,6,7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



$W(r)$

$1 \quad red = \{1,2,3,4,5,7,8\}$

$1 \quad green = \{1,2,7,8\}$

$1 \quad orange = \{1,2,3,4,7,8\}$

$1 \quad purple = \{1,2,3,4,5,6\}$

$1 \quad blue = \{2,3,5,6\}$

$1 \quad pink = \{1,2,3,4,5,6,7\}$

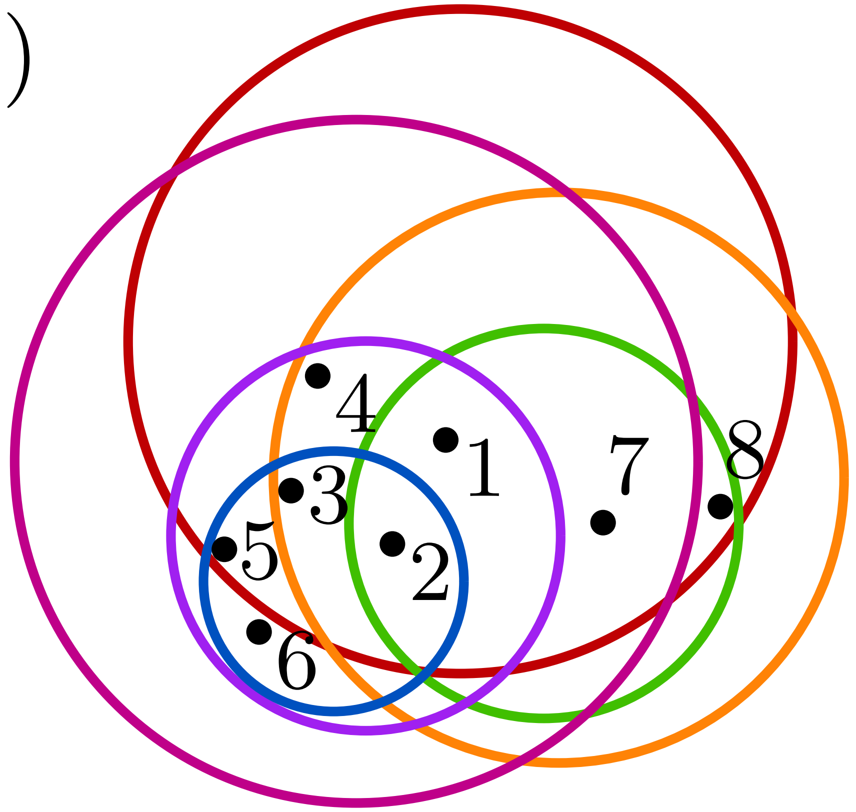
Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{red, green\}$

$\mathcal{R}'$ does not cover X, namely 6 is not covered

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{red, green\}$

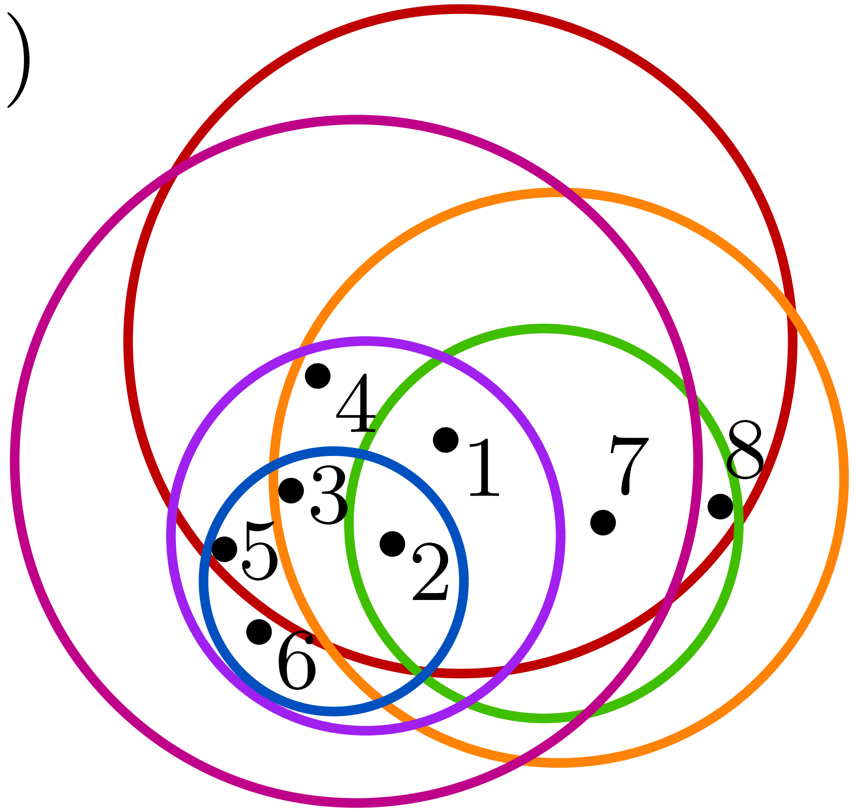
$\mathcal{R}'$ does not cover X, namely 6 is not covered

$\mathcal{R}_6 = \{purple, blue, pink\}$

$W(r)$	
1	$red = \{1,2,3,4,5,7,8\}$
1	$green = \{1,2,7,8\}$
1	$orange = \{1,2,3,4,7,8\}$
1	$purple = \{1,2,3,4,5,6\}$
1	$blue = \{2,3,5,6\}$
1	$pink = \{1,2,3,4,5,6,7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



$W(r)$	
1	$red = \{1,2,3,4,5,7,8\}$
1	$green = \{1,2,7,8\}$
1	$orange = \{1,2,3,4,7,8\}$
1	$purple = \{1,2,3,4,5,6\}$
1	$blue = \{2,3,5,6\}$
1	$pink = \{1,2,3,4,5,6,7\}$

Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{red, green\}$

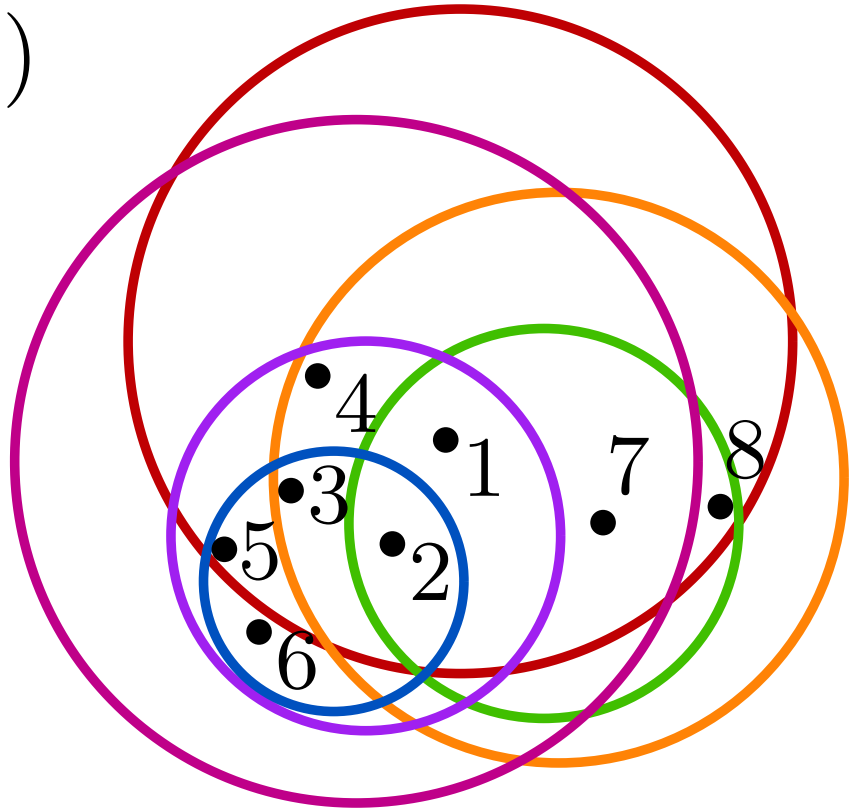
$\mathcal{R}'$ does not cover X, namely 6 is not covered

$\mathcal{R}_6 = \{purple, blue, pink\}$

$W(\mathcal{R}_6) = 3 < 4 = (2/3) \cdot 6 = \varepsilon W(\mathcal{R})$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{red, green\}$

$\mathcal{R}'$ does not cover X, namely 6 is not covered

$\mathcal{R}_6 = \{purple, blue, pink\}$

$W(\mathcal{R}_6) = 3 < 4 = (2/3) \cdot 6 = \varepsilon W(\mathcal{R})$

double

$W(r)$

$1 \quad red = \{1,2,3,4,5,7,8\}$

$1 \quad green = \{1,2,7,8\}$

$1 \quad orange = \{1,2,3,4,7,8\}$

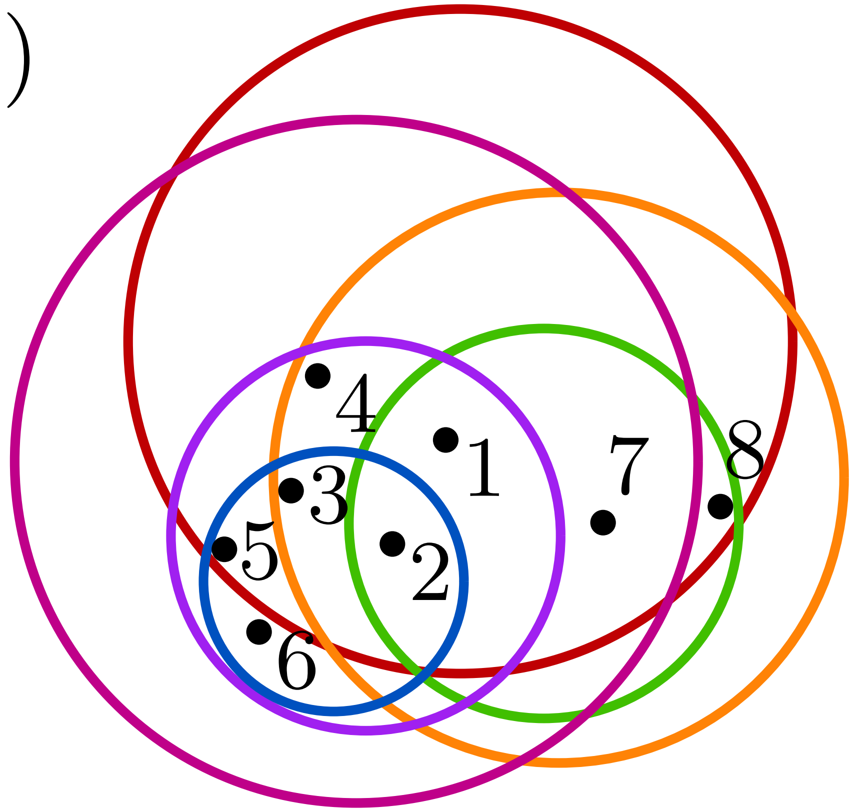
$1 \quad purple = \{1,2,3,4,5,6\}$

$1 \quad blue = \{2,3,5,6\}$

$1 \quad pink = \{1,2,3,4,5,6,7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



$W(r)$

1	$red = \{1,2,3,4,5,7,8\}$
1	$green = \{1,2,7,8\}$
1	$orange = \{1,2,3,4,7,8\}$
2	$purple = \{1,2,3,4,5,6\}$
2	$blue = \{2,3,5,6\}$
2	$pink = \{1,2,3,4,5,6,7\}$

Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{red, green\}$

$\mathcal{R}'$ does not cover X, namely 6 is not covered

$\mathcal{R}_6 = \{purple, blue, pink\}$

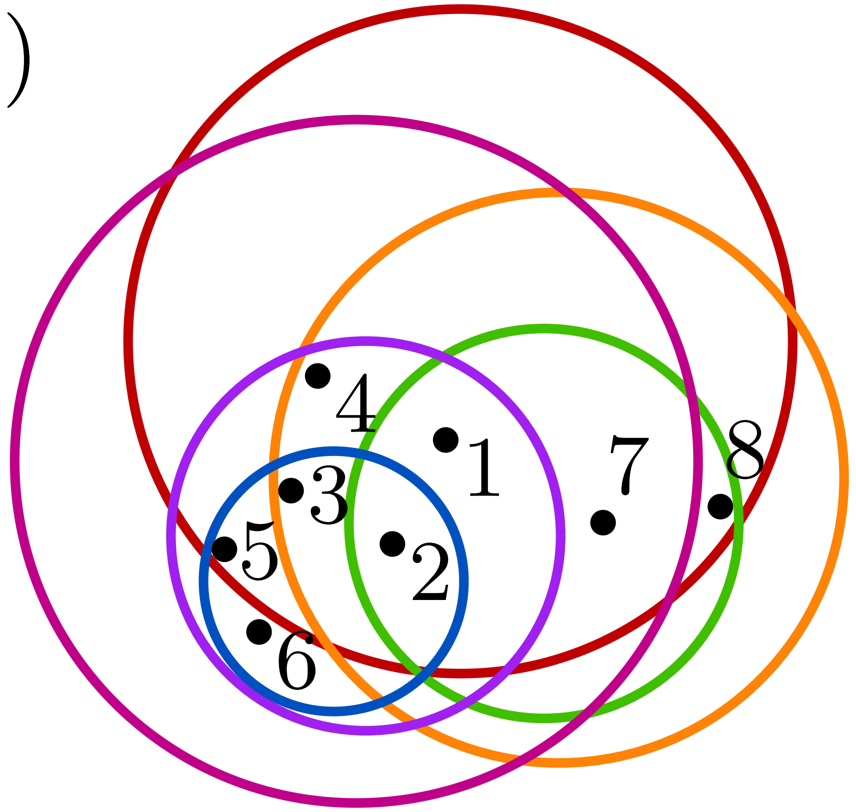
$W(\mathcal{R}_6) = 3 < 4 = (2/3)\cdot 6 = \varepsilon W(\mathcal{R})$

double

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$

Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$



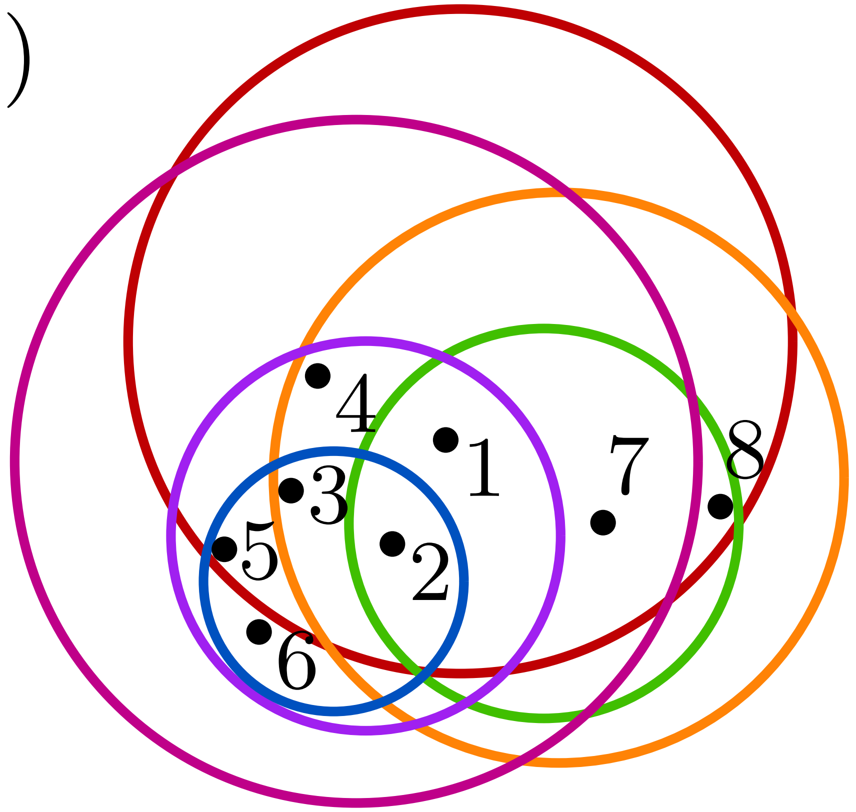
$W(r)$	
1	$red = \{1,2,3,4,5,7,8\}$
1	$green = \{1,2,7,8\}$
1	$orange = \{1,2,3,4,7,8\}$
2	$purple = \{1,2,3,4,5,6\}$
2	$blue = \{2,3,5,6\}$
2	$pink = \{1,2,3,4,5,6,7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$

Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

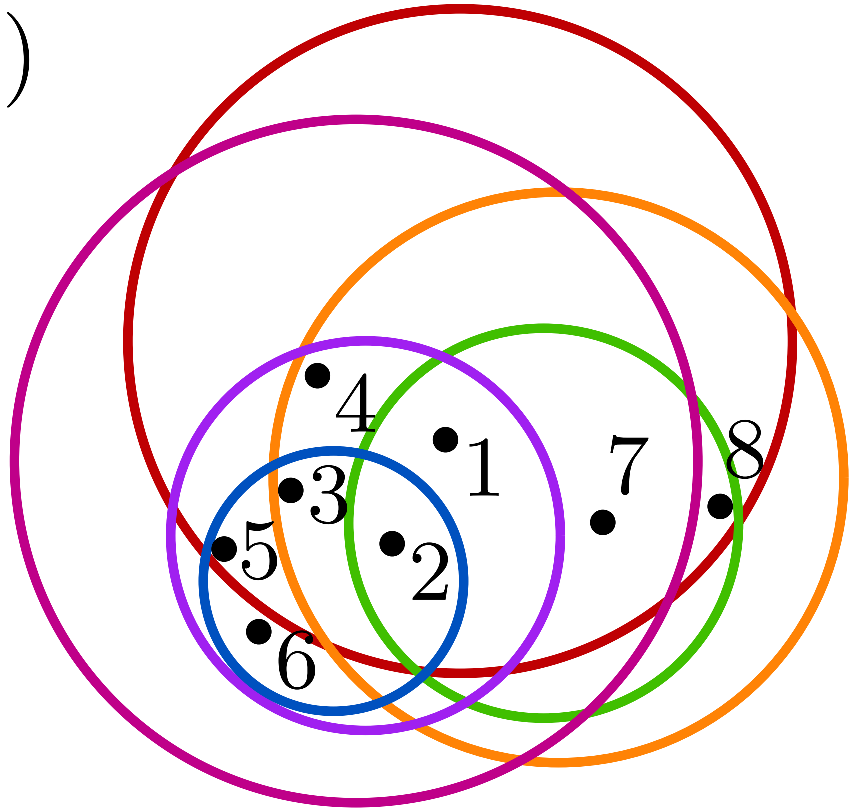
sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{purple, blue\}$



$W(r)$	
1	$red = \{1,2,3,4,5,7,8\}$
1	$green = \{1,2,7,8\}$
1	$orange = \{1,2,3,4,7,8\}$
2	$purple = \{1,2,3,4,5,6\}$
2	$blue = \{2,3,5,6\}$
2	$pink = \{1,2,3,4,5,6,7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

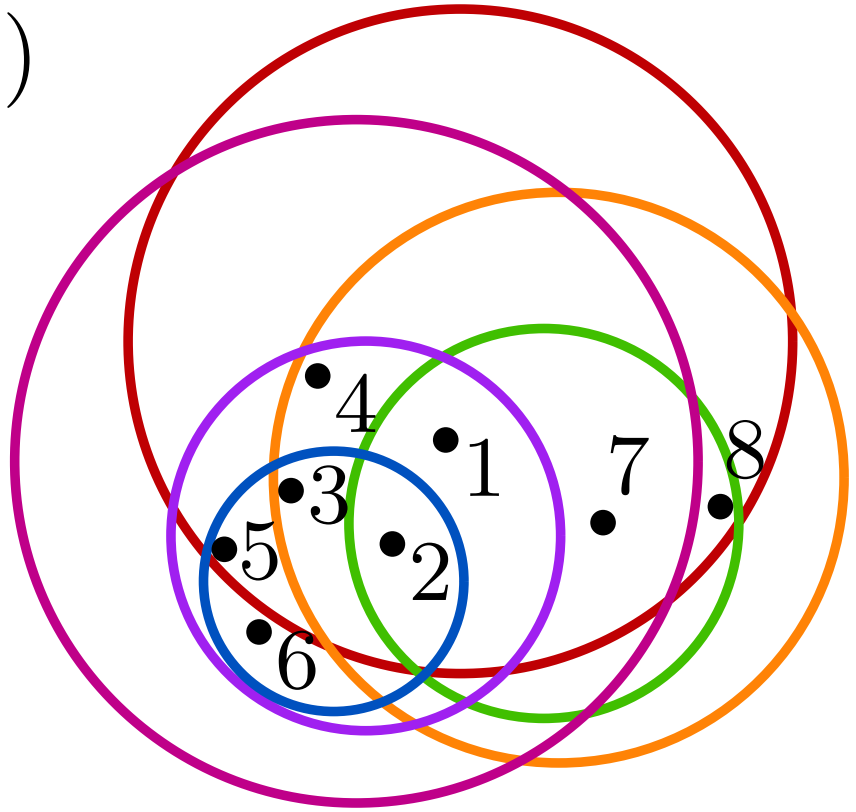
sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{purple, blue\}$

$\mathcal{R}'$ does not cover X, namely 7 is not covered

$W(r)$	
1	$red = \{1,2,3,4,5,7,8\}$
1	$green = \{1,2,7,8\}$
1	$orange = \{1,2,3,4,7,8\}$
2	$purple = \{1,2,3,4,5,6\}$
2	$blue = \{2,3,5,6\}$
2	$pink = \{1,2,3,4,5,6,7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



$W(r)$	
1	$red = \{1,2,3,4,5,7,8\}$
1	$green = \{1,2,7,8\}$
1	$orange = \{1,2,3,4,7,8\}$
2	$purple = \{1,2,3,4,5,6\}$
2	$blue = \{2,3,5,6\}$
2	$pink = \{1,2,3,4,5,6,7\}$

Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

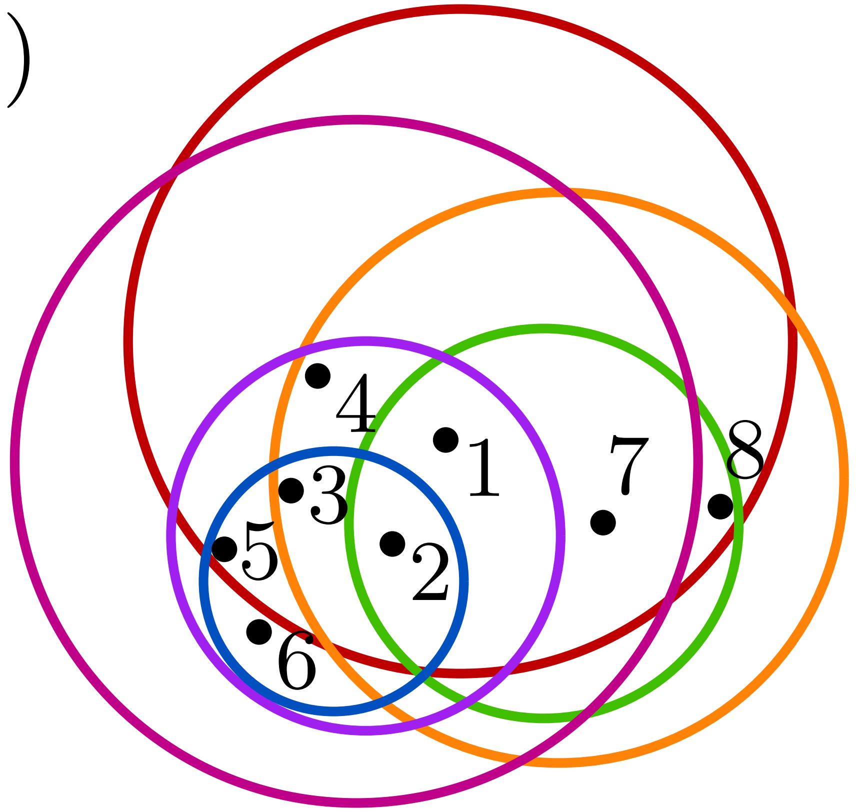
sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{purple, blue\}$

$\mathcal{R}'$ does not cover X, namely 7 is not covered

$\mathcal{R}_7 = \{red, green, orange, pink\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{purple, blue\}$

$\mathcal{R}'$ does not cover X, namely 7 is not covered

$\mathcal{R}_7 = \{red, green, orange, pink\}$

$W(\mathcal{R}_7) = 5 < 6 = (2/3) \cdot 9 = \varepsilon W(\mathcal{R})$

$W(r)$

$1 \quad red = \{1, 2, 3, 4, 5, 7, 8\}$

$1 \quad green = \{1, 2, 7, 8\}$

$1 \quad orange = \{1, 2, 3, 4, 7, 8\}$

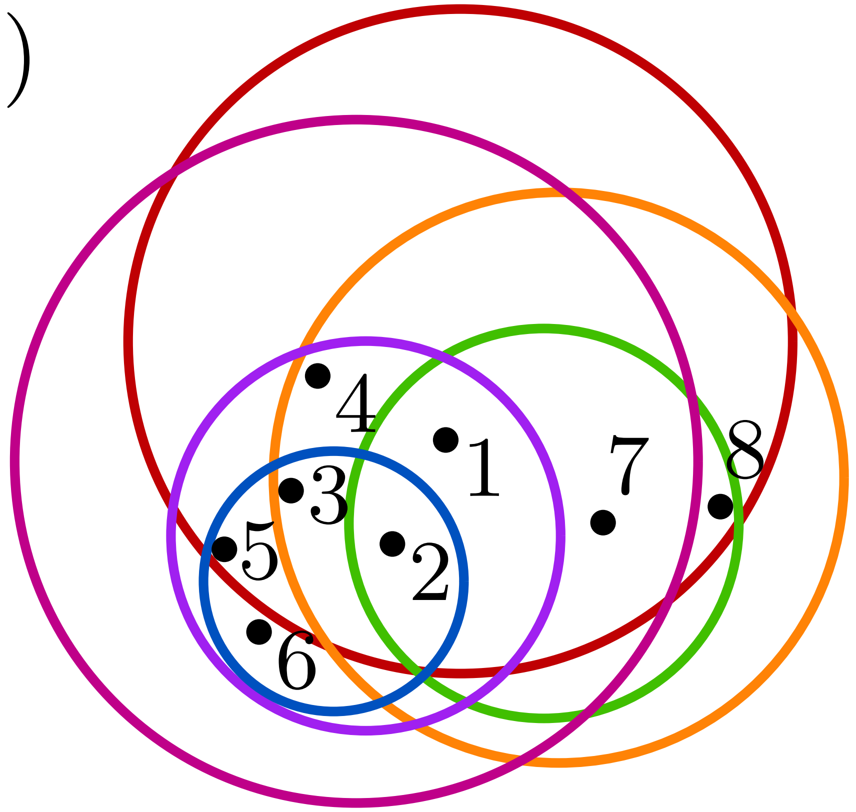
$2 \quad purple = \{1, 2, 3, 4, 5, 6\}$

$2 \quad blue = \{2, 3, 5, 6\}$

$2 \quad pink = \{1, 2, 3, 4, 5, 6, 7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



$W(r)$

1 $\quad red = \{1,2,3,4,5,7,8\}$

1 $\quad green = \{1,2,7,8\}$

1 $\quad orange = \{1,2,3,4,7,8\}$

2 $\quad purple = \{1,2,3,4,5,6\}$

2 $\quad blue = \{2,3,5,6\}$

2 $\quad pink = \{1,2,3,4,5,6,7\}$

Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{purple, blue\}$

$\mathcal{R}'$ does not cover X, namely 7 is not covered

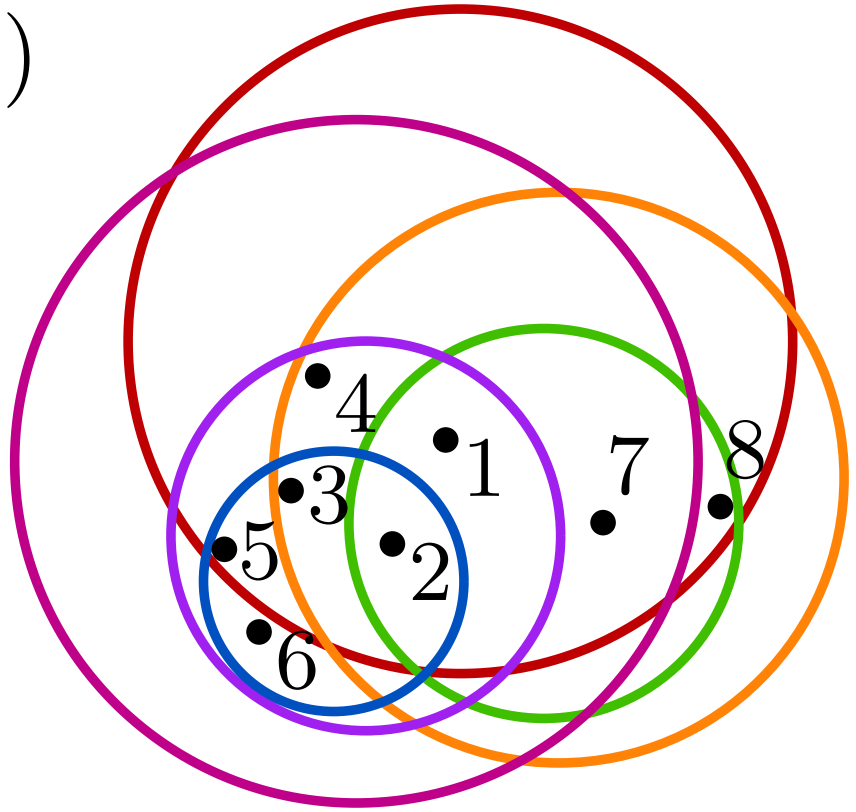
$\mathcal{R}_7 = \{red, green, orange, pink\}$

$W(\mathcal{R}_7) = 5 < 6 = (2/3) \cdot 9 = \varepsilon W(\mathcal{R})$

double

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{purple, blue\}$

$\mathcal{R}'$ does not cover X, namely 7 is not covered

$\mathcal{R}_7 = \{red, green, orange, pink\}$

$W(\mathcal{R}_7) = 5 < 6 = (2/3) \cdot 9 = \varepsilon W(\mathcal{R})$

double

$W(r)$

$2 \quad red = \{1,2,3,4,5,7,8\}$

$2 \quad green = \{1,2,7,8\}$

$2 \quad orange = \{1,2,3,4,7,8\}$

$2 \quad purple = \{1,2,3,4,5,6\}$

$2 \quad blue = \{2,3,5,6\}$

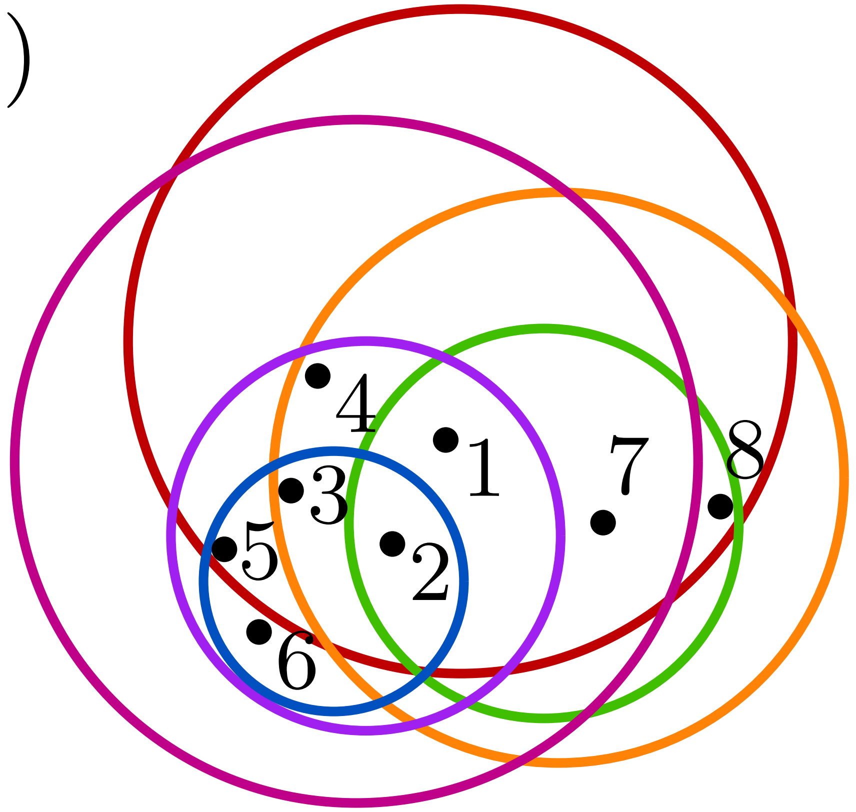
$4 \quad pink = \{1,2,3,4,5,6,7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$

Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{red, pink\}$



$W(r)$

$2 \quad red = \{1,2,3,4,5,7,8\}$

$2 \quad green = \{1,2,7,8\}$

$2 \quad orange = \{1,2,3,4,7,8\}$

$2 \quad purple = \{1,2,3,4,5,6\}$

$2 \quad blue = \{2,3,5,6\}$

$4 \quad pink = \{1,2,3,4,5,6,7\}$

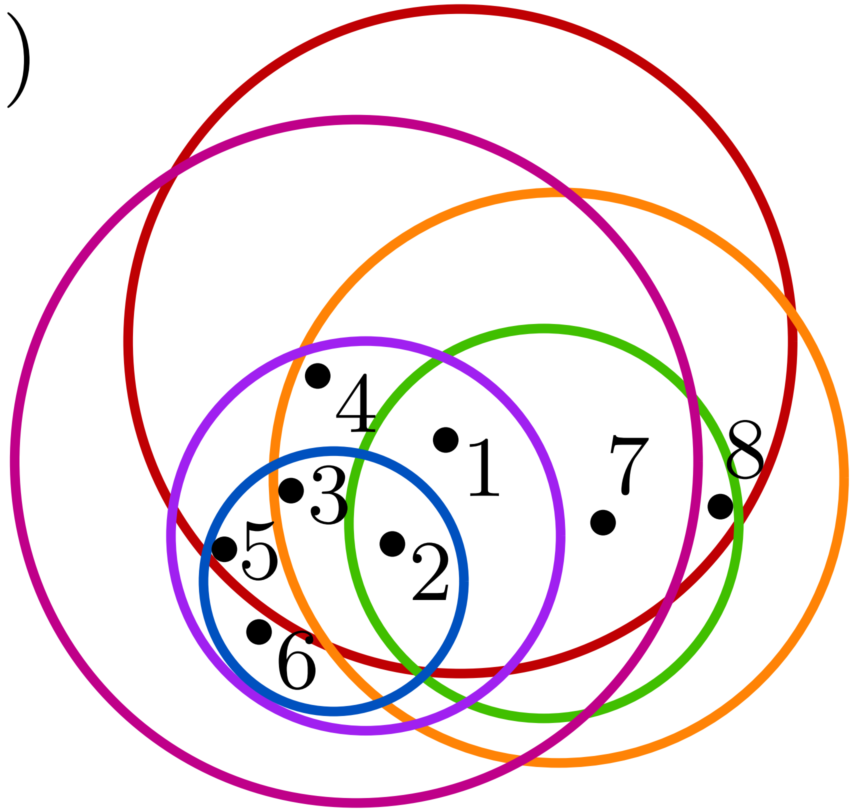
The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$

Let $\varepsilon = 2/3$ and $|\mathcal{R}'| = 2$

sample $\mathcal{R}'$, e.g., $\mathcal{R}' = \{red, pink\}$

$\mathcal{R}'$ covers X, We are done



$W(r)$

$2 \quad red = \{1,2,3,4,5,7,8\}$

$2 \quad green = \{1,2,7,8\}$

$2 \quad orange = \{1,2,3,4,7,8\}$

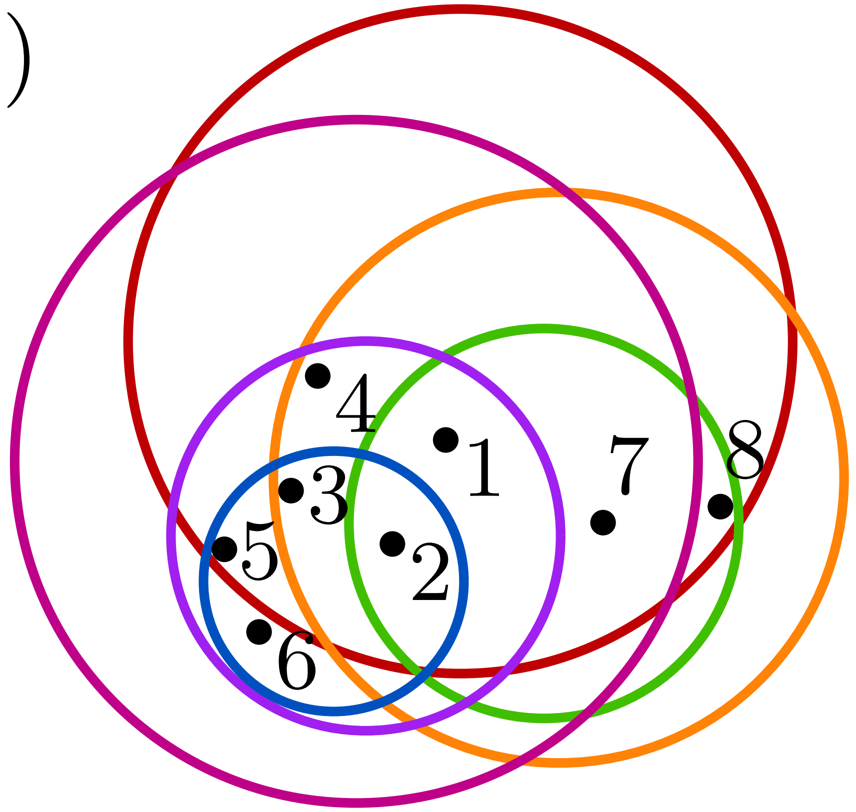
$2 \quad purple = \{1,2,3,4,5,6\}$

$2 \quad blue = \{2,3,5,6\}$

$4 \quad pink = \{1,2,3,4,5,6,7\}$

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



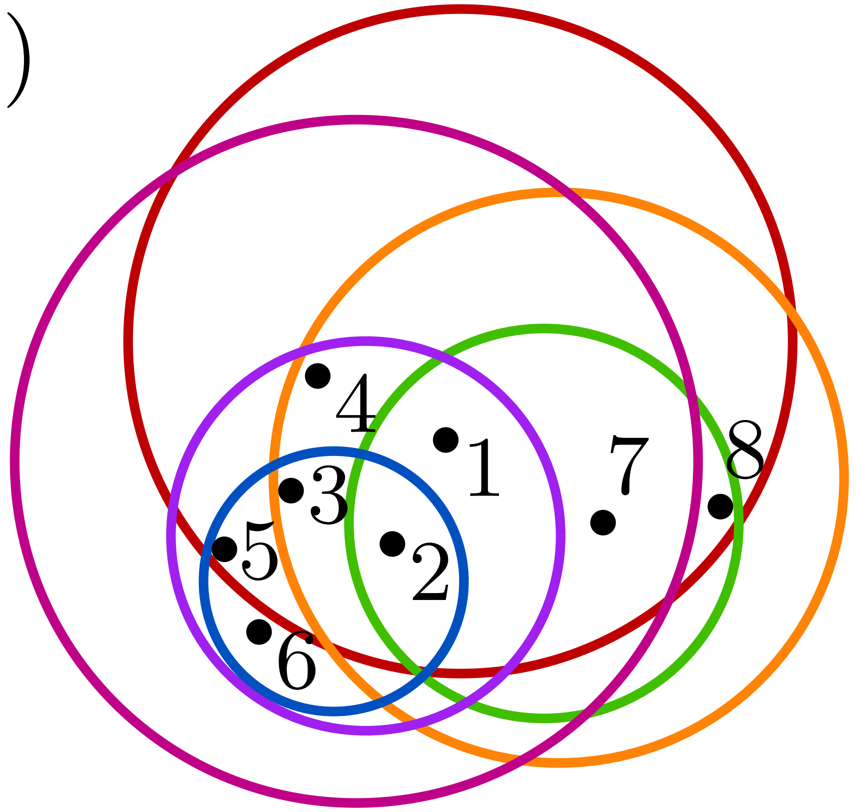
$W(r)$	
2	$red = \{1,2,3,4,5,7,8\}$
2	$green = \{1,2,7,8\}$
2	$orange = \{1,2,3,4,7,8\}$
2	$purple = \{1,2,3,4,5,6\}$
2	$blue = \{2,3,5,6\}$
4	$pink = \{1,2,3,4,5,6,7\}$

intuition:

1. With probability 1/2: $\mathcal{R}'$ is ε-net

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



$W(r)$

$2 \quad red = \{1,2,3,4,5,7,8\}$

$2 \quad green = \{1,2,7,8\}$

$2 \quad orange = \{1,2,3,4,7,8\}$

$2 \quad purple = \{1,2,3,4,5,6\}$

$2 \quad blue = \{2,3,5,6\}$

$4 \quad pink = \{1,2,3,4,5,6,7\}$

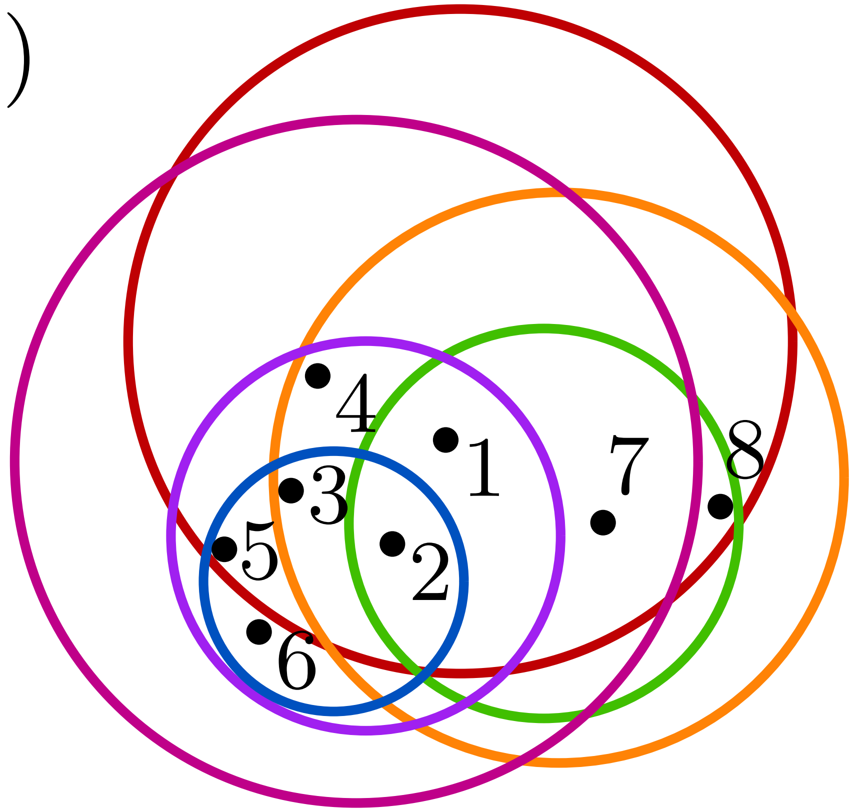
intuition:

1. With probability 1/2: $\mathcal{R}'$ is ε-net

$\rightarrow$ doubling happens often, but to very few ranges

The algorithm

1. sample $\mathcal{R}'$ a random subset of size $\mathcal{O}((\delta^*/\varepsilon)\log(\delta^*/\varepsilon))$
2. While $\mathcal{R}'$ does not cover all points in U
3. let $p \in U$ be the point not covered by $\mathcal{R}'$
4. if $(W(\mathcal{R}_p) < \varepsilon W(\mathcal{R}))$: double all weight of $\mathcal{R}_p$
5. sample new $\mathcal{R}'$
6. return $\mathcal{R}'$



$W(r)$	
2	$red = \{1,2,3,4,5,7,8\}$
2	$green = \{1,2,7,8\}$
2	$orange = \{1,2,3,4,7,8\}$
2	$purple = \{1,2,3,4,5,6\}$
2	$blue = \{2,3,5,6\}$
4	$pink = \{1,2,3,4,5,6,7\}$

intuition:

1. With probability 1/2: $\mathcal{R}'$ is ε-net
   $\rightarrow$ doubling happens often, but to very few ranges

2. $W(\mathcal{R})$ grows slowly, $W(\mathcal{R}_p)$ for uncovered p exponentially $\rightarrow$ eventually p covered

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$

give lower bound on weight of optimal set

compare weight of optimal and $W(\mathcal{R})$ to derive bound on $i \leq 2k \log(m/k)$

conclude that the algorithm terminates successfully

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$

$m = |\mathcal{R}|$

$W_0 = m$ and $W_i =$ the weight after i^{th} doubling

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$

$m = |\mathcal{R}|$

$W_0 = m$ and $W_i =$ the weight after i^{th} doubling

weight of R_p doubled if $W(R_p) < \varepsilon W(R)$

$W_i \leq (1+\varepsilon)W_{i-1} = (1+\varepsilon)^i \cdot m \leq m \cdot e^{\varepsilon i}$

Ingredients of argument

after doubling i times:

- give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$
- give lower bound on weight of optimal set
- compare weight of optimal and $W(\mathcal{R})$ to derive bound on $i \leq 2k \log(m/k)$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set

$t_i(j)$ is the times the weight of j^{th} range in the optimal solution was doubled

the weight of the optimal set at the i^{th} iteration is $\sum_{j=1}^{k} 2^{t_i(j)}$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set

$t_i(j)$ is the times the weight of j^{th} range in the optimal solution was doubled

the weight of the optimal set at the i^{th} iteration is $\sum_{j=1}^{k} 2^{t_i(j)}$

$2^a + 2^b \geq 2 \cdot 2^{\lfloor (a+b)/2 \rfloor}$

To minimize the weight of the optimal set $t_i(1) = t_i(2) = \cdots = t_i(k)$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set

$t_i(j)$ is the times the weight of j^{th} range in the optimal solution was doubled

the weight of the optimal set at the i^{th} iteration is $\sum_{j=1}^{k} 2^{t_i(j)}$

$2^a + 2^b \geq 2 \cdot 2^{\lfloor (a+b)/2 \rfloor}$

To minimize the weight of the optimal set $t_i(1) = t_i(2) = \cdots = t_i(k)$

minimal weight of the optimal set $\geq k 2^{\lfloor i/k \rfloor}$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set $\geq k2^{\lfloor i/k \rfloor}$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set $\geq k2^{\lfloor i/k \rfloor}$

compare weight of optimal and $W(\mathcal{R})$ to derive upper bound on i

optimal set $\subset \mathcal{R}$

$\Rightarrow k2^{\lfloor i/k \rfloor} \leq m \cdot e^{\varepsilon i} = m \cdot e^{(i/k)/4}$, since $\varepsilon = \frac{1}{4k}$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set $\geq k2^{\lfloor i/k \rfloor}$

compare weight of optimal and $W(\mathcal{R})$ to derive upper bound on i

optimal set $\subset \mathcal{R}$

$\Rightarrow k2^{\lfloor i/k \rfloor} \leq m \cdot e^{\varepsilon i} = m \cdot e^{(i/k)/4}$, since $\varepsilon = \frac{1}{4k}$

$\Rightarrow \left(\frac{2}{e^{1/4}}\right)^{i/k} \leq m/k$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set $\geq k 2^{\lfloor i/k \rfloor}$

compare weight of optimal and $W(\mathcal{R})$ to derive upper bound on i

optimal set $\subset \mathcal{R}$

$$\Rightarrow k 2^{\lfloor i/k \rfloor} \leq m \cdot e^{\varepsilon i} = m \cdot e^{(i/k)/4}, \text{ since } \varepsilon = \frac{1}{4k}$$

$$\Rightarrow \left(\frac{2}{e^{1/4}}\right)^{i/k} \leq m/k$$

$$\Rightarrow i/k \leq \log(m/k)/\log\left(\frac{2}{e^{1/4}}\right) \leq 2\log(m/k)$$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set $\geq k 2^{\lfloor i/k \rfloor}$

compare weight of optimal and $W(\mathcal{R})$ to derive upper bound on i

optimal set $\subset \mathcal{R}$

$$\Rightarrow k 2^{\lfloor i/k \rfloor} \leq m \cdot e^{\varepsilon i} = m \cdot e^{(i/k)/4}, \text{ since } \varepsilon = \frac{1}{4k}$$

$$\Rightarrow \left(\frac{2}{e^{1/4}}\right)^{i/k} \leq m/k$$

$$\Rightarrow i/k \leq \log(m/k) / \log\left(\frac{2}{e^{1/4}}\right) \leq 2 \log(m/k)$$

$$\Rightarrow i \leq 2k \log(m/k)$$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set $\geq k2^{\lfloor i/k \rfloor}$

compare weight of optimal and $W(\mathcal{R})$ to derive bound on $i \leq 2k \log(m/k)$

conclude that the algorithm terminates successfully

$\mathcal{R}'$ is ε-net with probability $\geq 1/2$:

expected # iterations $\leq 4k \log(m/k)$

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set $\geq k 2^{\lfloor i/k \rfloor}$

compare weight of optimal and $W(\mathcal{R})$ to derive bound on $i \leq 2k \log(m/k)$

conclude that the algorithm terminates successfully

$\mathcal{R}'$ is ε-net with probability $\geq 1/2$:

expected # iterations $\leq 4k \log(m/k)$

iterations $\leq 8k \log(m/k)$ with high prob. (Chernoff)

Ingredients of argument

after doubling i times:

give upper bound on $W(\mathcal{R})$ $\leq m \cdot e^{\varepsilon i}$

give lower bound on weight of optimal set $\geq k 2^{\lfloor i/k \rfloor}$

compare weight of optimal and $W(\mathcal{R})$ to derive bound on $i \leq 2k \log(m/k)$

conclude that the algorithm terminates successfully

$\mathcal{R}'$ is ε-net with probability $\geq 1/2$:

expected # iterations $\leq 4k \log(m/k)$

iterations $\leq 8k \log(m/k)$ with high prob. (Chernoff)

If we need more iterations, we can assume k was guessed too small, and we double k

Quiz

How many values do we test for k?

A $\log n = \log |X|$

B $\log m = \log |\mathcal{R}|$

C $\min(\log m, \log n)$

Quiz

How many values do we test for k?

A $\log n = \log|X|$

B $\log m = \log|\mathcal{R}|$

C $\boxed{\min(\log m, \log n)}$

Summary for Algorithm

Given (X, R) with $n = |U|$, $m = |\mathcal{R}|$, and dual shattering dimension δ^*, we can compute a set cover which uses $\mathcal{O}(\delta^* k \cdot \log(\delta^* k))$ sets where k is the number of sets used by the optimal solution. The run time is $O((m + n\delta^* k \cdot \log(\delta^* k)) \cdot \log(m/k) \cdot \log(n))$ with high probability assuming we can decide if a point is inside a range in constant time.

Application to the art gallery problem

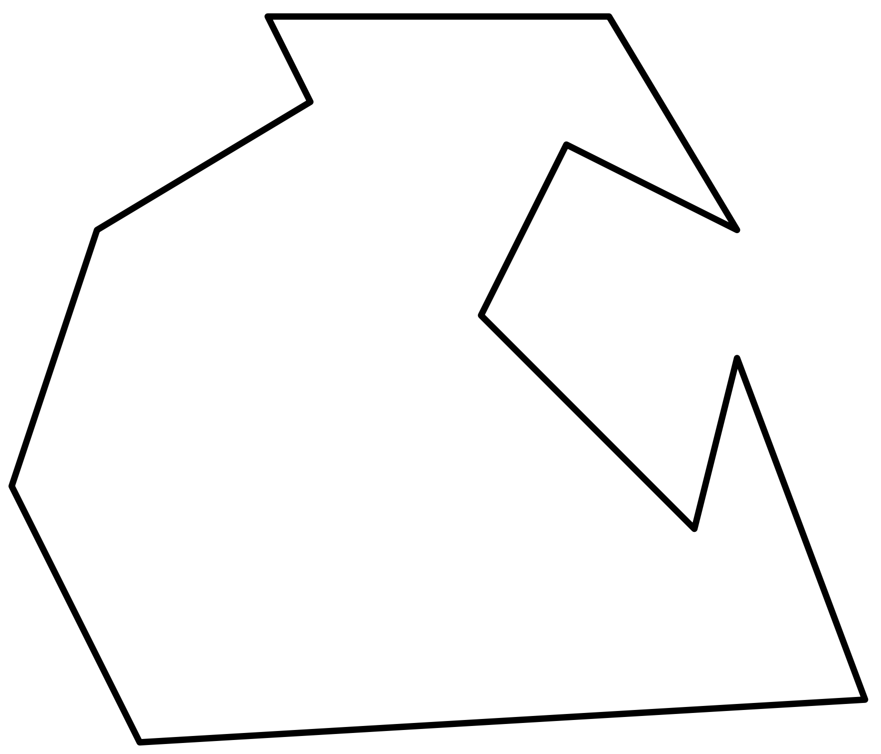
covering simple polygons with guards

The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$

a guard at p sees all of $\mathcal{V}_P(p)$

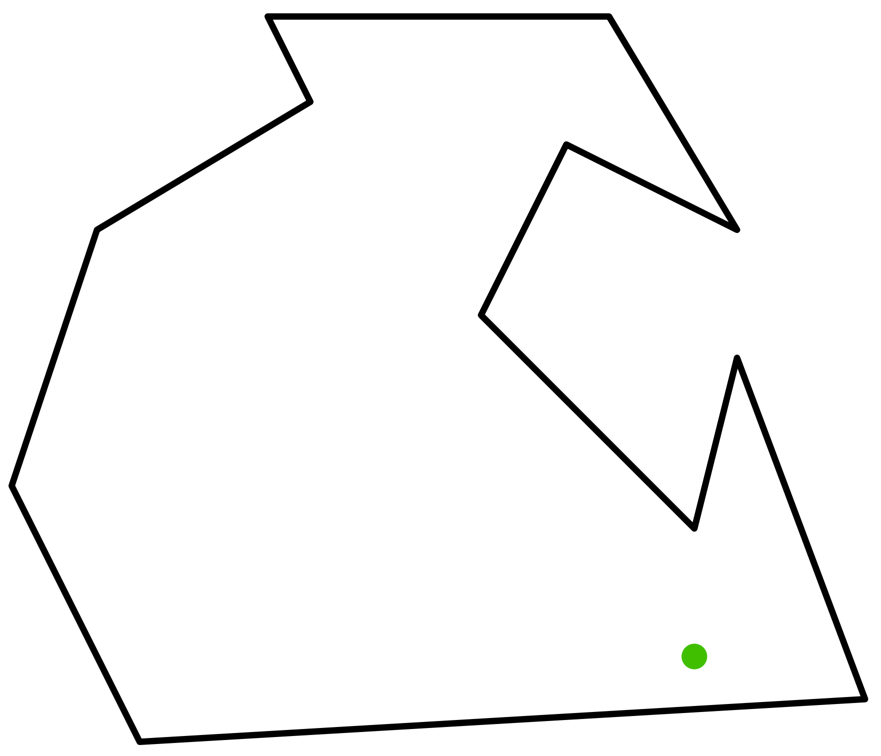
Free placement of point p



The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

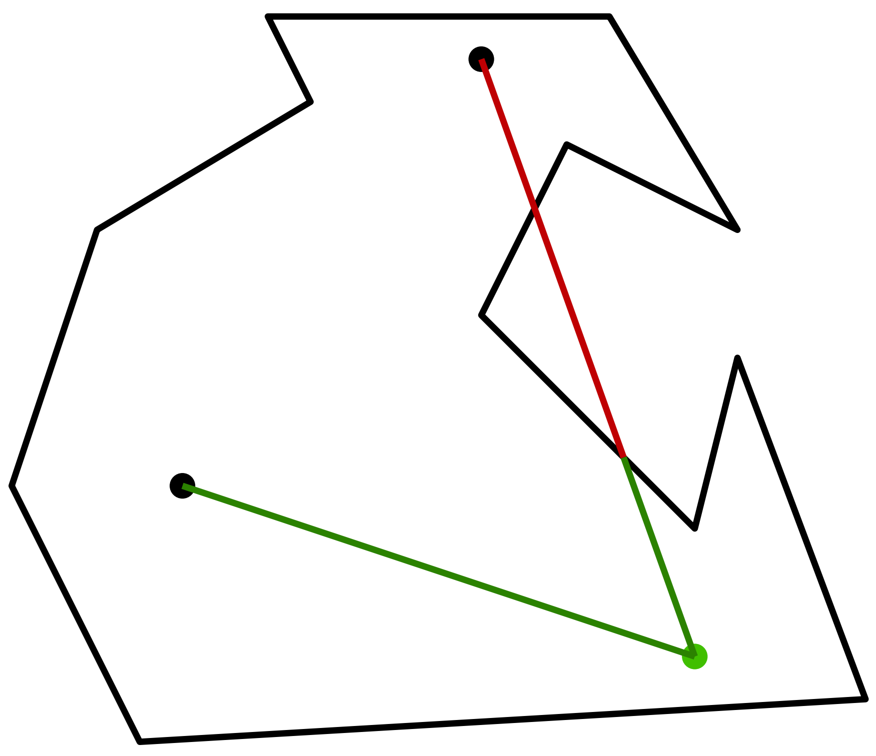
Free placement of point p



The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

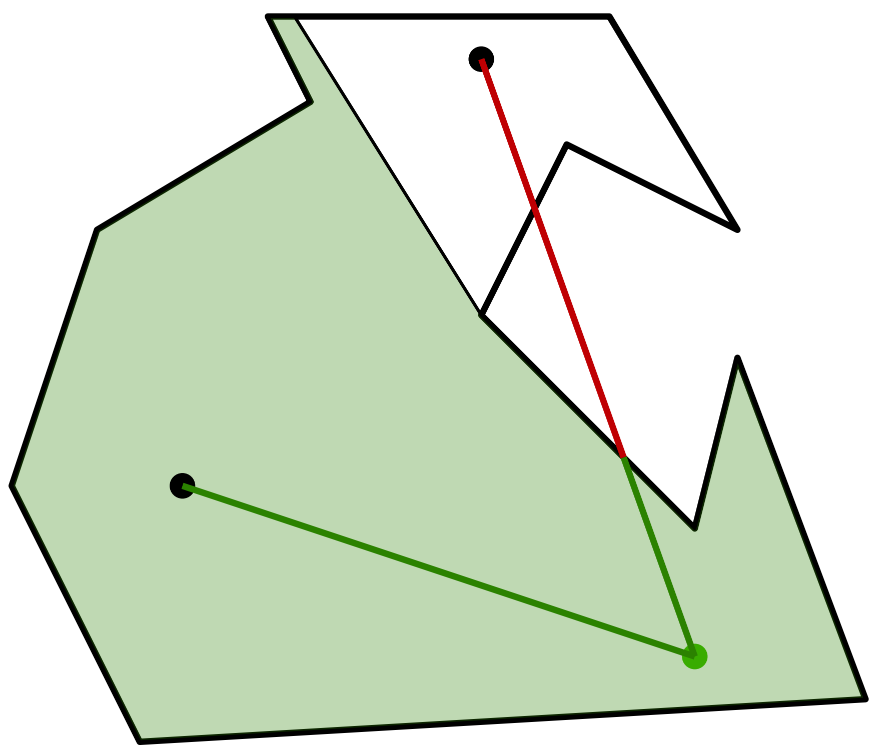
Free placement of point p



The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

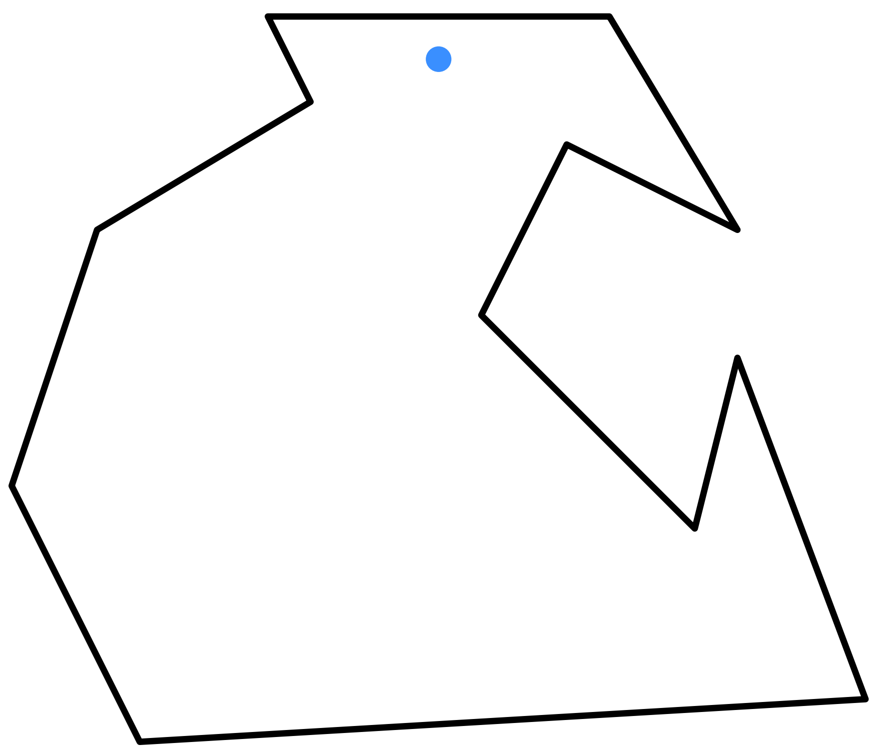
Free placement of point p



The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

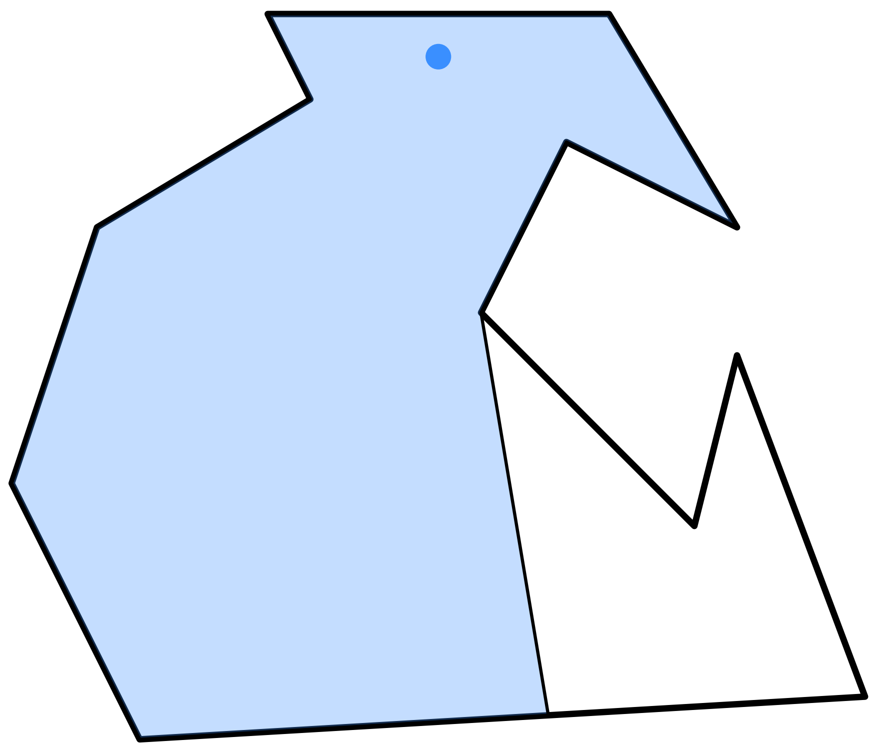
Free placement of point p



The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

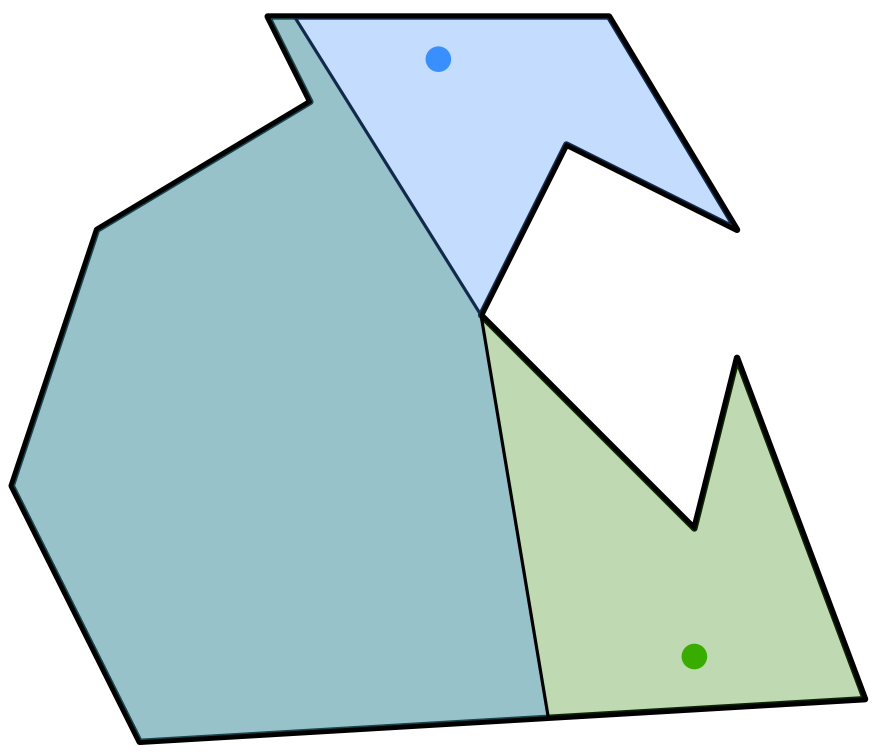
Free placement of point p



The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

Free placement of point p



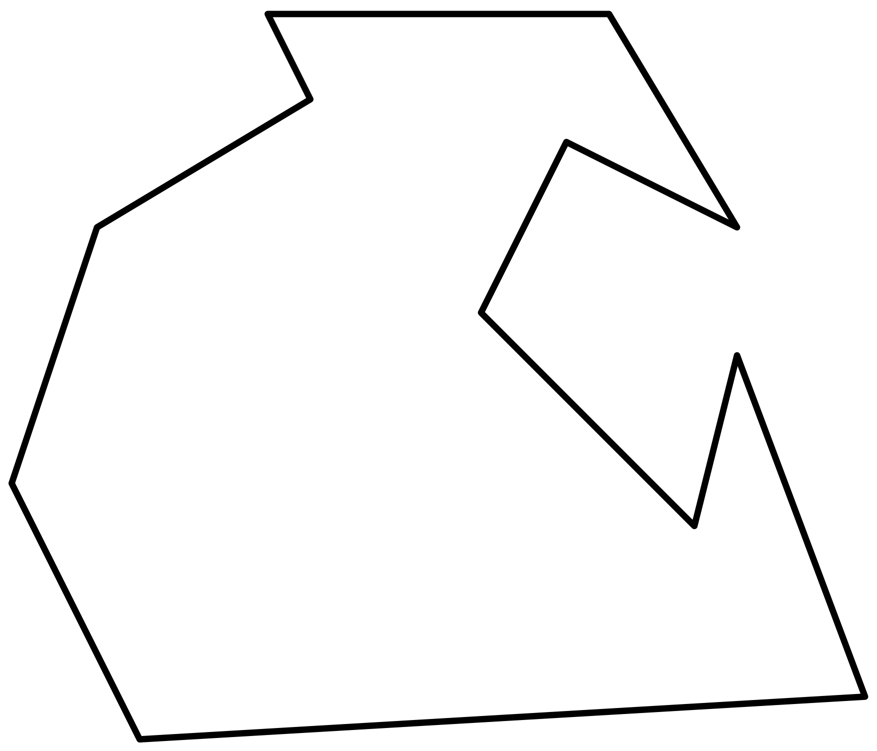
The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$

a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset



The art gallery problem: covering a polygon

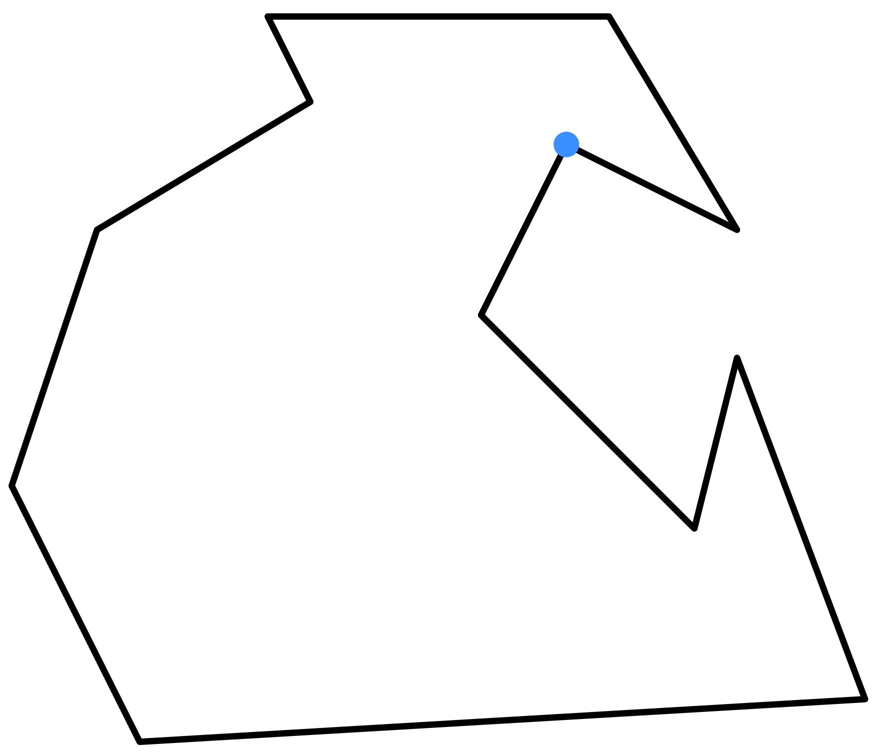
Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$

a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



The art gallery problem: covering a polygon

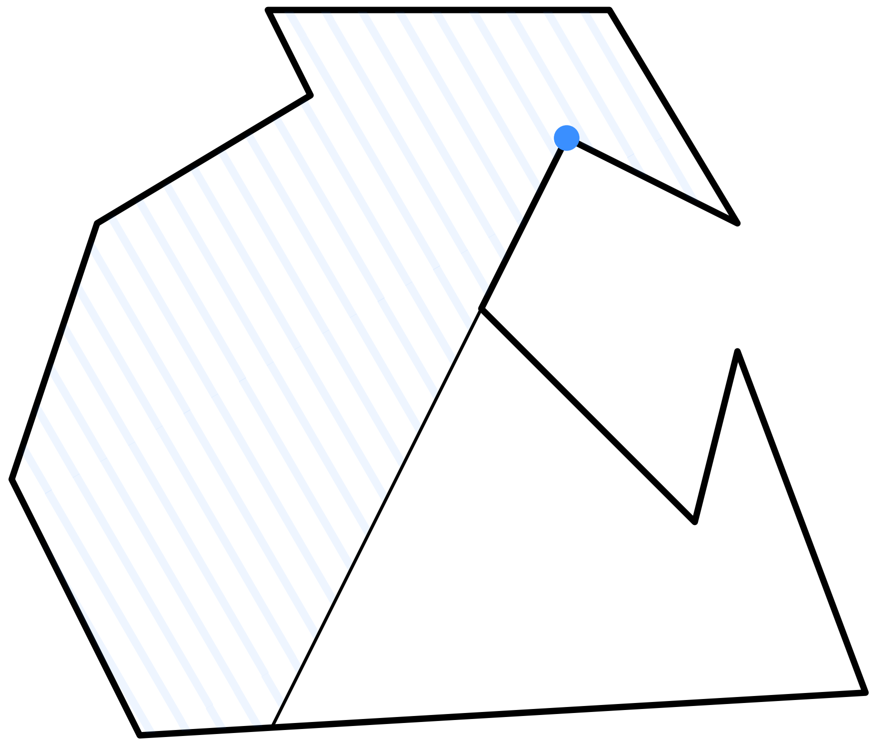
Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$

a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



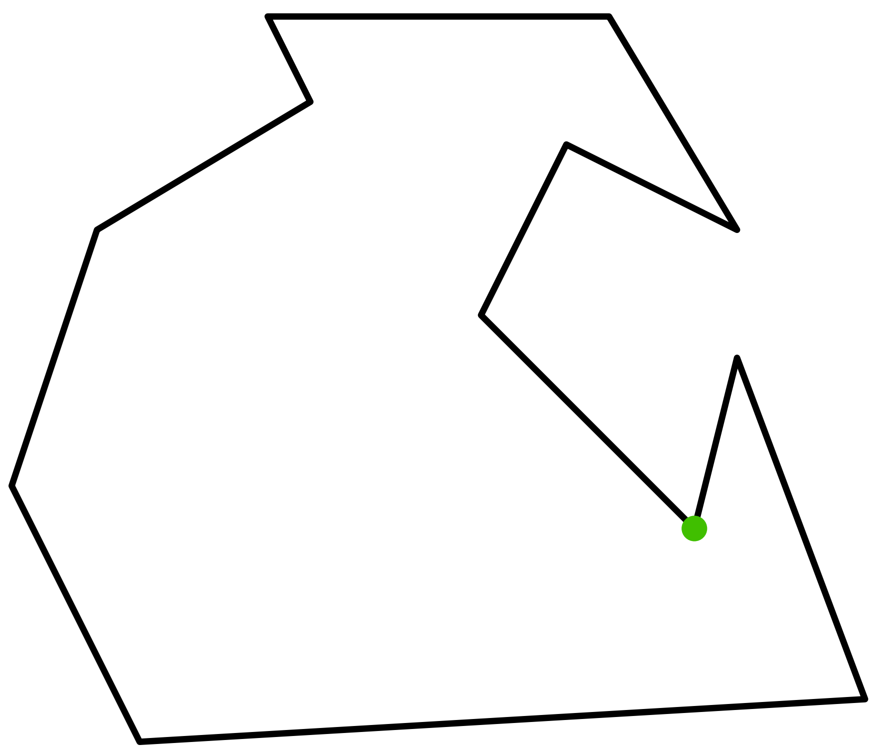
The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



The art gallery problem: covering a polygon

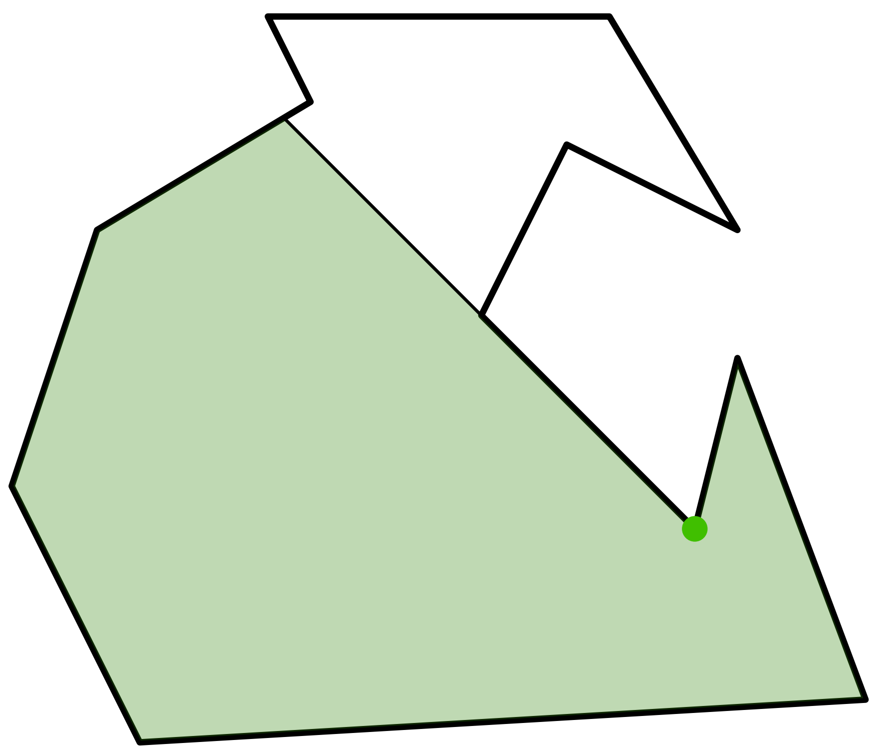
Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$

a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



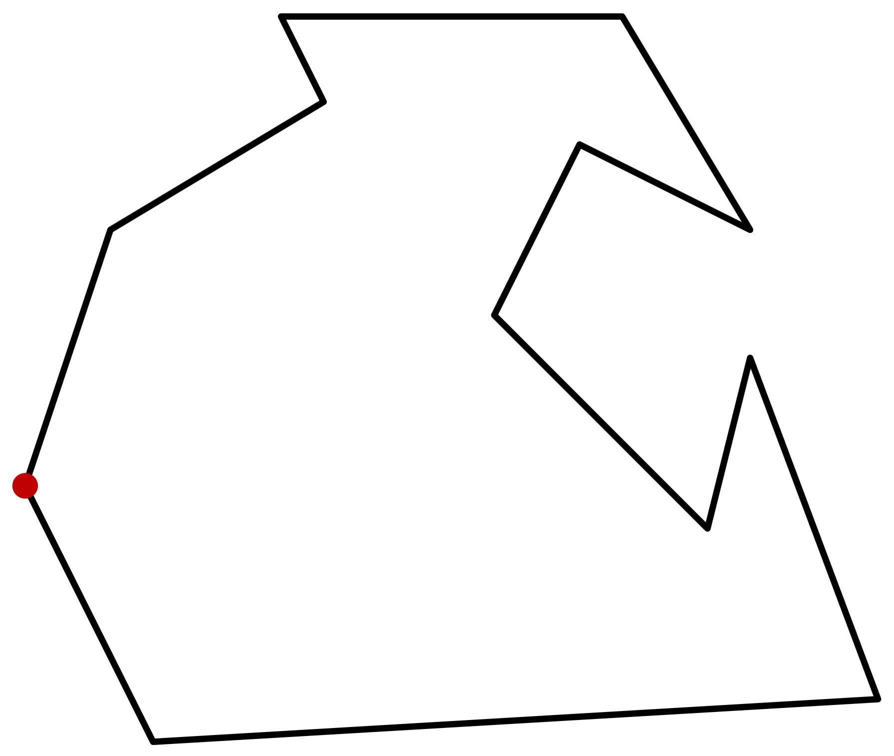
The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



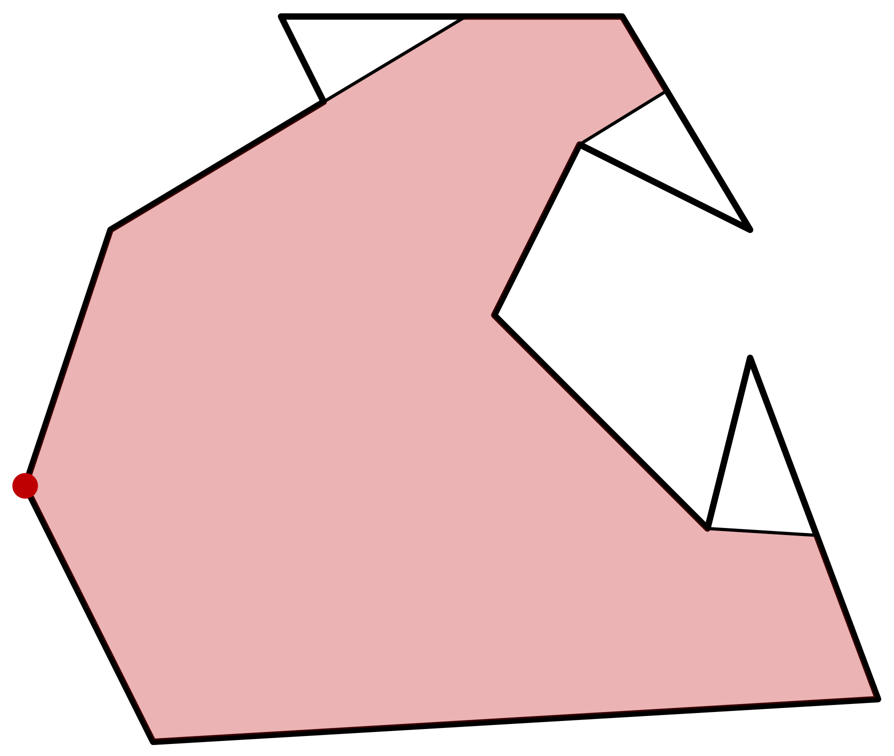
The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



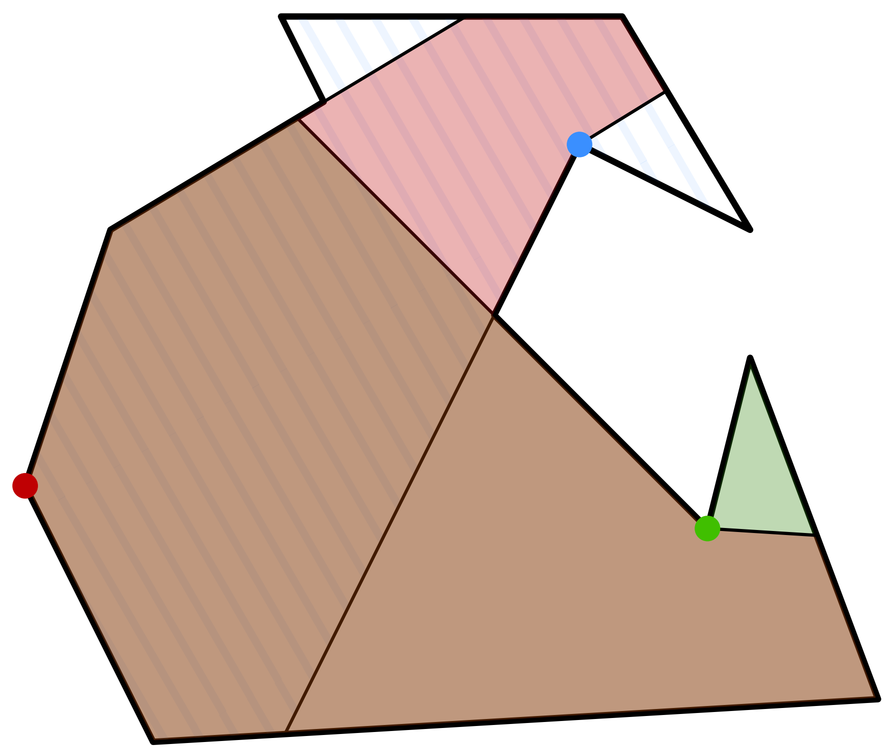
The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



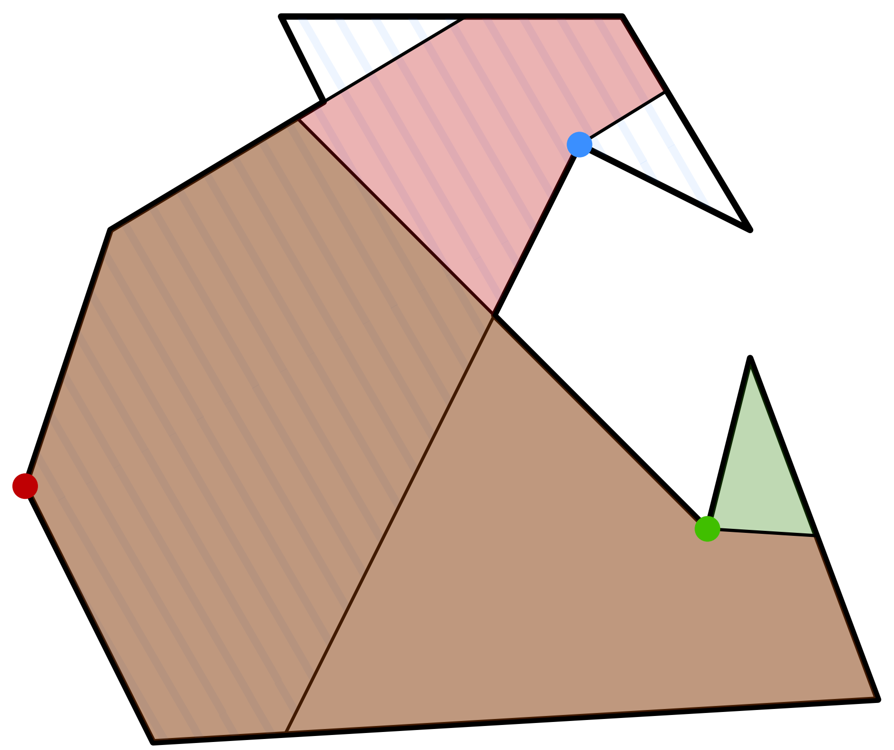
The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



Question: Which sets cover the polygon?

$G_1 = \{blue\}$

$G_2 = \{blue, red\}$

$G_3 = \{blue, green\}$

$G_4 = \{blue, green, red\}$

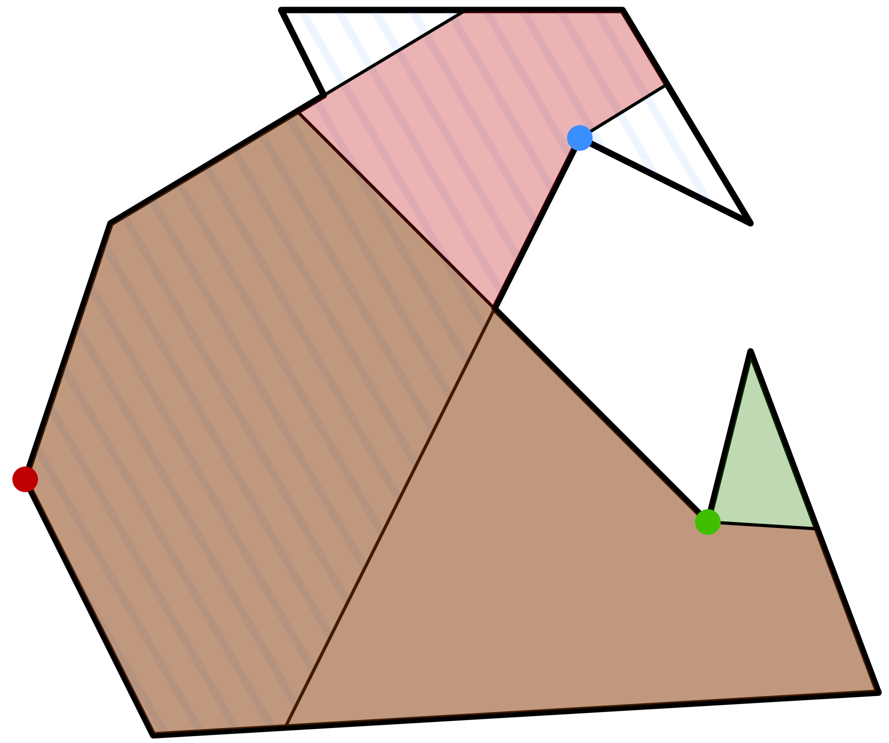
The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



Question: Which sets cover the polygon?

$G_1 = \{blue\}$

$G_2 = \{blue, red\}$

$G_3 = \{blue, green\}$

$G_4 = \{blue, green, red\}$

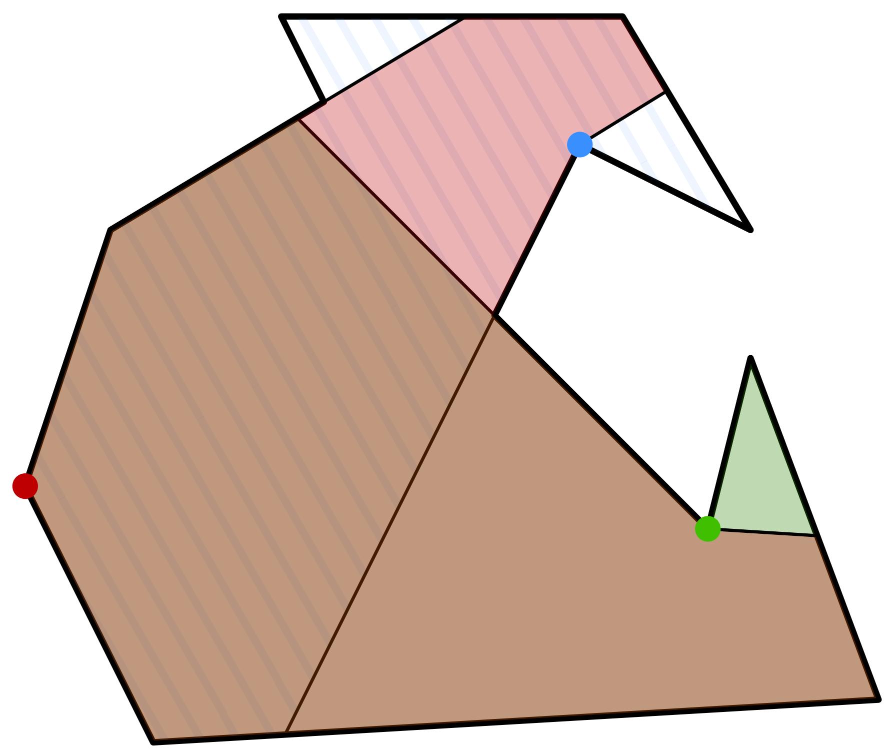
The art gallery problem: covering a polygon

Point p covers $\mathcal{V}_P(p) = \{q \mid q \in P, pq \subseteq P\}$ a guard at p sees all of $\mathcal{V}_P(p)$

Infinity many points in P

Restrict possible placement of p to a finite subset

Restrict placement of p to vertices of P



Question: Which sets cover the polygon?

$G_1 = \{blue\}$

$G_2 = \{blue, red\}$

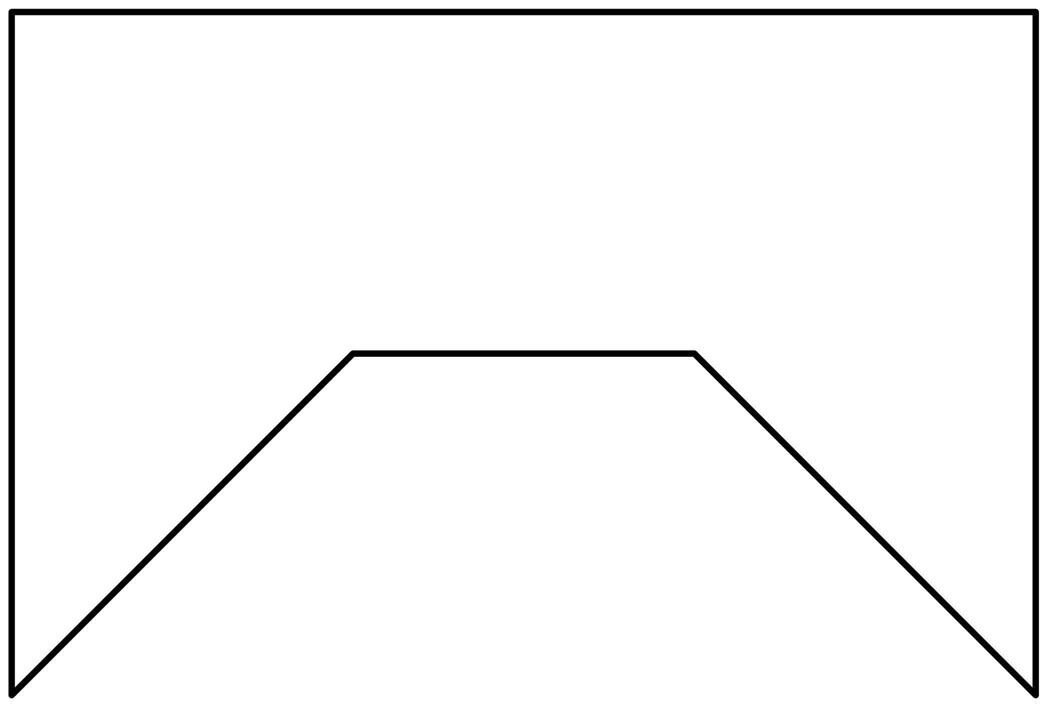
$G_3 = \{blue, green\}$

$G_4 = \{blue, green, red\}$

goal: cover with as few $\mathcal{V}_P(p)$ as possible

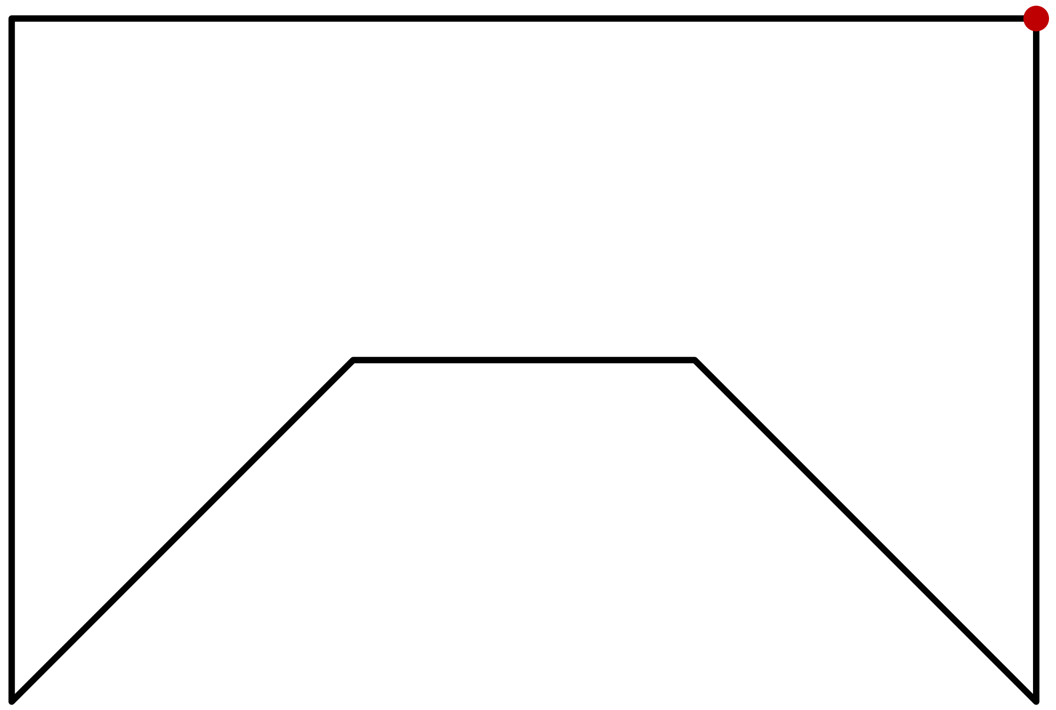
Compute range space

1. Calculate all visibility polygons for the vertices of P



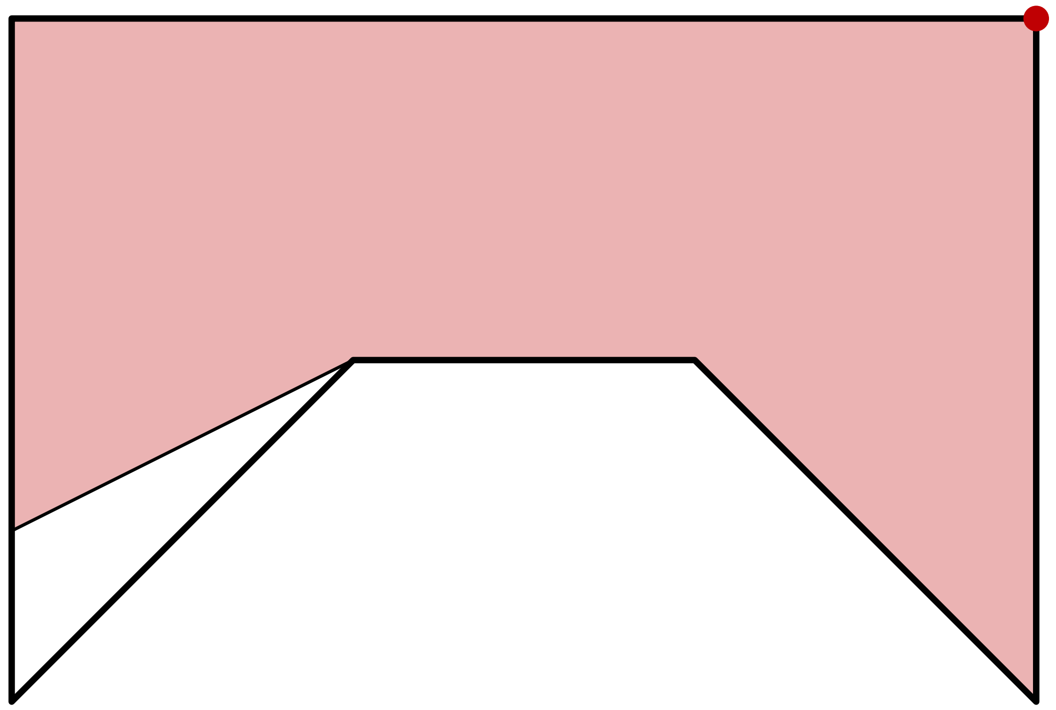
Compute range space

1. Calculate all visibility polygons for the vertices of P



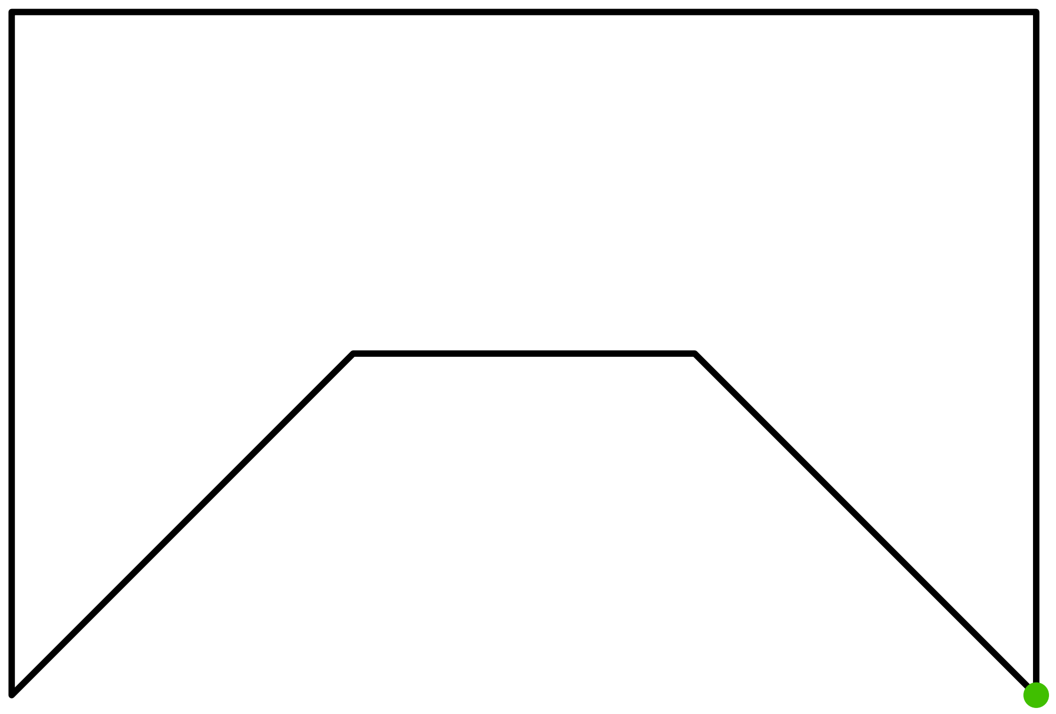
Compute range space

1. Calculate all visibility polygons for the vertices of P



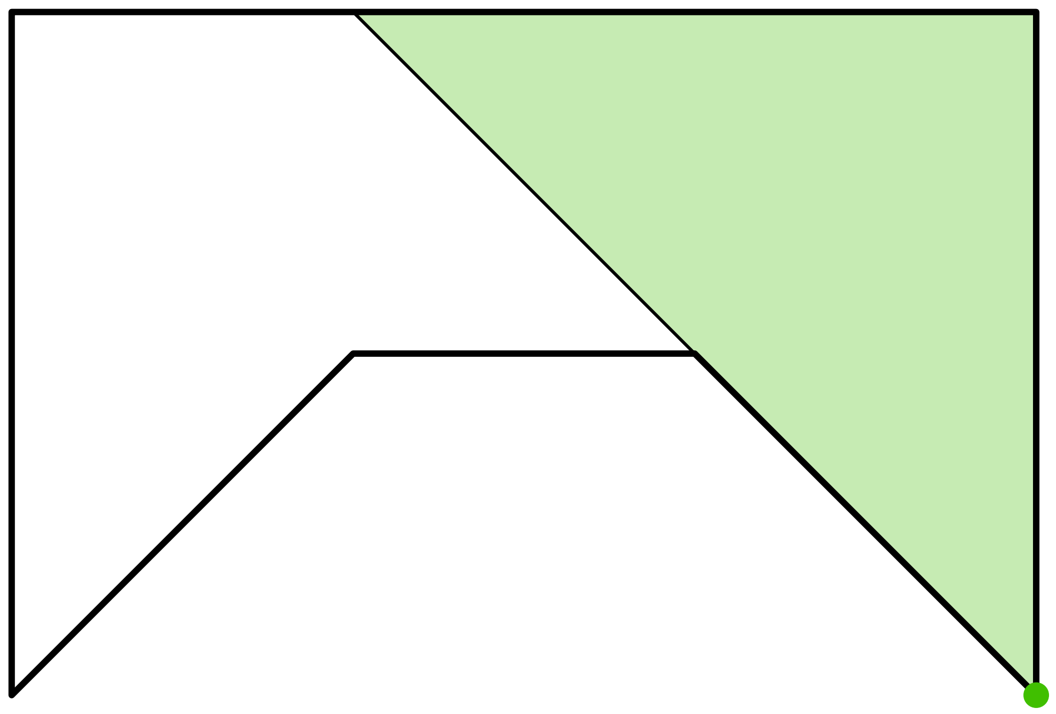
Compute range space

1. Calculate all visibility polygons for the vertices of P



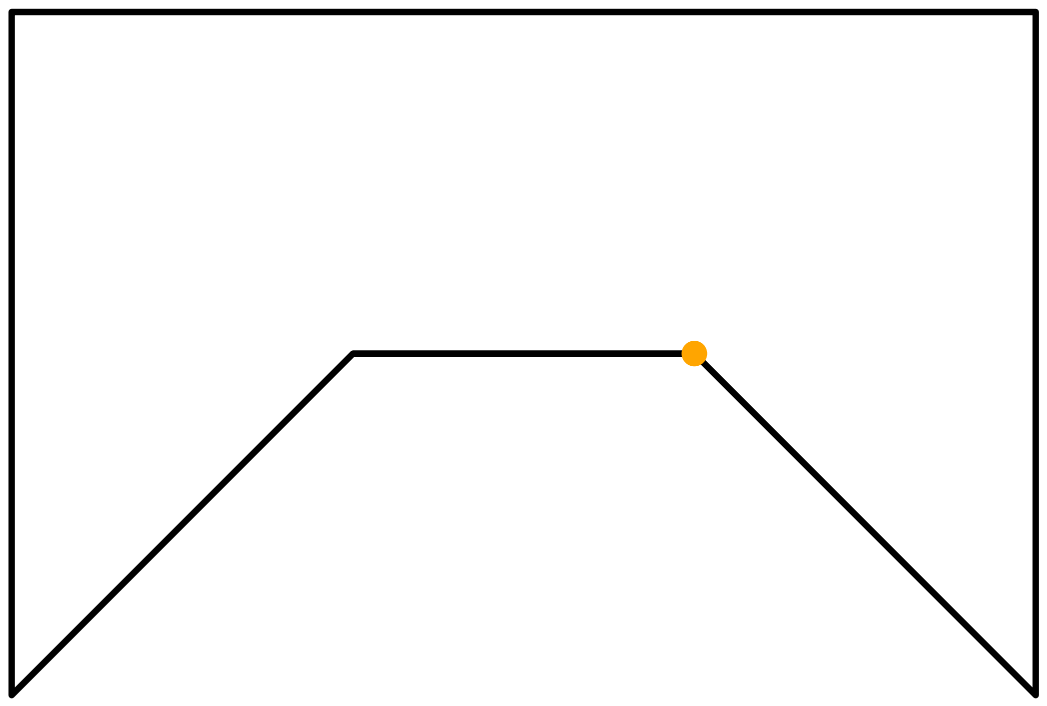
Compute range space

1. Calculate all visibility polygons for the vertices of P



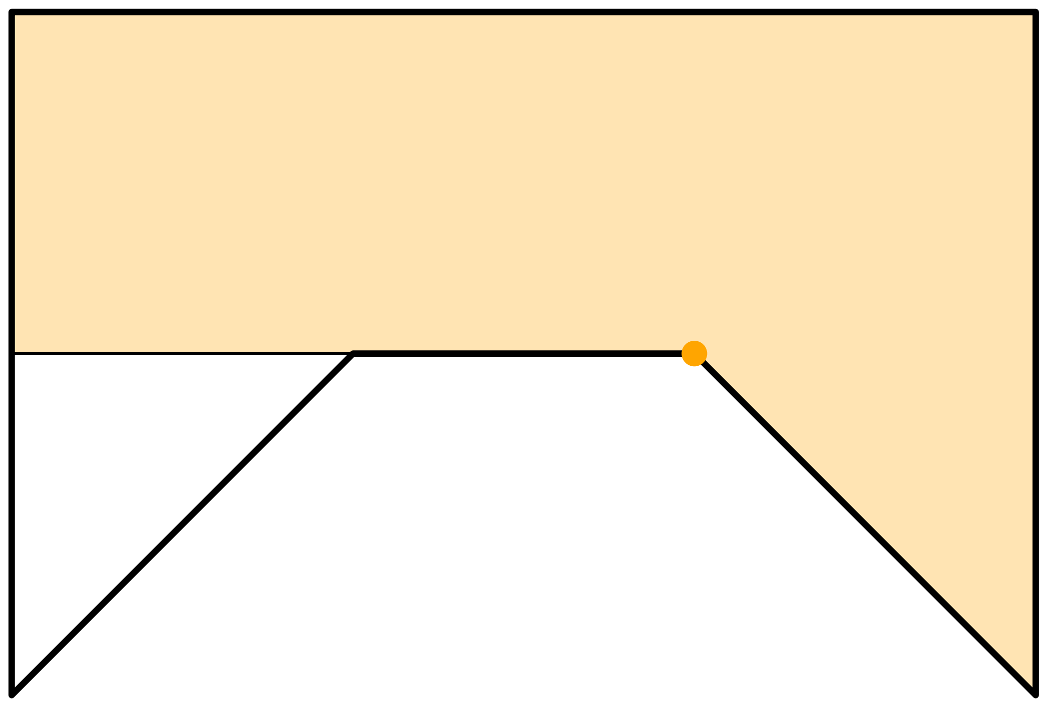
Compute range space

1. Calculate all visibility polygons for the vertices of P



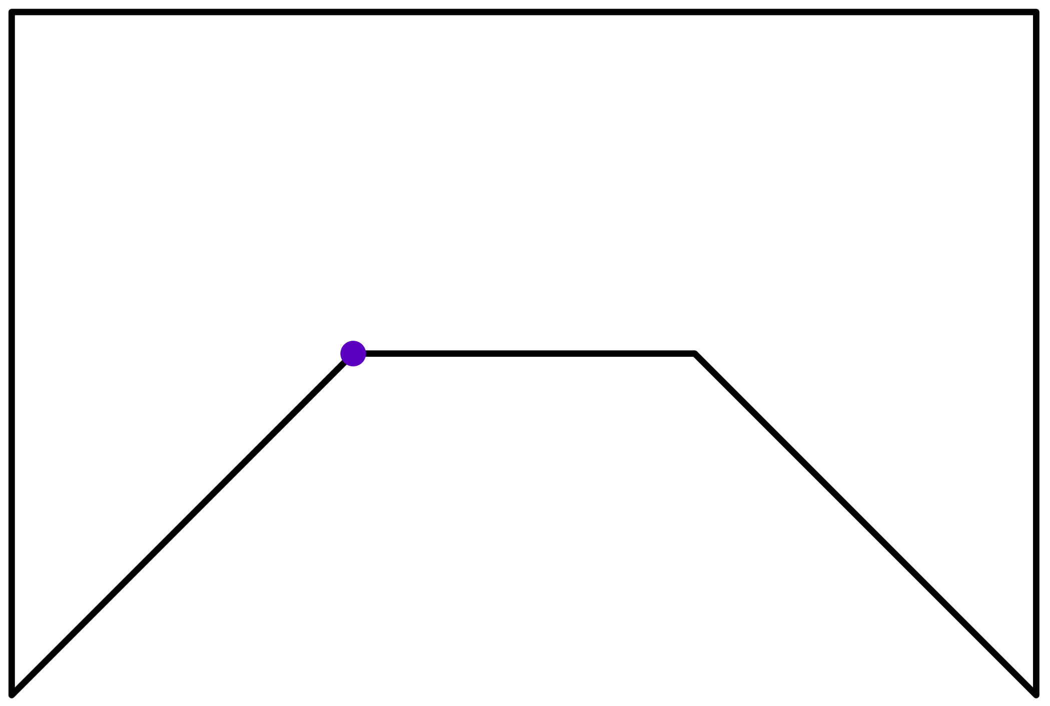
Compute range space

1. Calculate all visibility polygons for the vertices of P



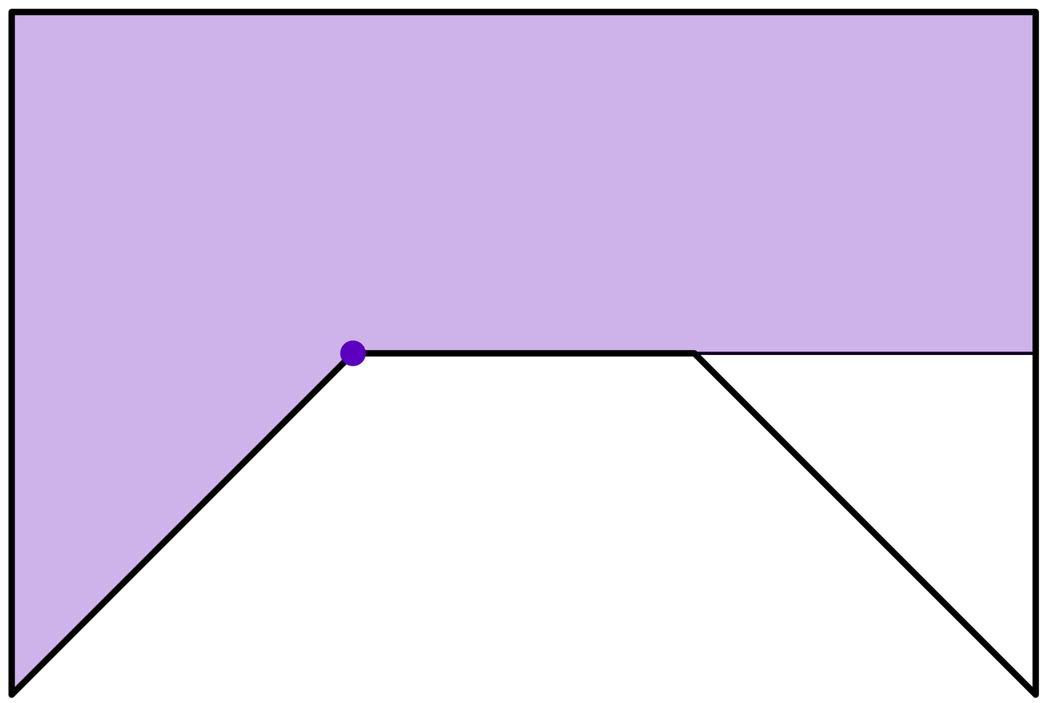
Compute range space

1. Calculate all visibility polygons for the vertices of P



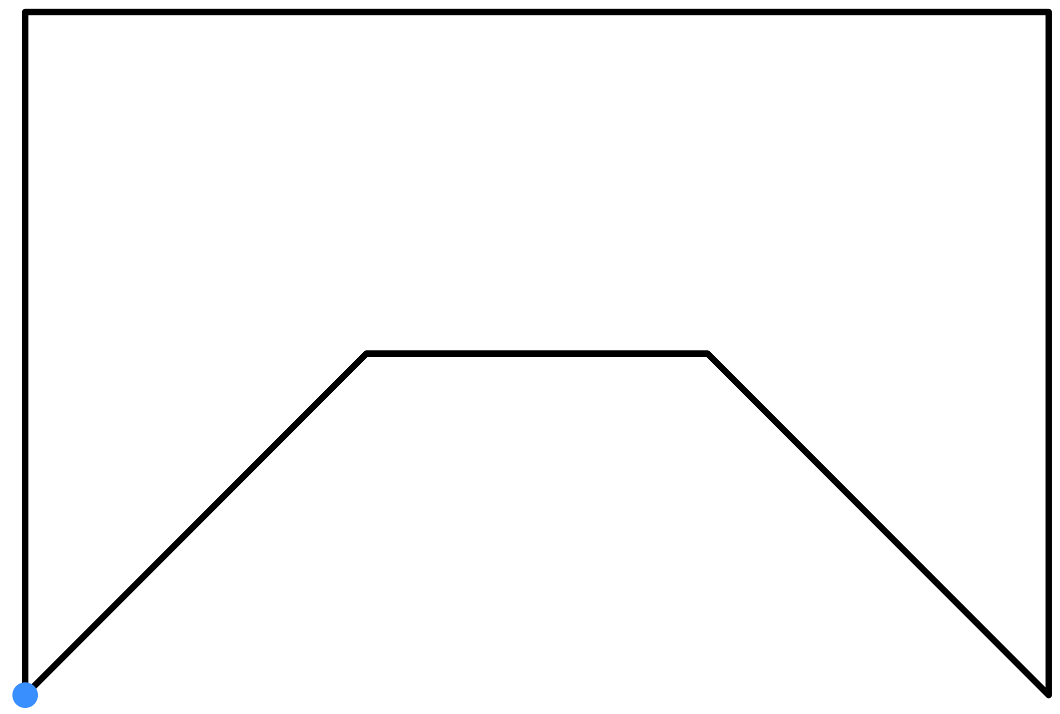
Compute range space

1. Calculate all visibility polygons for the vertices of P



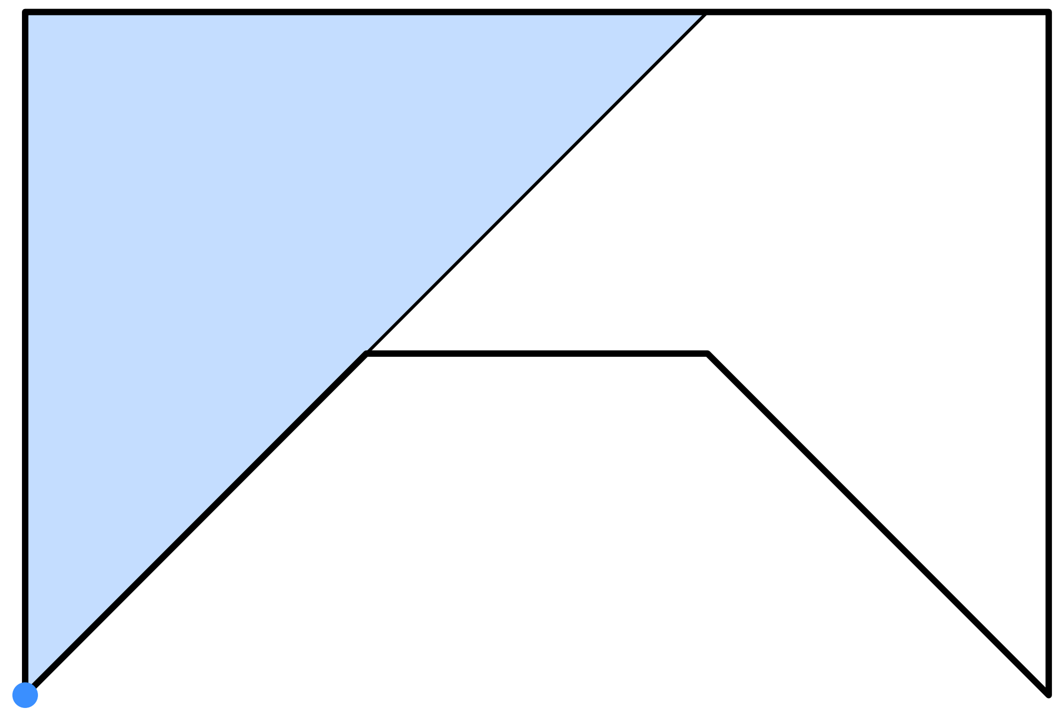
Compute range space

1. Calculate all visibility polygons for the vertices of P



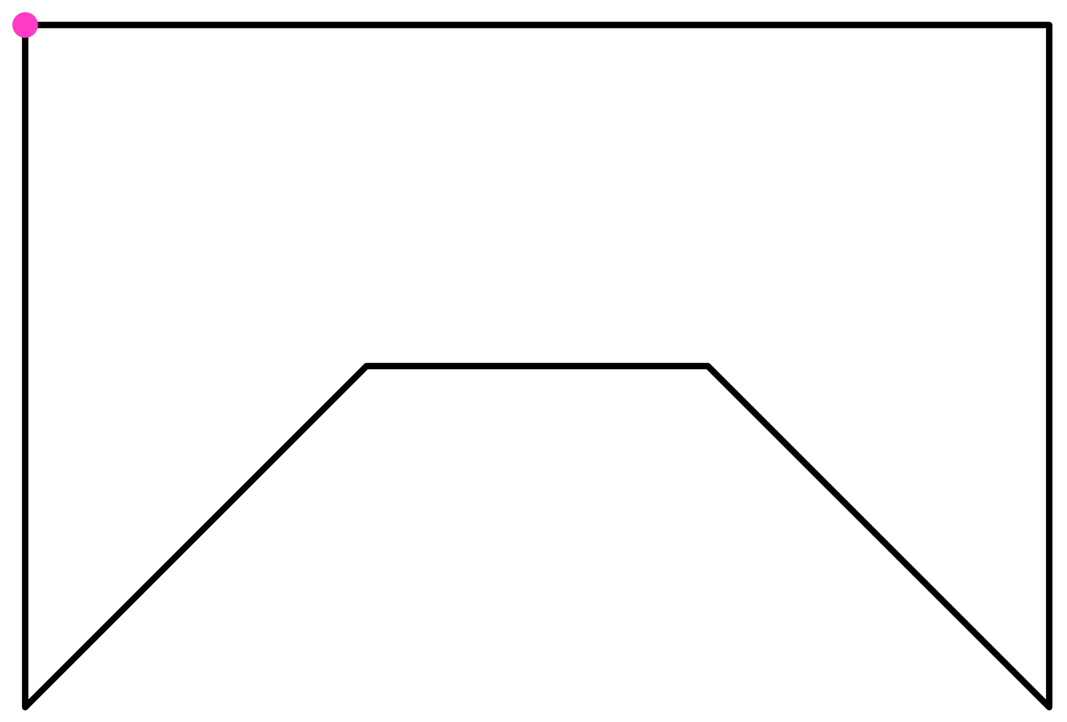
Compute range space

1. Calculate all visibility polygons for the vertices of P



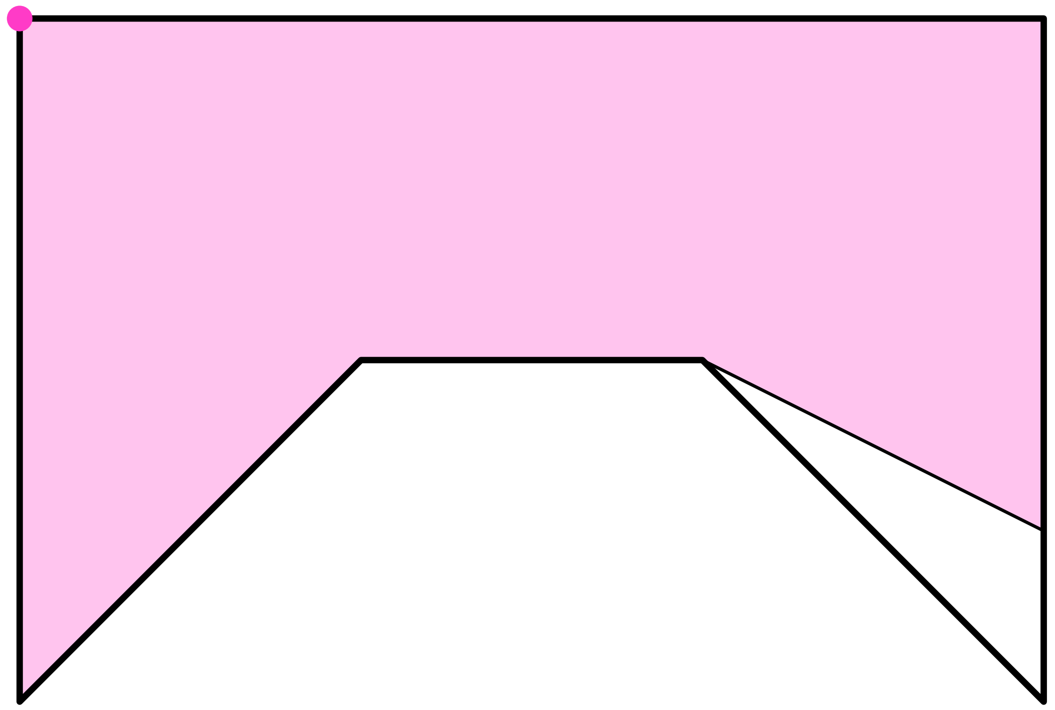
Compute range space

1. Calculate all visibility polygons for the vertices of P



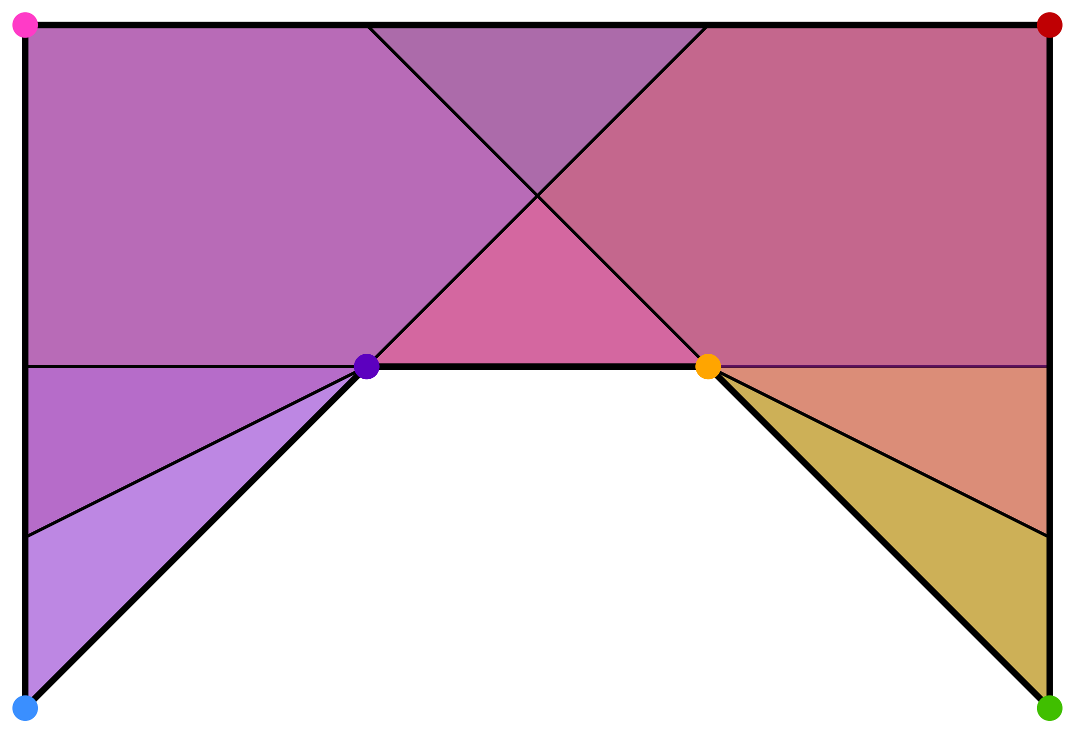
Compute range space

1. Calculate all visibility polygons for the vertices of P



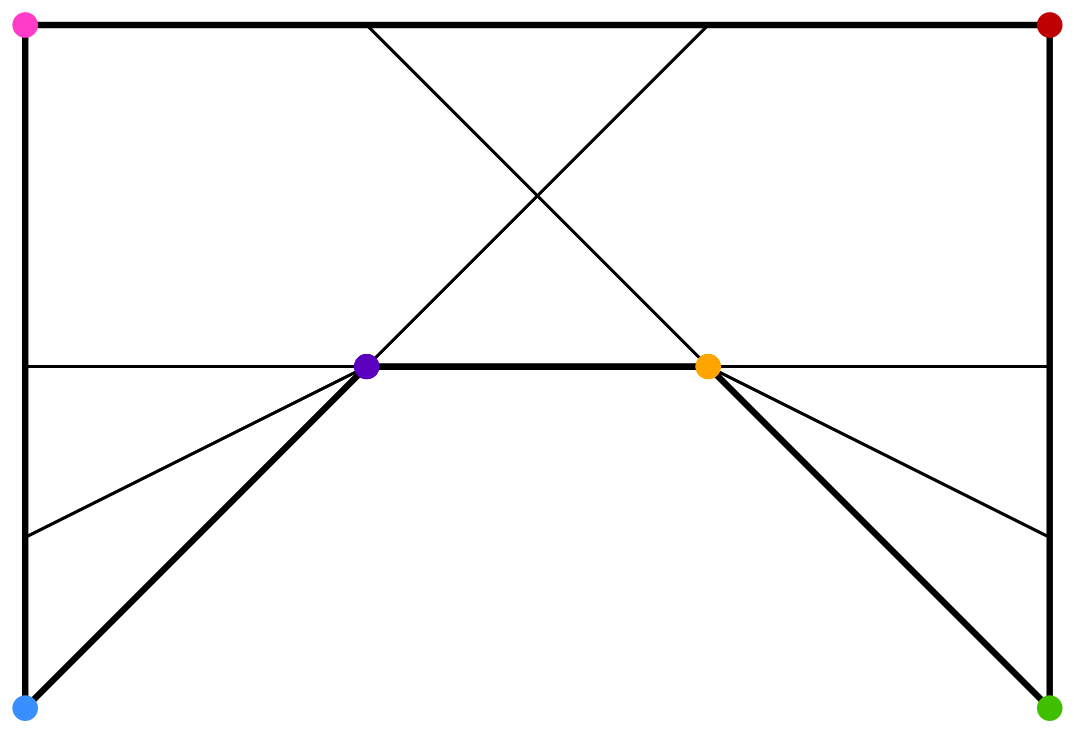
Compute range space

1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons



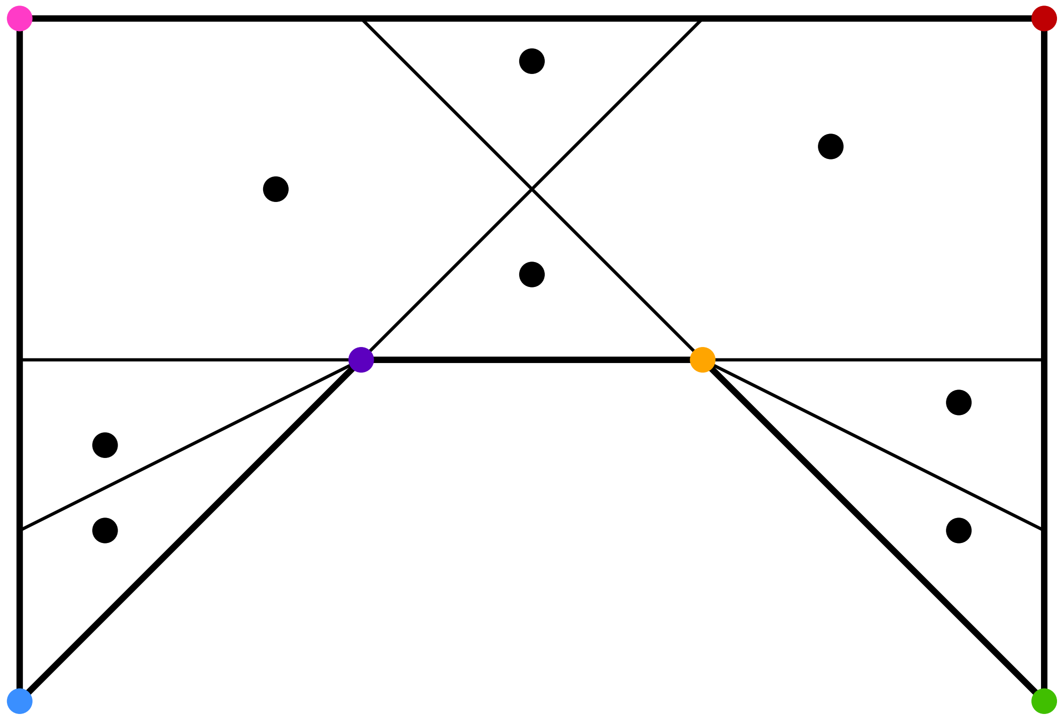
Compute range space

1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons



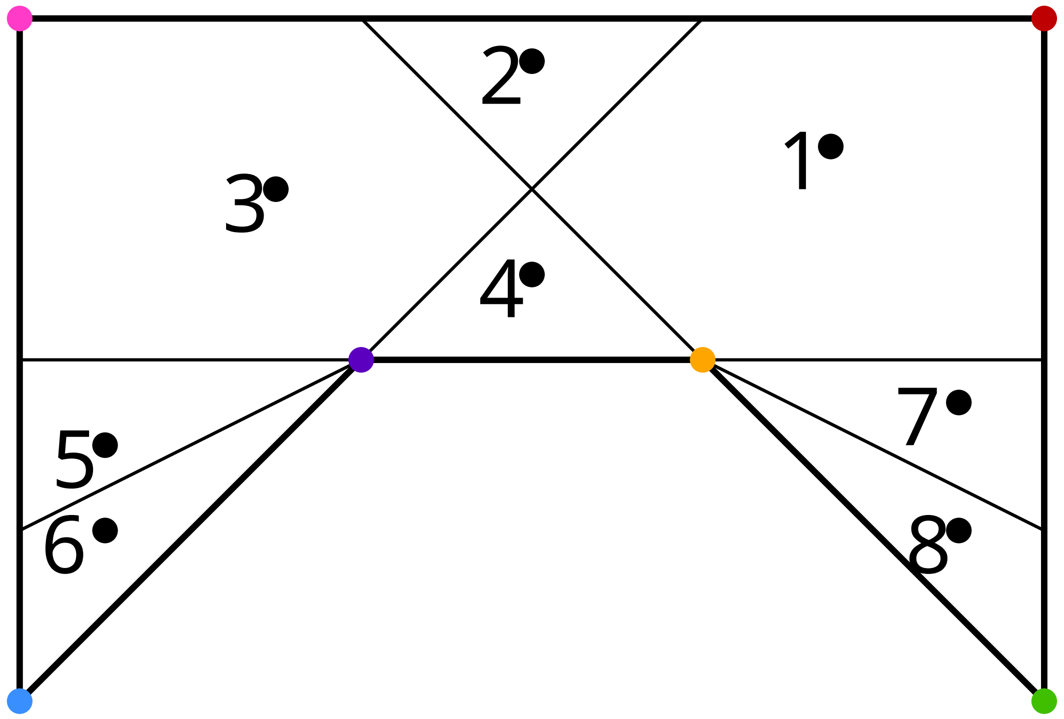
Compute range space

1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons
3. Place a point in each face of the arrangement (or simply take set of faces)



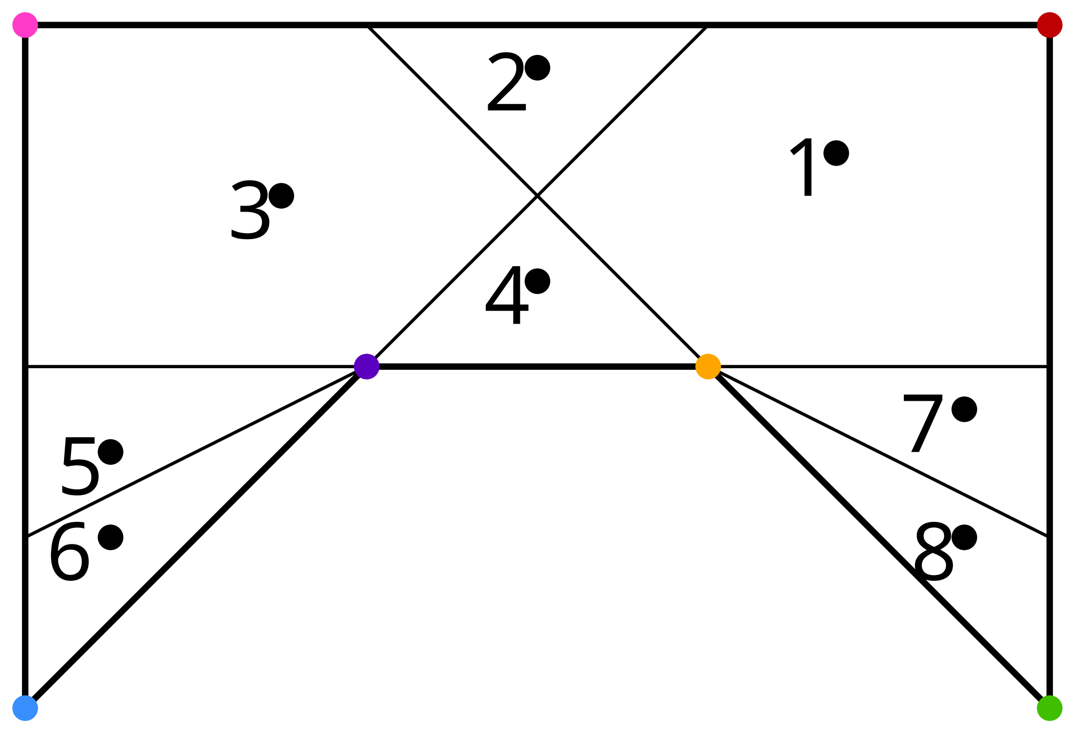
Compute range space

1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons
3. Place a point in each face of the arrangement (or simply take set of faces)
4. Label each point/faces for clarity



Compute range space

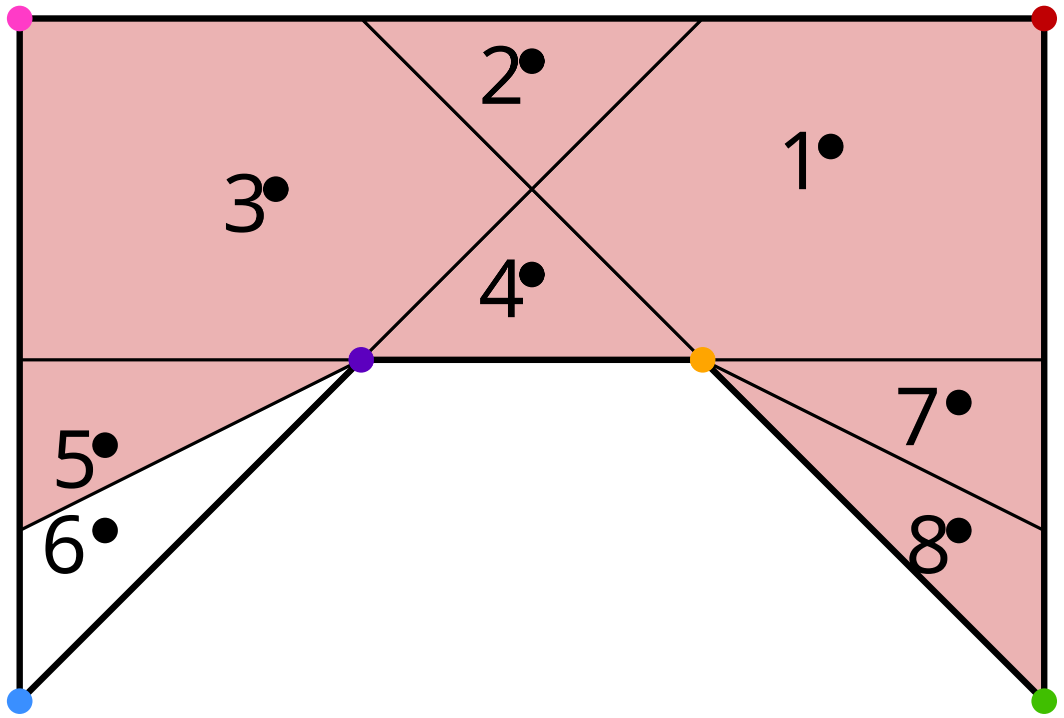
1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons
3. Place a point in each face of the arrangement (or simply take set of faces)
4. Label each point/faces for clarity $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$



Compute range space

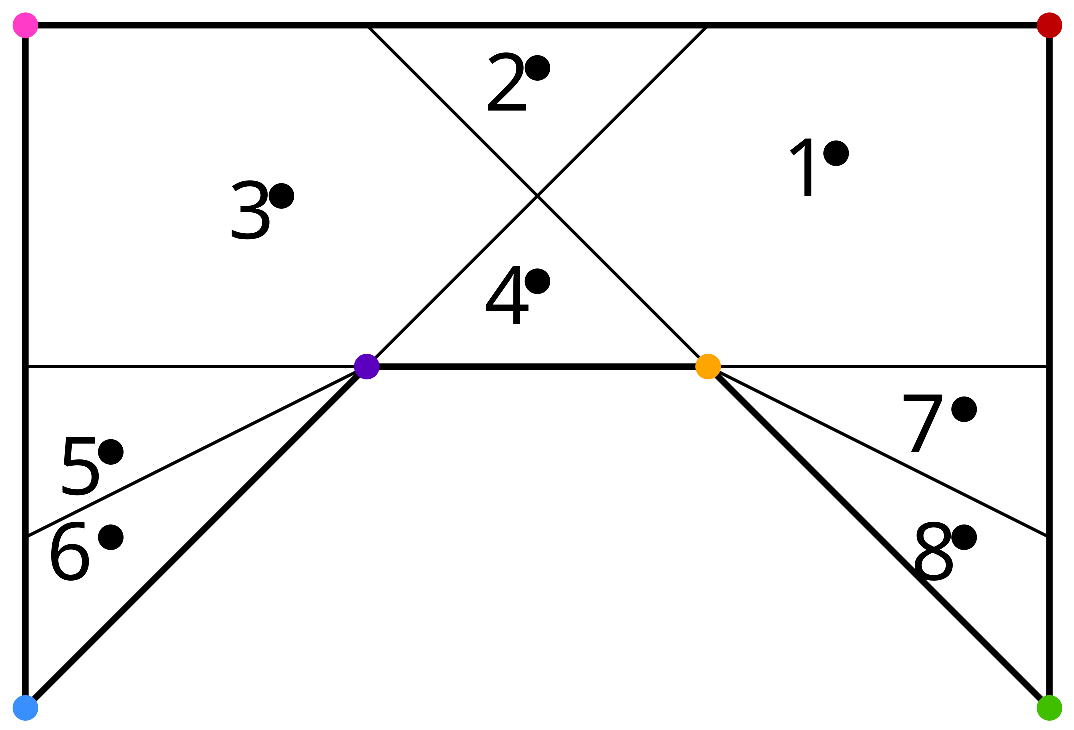
1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons
3. Place a point in each face of the arrangement (or simply take set of faces)
4. Label each point/faces for clarity $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$
5. For visibility polygons create group of visible points

$$red = \{1, 2, 3, 4, 5, 7, 8\}$$



Compute range space

1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons
3. Place a point in each face of the arrangement (or simply take set of faces)
4. Label each point/faces for clarity $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$
5. For visibility polygons create group of visible points

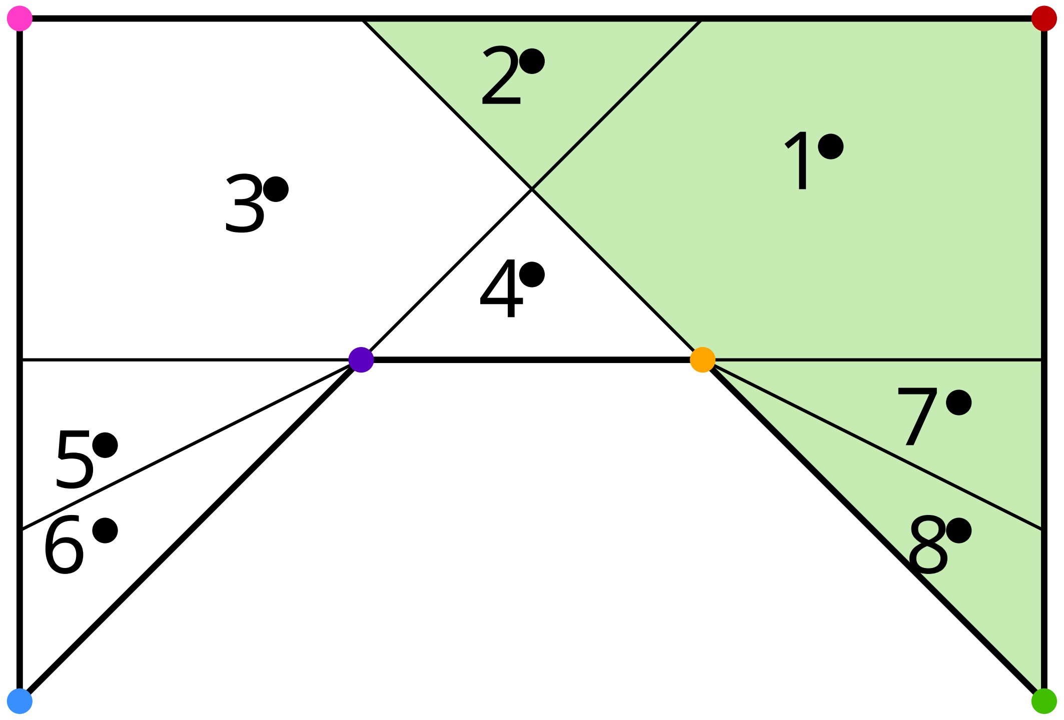


$$red = \{1, 2, 3, 4, 5, 7, 8\}$$
$$green = \{?\}$$

$$S_1 = \{2, 7, 8\}$$
$$S_2 = \{1, 2, 7, 8\}$$
$$S_3 = \{1, 2, 3, 4, 7, 8\}$$
$$S_4 = \{7, 8\}$$

Compute range space

1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons
3. Place a point in each face of the arrangement (or simply take set of faces)
4. Label each point/faces for clarity $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$
5. For visibility polygons create group of visible points



$red = \{1, 2, 3, 4, 5, 7, 8\}$

$green = \{1, 2, 7, 8\}$

$S_1 = \{2, 7, 8\}$

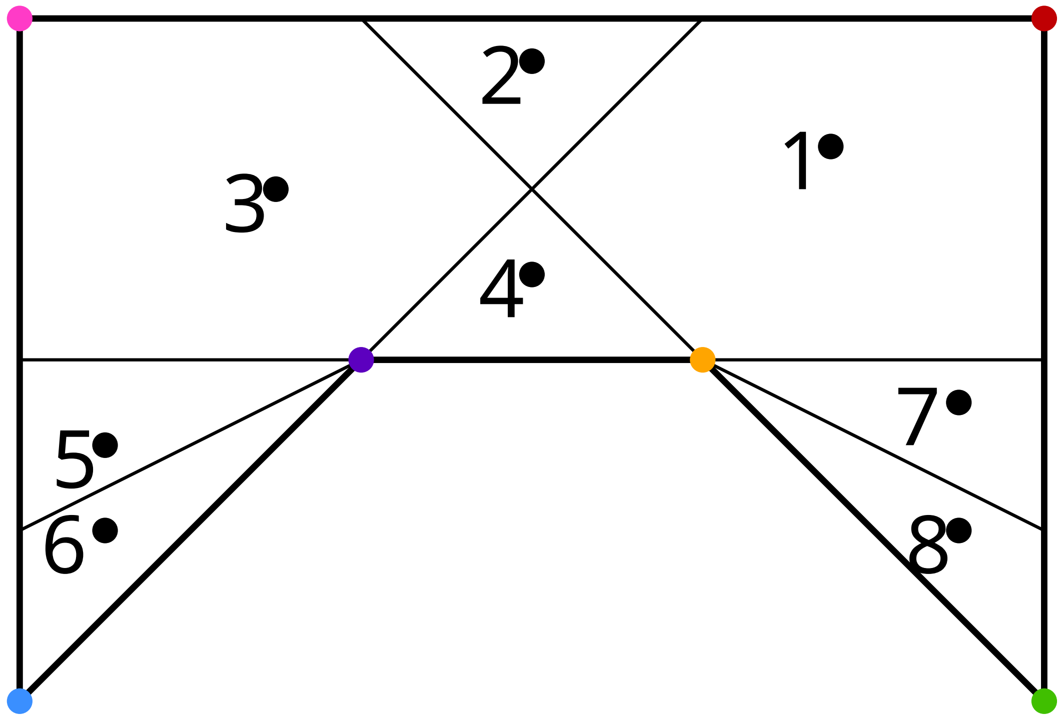
$\boxed{S_2 = \{1, 2, 7, 8\}}$

$S_3 = \{1, 2, 3, 4, 7, 8\}$

$S_4 = \{7, 8\}$

Compute range space

1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons
3. Place a point in each face of the arrangement (or simply take set of faces)
4. Label each point/faces for clarity $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$
5. For visibility polygons create group of visible points

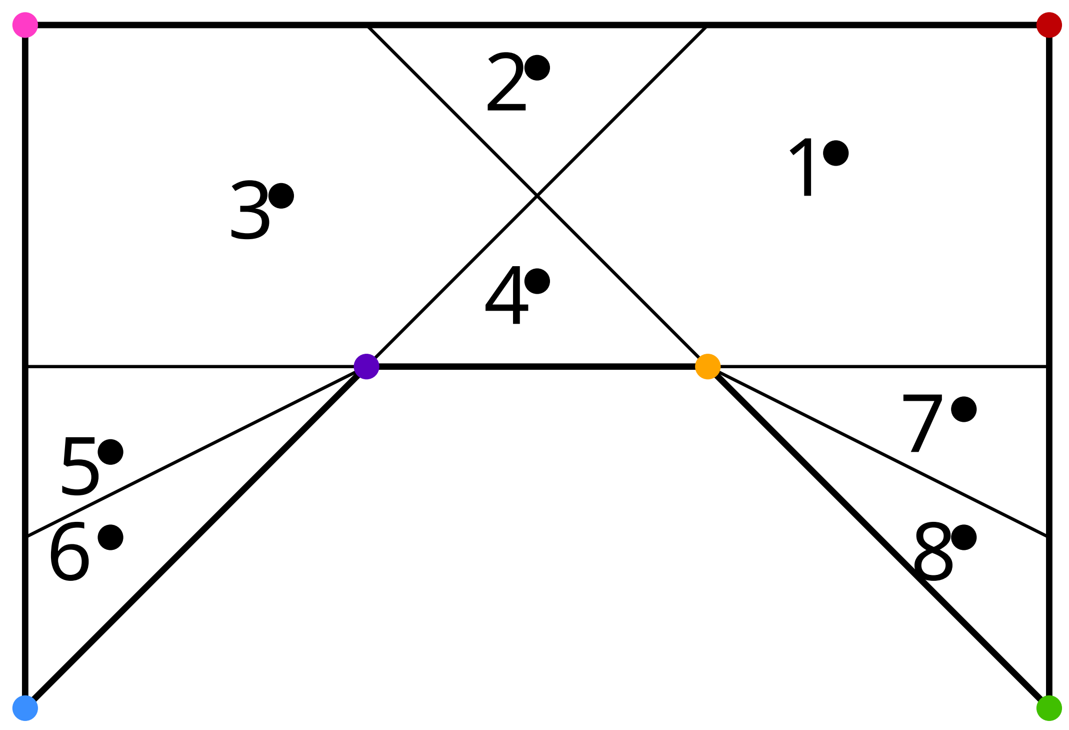


$red = \{1, 2, 3, 4, 5, 7, 8\}$
$green = \{1, 2, 7, 8\}$
$orange = \{1, 2, 3, 4, 7, 8\}$
$purple = \{1, 2, 3, 4, 5, 6\}$
$blue = \{2, 3, 5, 6\}$
$pink = \{1, 2, 3, 4, 5, 6, 7\}$

Compute range space

1. Calculate all visibility polygons for the vertices of P
2. Create arrangement of visibility polygons
3. Place a point in each face of the arrangement (or simply take set of faces)
4. Label each point/faces for clarity $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$
5. For visibility polygons create group of visible points

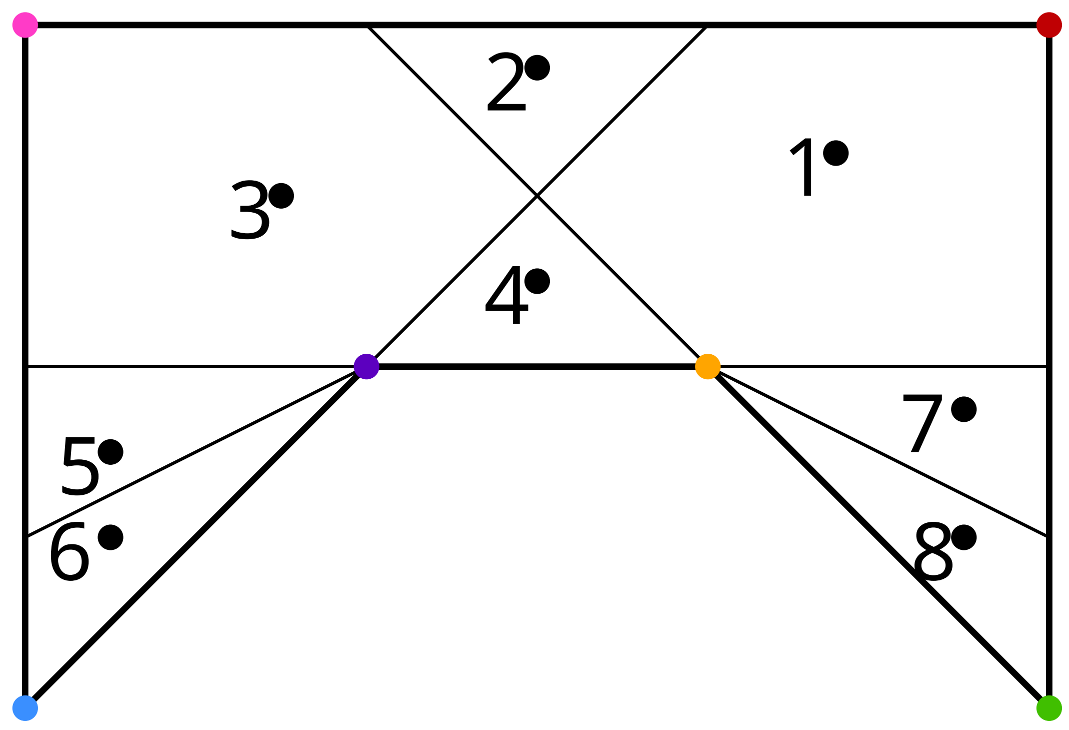
$$red = \{1, 2, 3, 4, 5, 7, 8\}$$
$$green = \{1, 2, 7, 8\}$$
$$orange = \{1, 2, 3, 4, 7, 8\}$$
$$purple = \{1, 2, 3, 4, 5, 6\}$$
$$blue = \{2, 3, 5, 6\}$$
$$pink = \{1, 2, 3, 4, 5, 6, 7\}$$



$$\mathcal{R} = \{red, green, orange, purple, blue, pink\}$$

Compute range space

art gallery problem: set cover problem on $(X, \mathcal{R})$



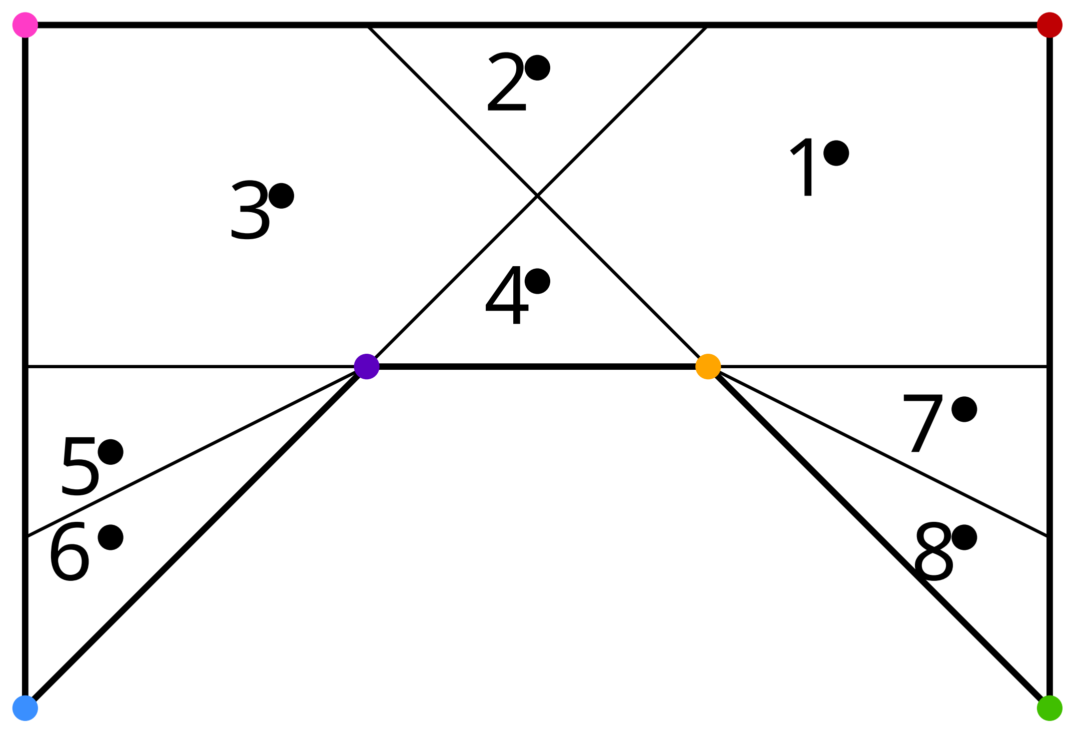
$$X = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

$$\begin{aligned} red &= \{1, 2, 3, 4, 5, 7, 8\} \\ green &= \{1, 2, 7, 8\} \\ orange &= \{1, 2, 3, 4, 7, 8\} \\ purple &= \{1, 2, 3, 4, 5, 6\} \\ blue &= \{2, 3, 5, 6\} \\ pink &= \{1, 2, 3, 4, 5, 6, 7\} \end{aligned}$$

$$\mathcal{R} = \{red, green, orange, purple, blue, pink\}$$

Compute range space

art gallery problem: set cover problem on $(X, \mathcal{R})$



$$X = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

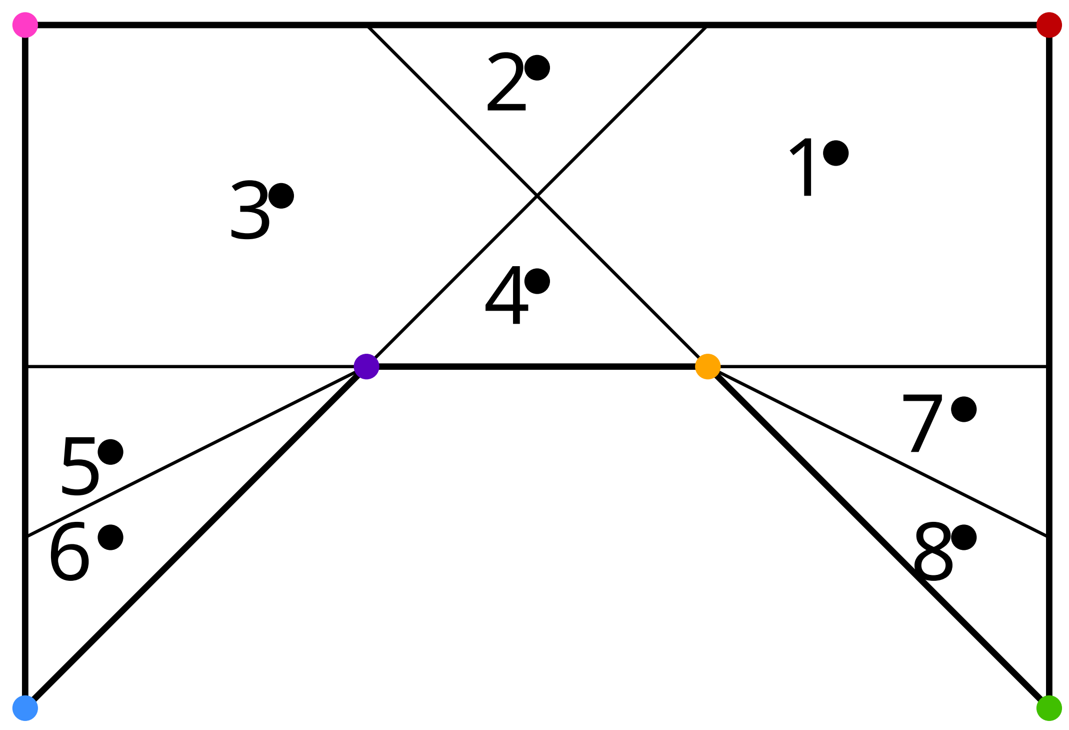
$$\begin{aligned}
\textit{red} &= \{1, 2, 3, 4, 5, 7, 8\} \\
\textit{green} &= \{1, 2, 7, 8\} \\
\textit{orange} &= \{1, 2, 3, 4, 7, 8\} \\
\textit{purple} &= \{1, 2, 3, 4, 5, 6\} \\
\textit{blue} &= \{2, 3, 5, 6\} \\
\textit{pink} &= \{1, 2, 3, 4, 5, 6, 7\}
\end{aligned}$$

$$\mathcal{R} = \{\textit{red}, \textit{green}, \textit{orange}, \textit{purple}, \textit{blue}, \textit{pink}\}$$

Compute range space

art gallery problem: set cover problem on $(X, \mathcal{R})$

dual VC-dimension is constant (see exercises)



$$X = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

$$red = \{1, 2, 3, 4, 5, 7, 8\}$$
$$green = \{1, 2, 7, 8\}$$
$$orange = \{1, 2, 3, 4, 7, 8\}$$
$$purple = \{1, 2, 3, 4, 5, 6\}$$
$$blue = \{2, 3, 5, 6\}$$
$$pink = \{1, 2, 3, 4, 5, 6, 7\}$$

$$\mathcal{R} = \{red, green, orange, purple, blue, pink\}$$

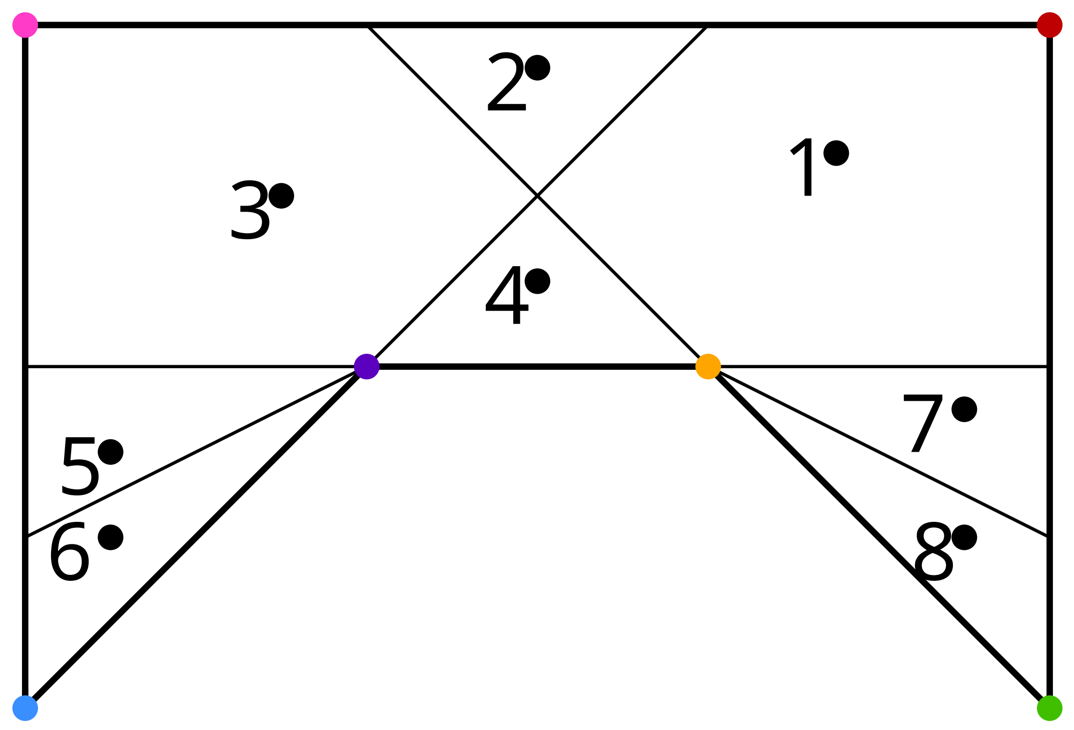
Compute range space

art gallery problem: set cover problem on $(X, \mathcal{R})$

dual VC-dimension is constant (see exercises)

Previous algorithm applies

$$X = \{1,2,3,4,5,6,7,8\}$$



$$\mathit{red} = \{1,2,3,4,5,7,8\}$$
$$\mathit{green} = \{1,2,7,8\}$$
$$\mathit{orange} = \{1,2,3,4,7,8\}$$
$$\mathit{purple} = \{1,2,3,4,5,6\}$$
$$\mathit{blue} = \{2,3,5,6\}$$
$$\mathit{pink} = \{1,2,3,4,5,6,7\}$$

$$\mathcal{R} = \{\mathit{red}, \mathit{green}, \mathit{orange}, \mathit{purple}, \mathit{blue}, \mathit{pink}\}$$

Summary

general set cover problem: $O(\log n)$-approximation using greedy algorithm

geometric set cover problem: $O(\log k)$-approximation using sampling with reweighting (for finite VC-dimension)

applications: covering with disks and art gallery problem