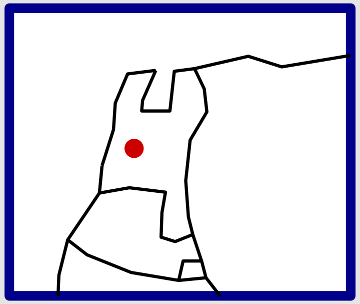
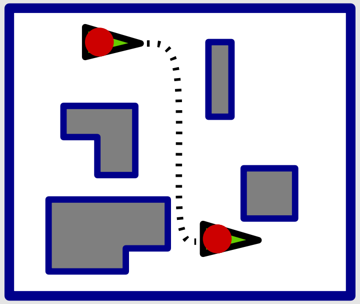
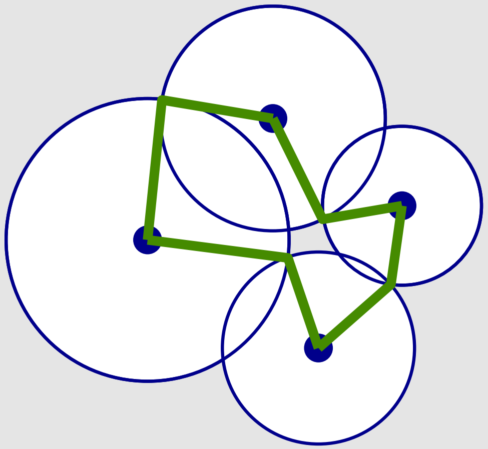
Computational and Combinatorial Geometry

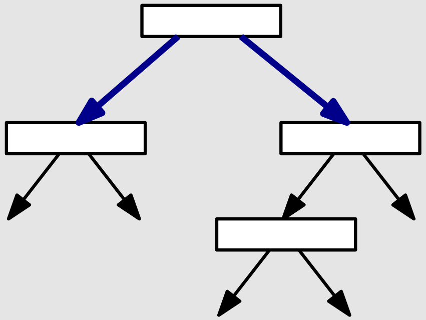
Mark de Berg

TU Eindhoven



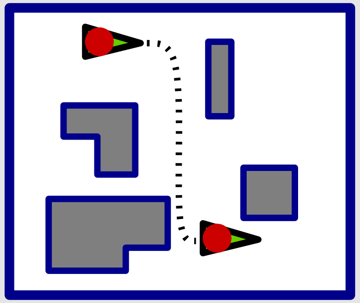


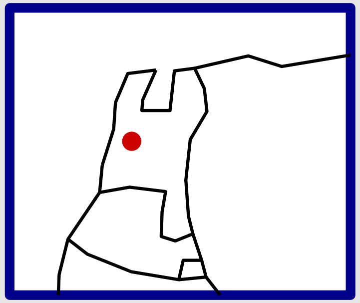


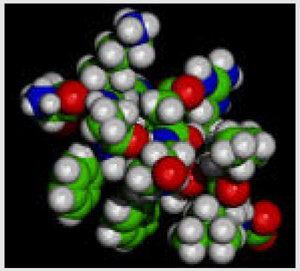


TU/e

Computational geometry: algorithms for spatial data.


motion planning


point location


docking

. . . and combinatorial geometry

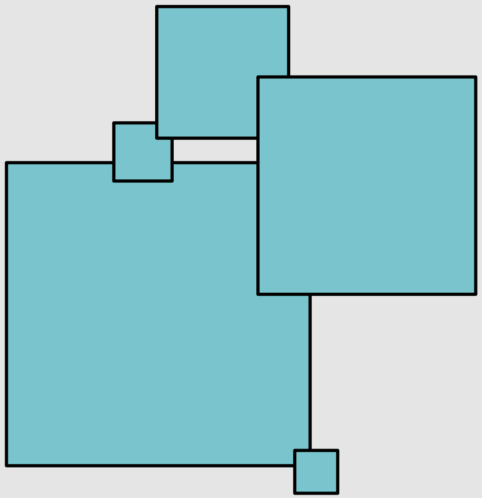
Puzzle I:

Place n squares of arbitrary sizes to make the most complicated figure.

Puzzle I:

Place n squares of arbitrary sizes to make the most complicated figure.

$n = 5$



TU/e

Puzzle I:

Place n squares of arbitrary sizes to make the most complicated figure.

$n = 5$

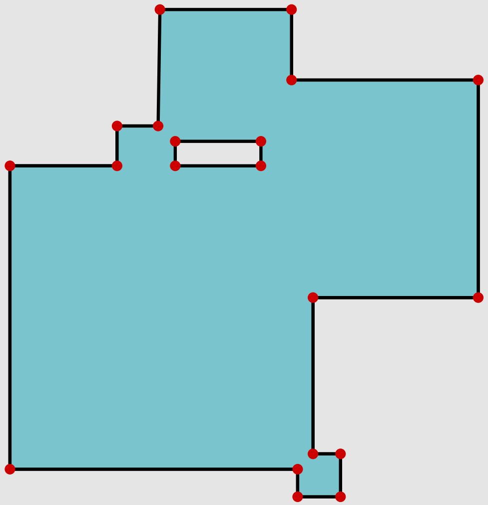


Figure with 20 vertices.

Puzzle I:

Place n squares of arbitrary sizes to make the most complicated figure.

$n = 5$

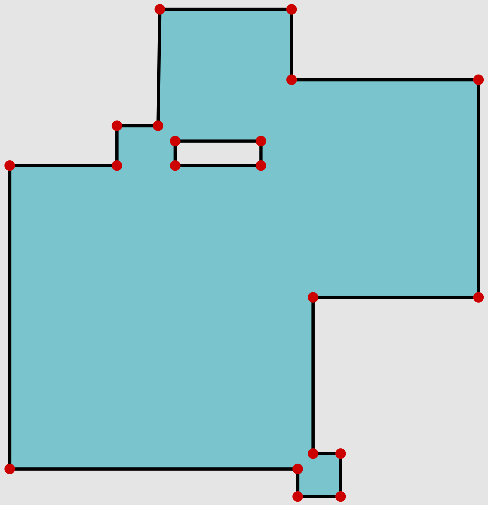
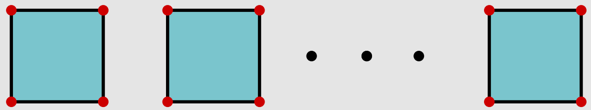
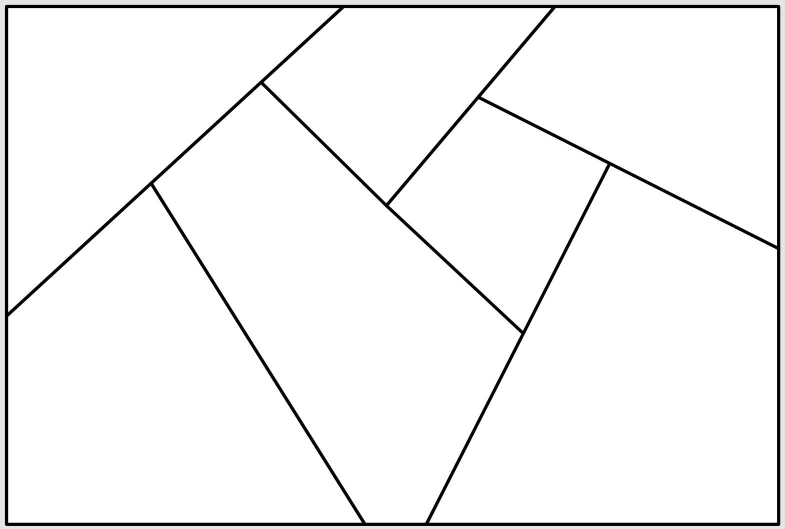


Figure with 20 vertices.

- Can we make a more complicated figure?
- What is the max complexity we can create as a function of n? Can we do more than $4n$?

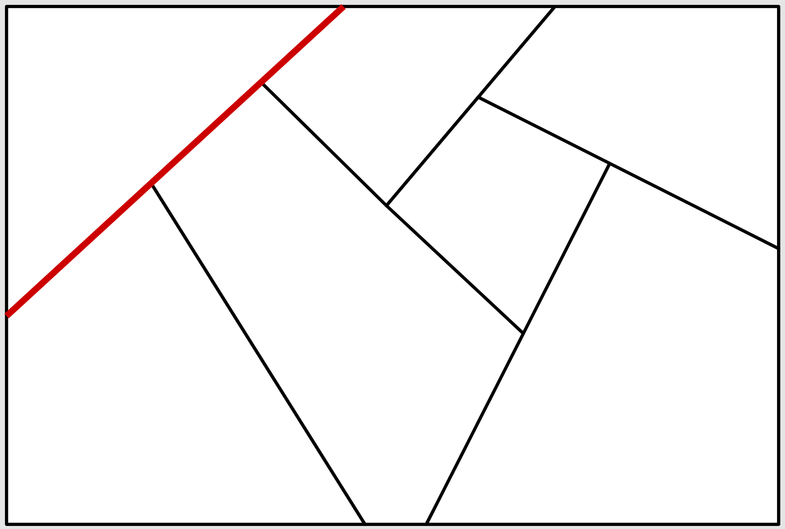


Puzzle II: Cutting glass plates.



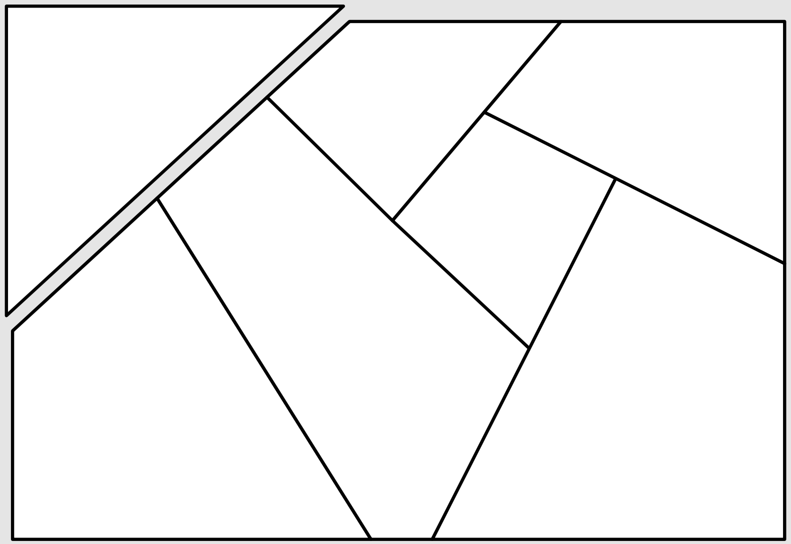
How many cuts do we need to cut out all the pieces if we always have to cut completely through?

Puzzle II: Cutting glass plates.



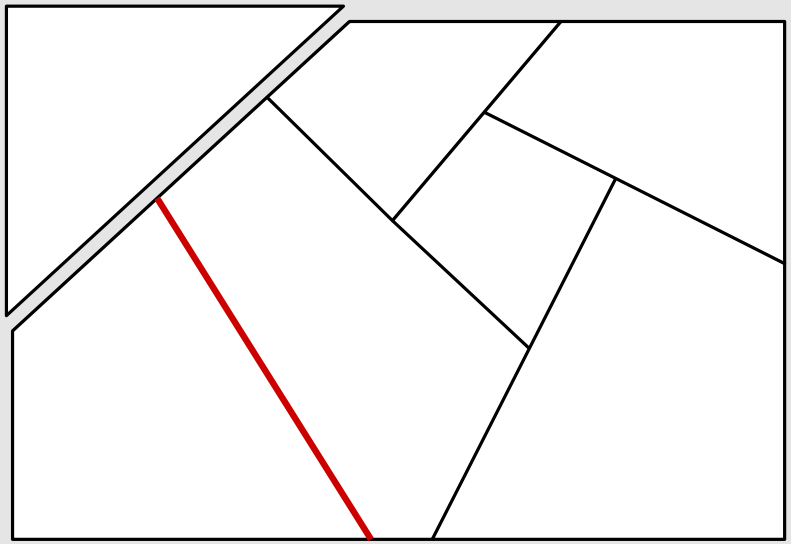
How many cuts do we need to cut out all the pieces if we always have to cut completely through?

Puzzle II: Cutting glass plates.



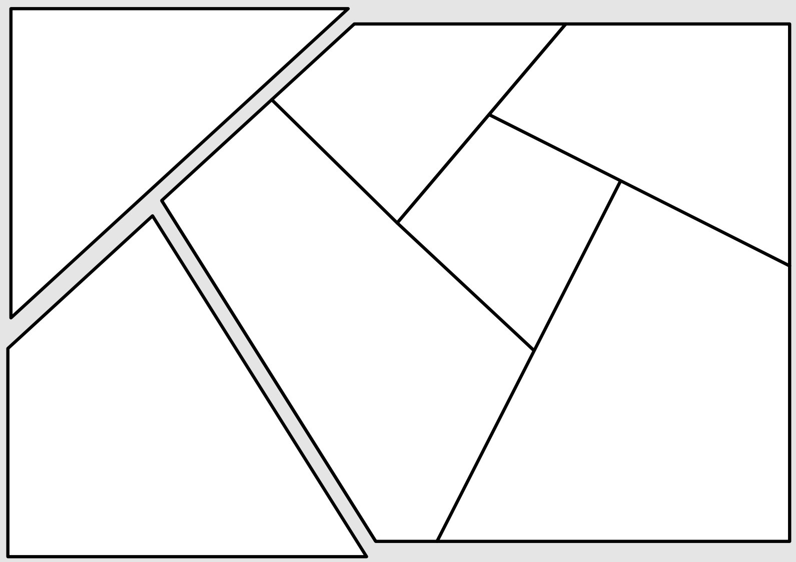
How many cuts do we need to cut out all the pieces if we always have to cut completely through?

Puzzle II: Cutting glass plates.



How many cuts do we need to cut out all the pieces if we always have to cut completely through?

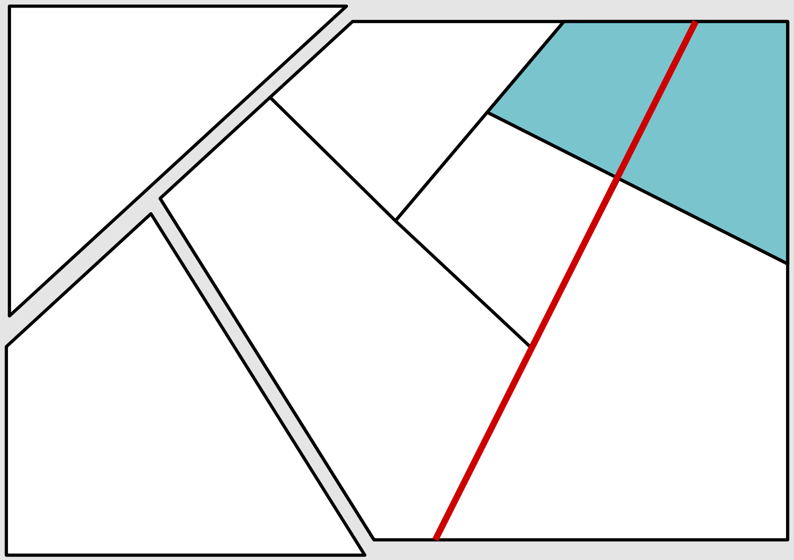
Puzzle II: Cutting glass plates.



How many cuts do we need to cut out all the pieces if we always have to cut completely through?

TU/e

Puzzle II: Cutting glass plates.

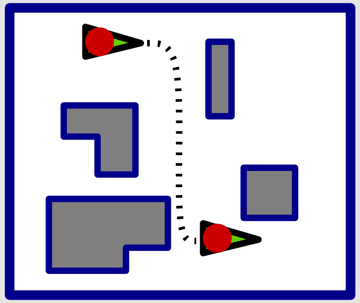
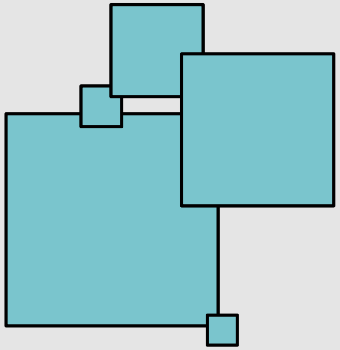


piece is cut into two fragments

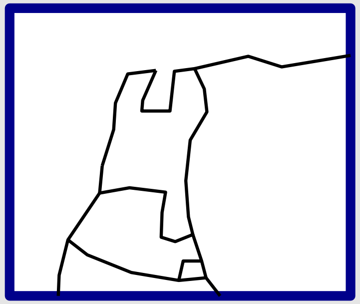
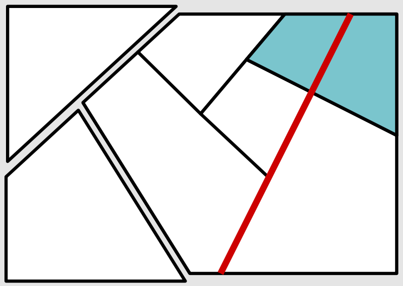
How many cuts do we need to cut out all the pieces if we always have to cut completely through?

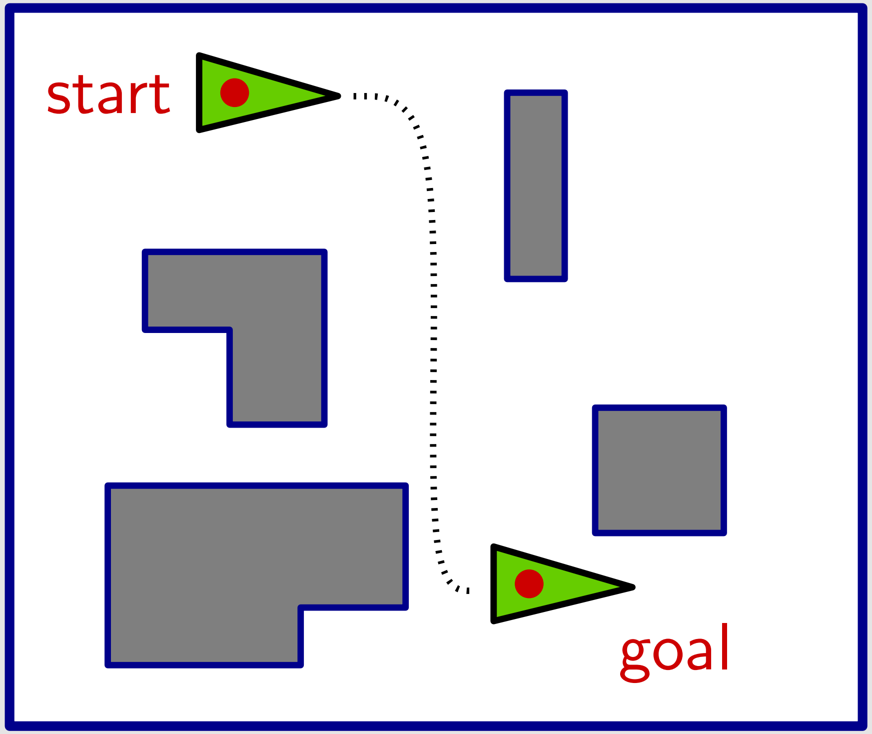
Rest of the talk:

- Relation motion planning to union complexity

- Relation point location to glass cutting

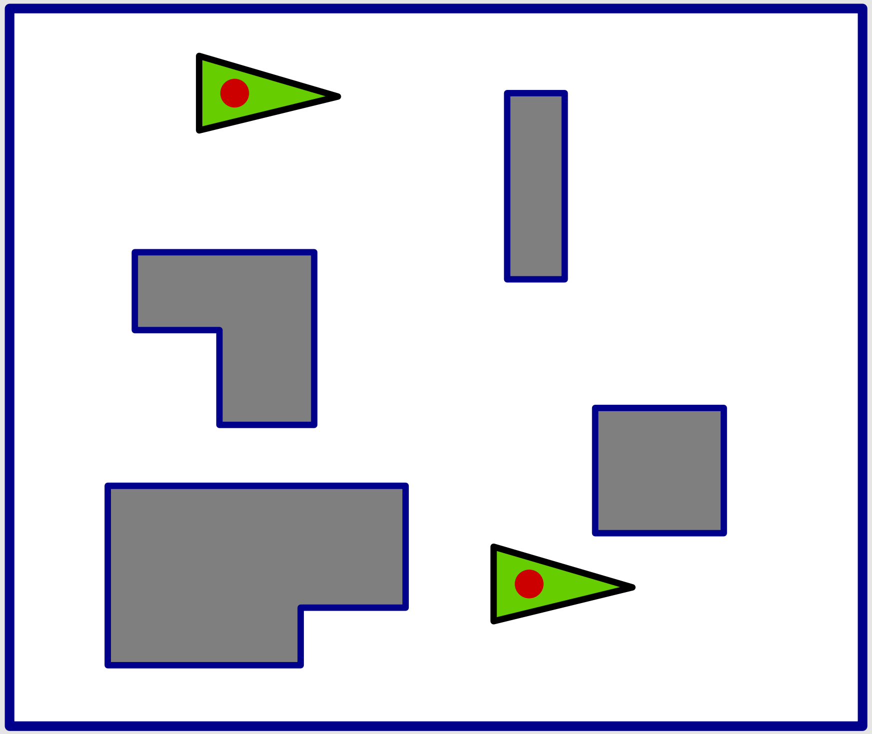


Given

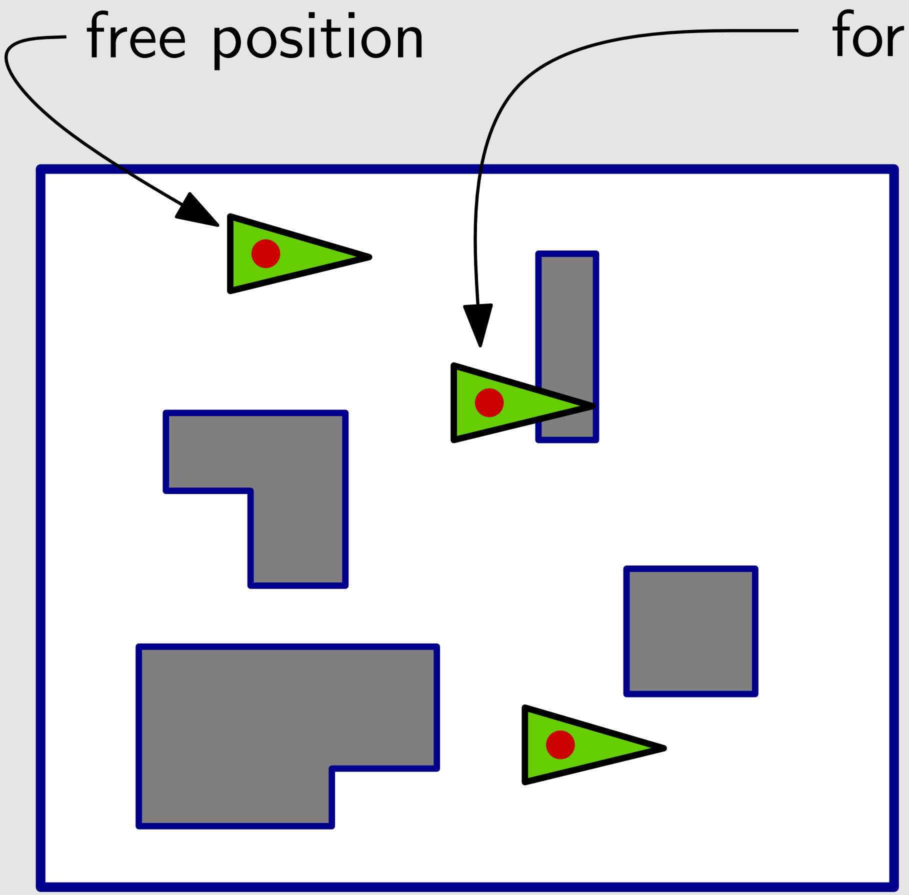
- a robot R, with start and goal position
- set S of obstacles

find collision-free path for the robot.

Simple case: 2D "triangle-robot" that cannot rotate

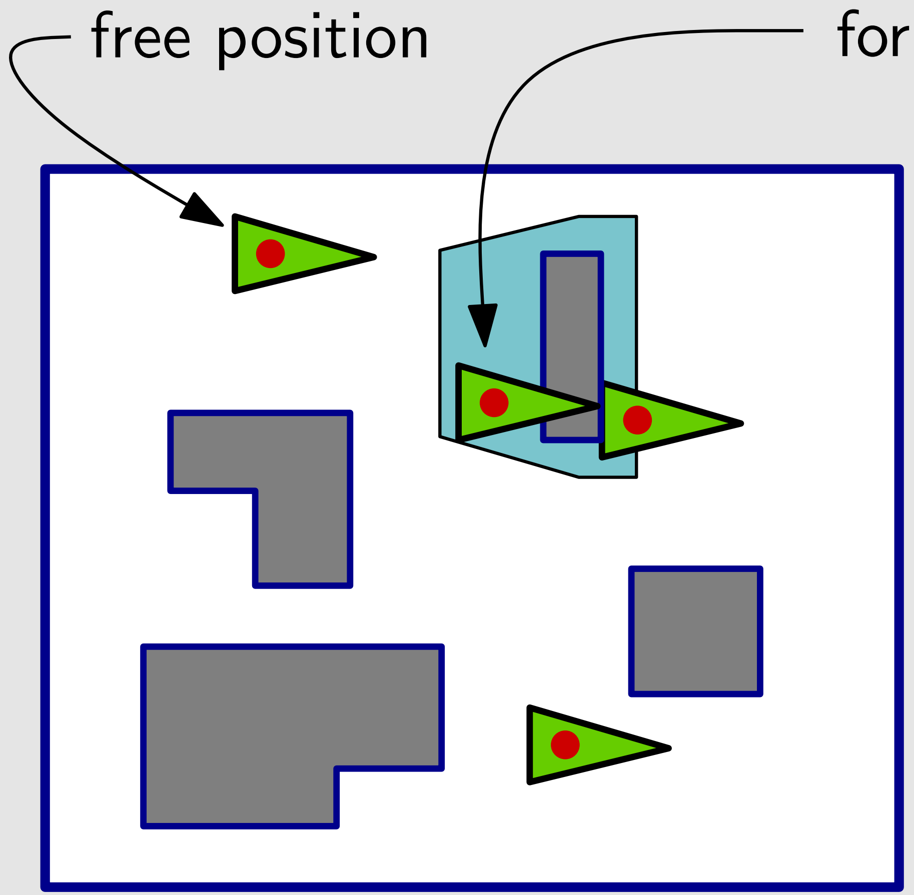


Simple case: 2D "triangle-robot" that cannot rotate



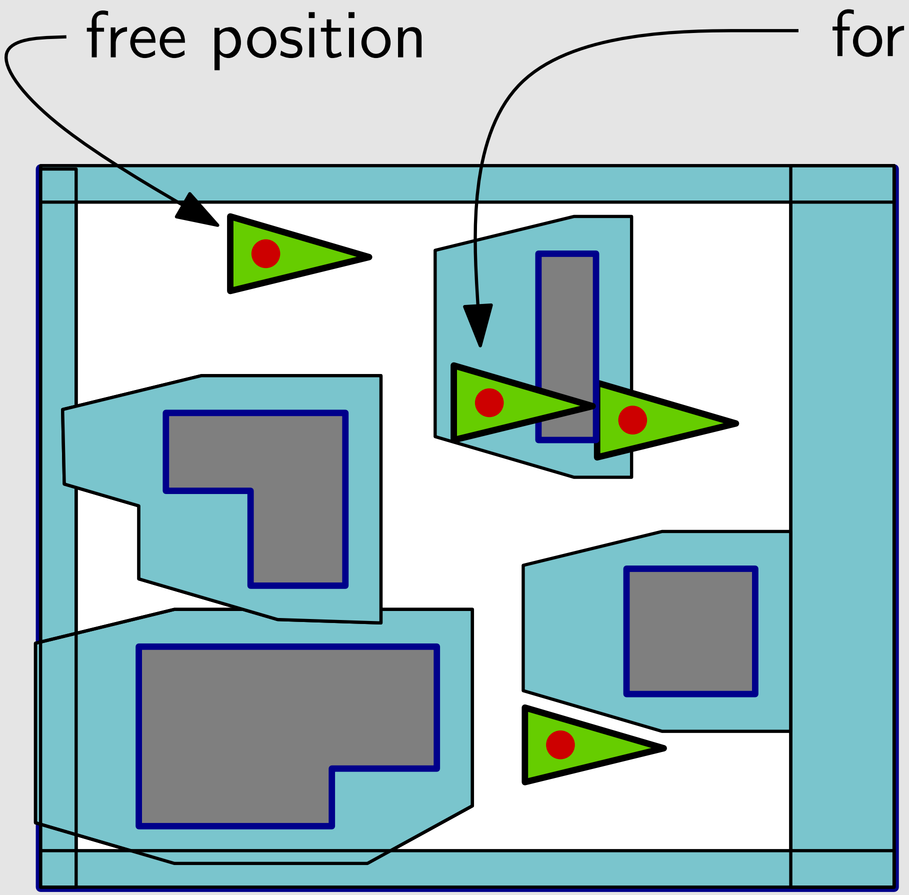
Find a path for the robot that only uses free positions.

Simple case: 2D "triangle-robot" that cannot rotate



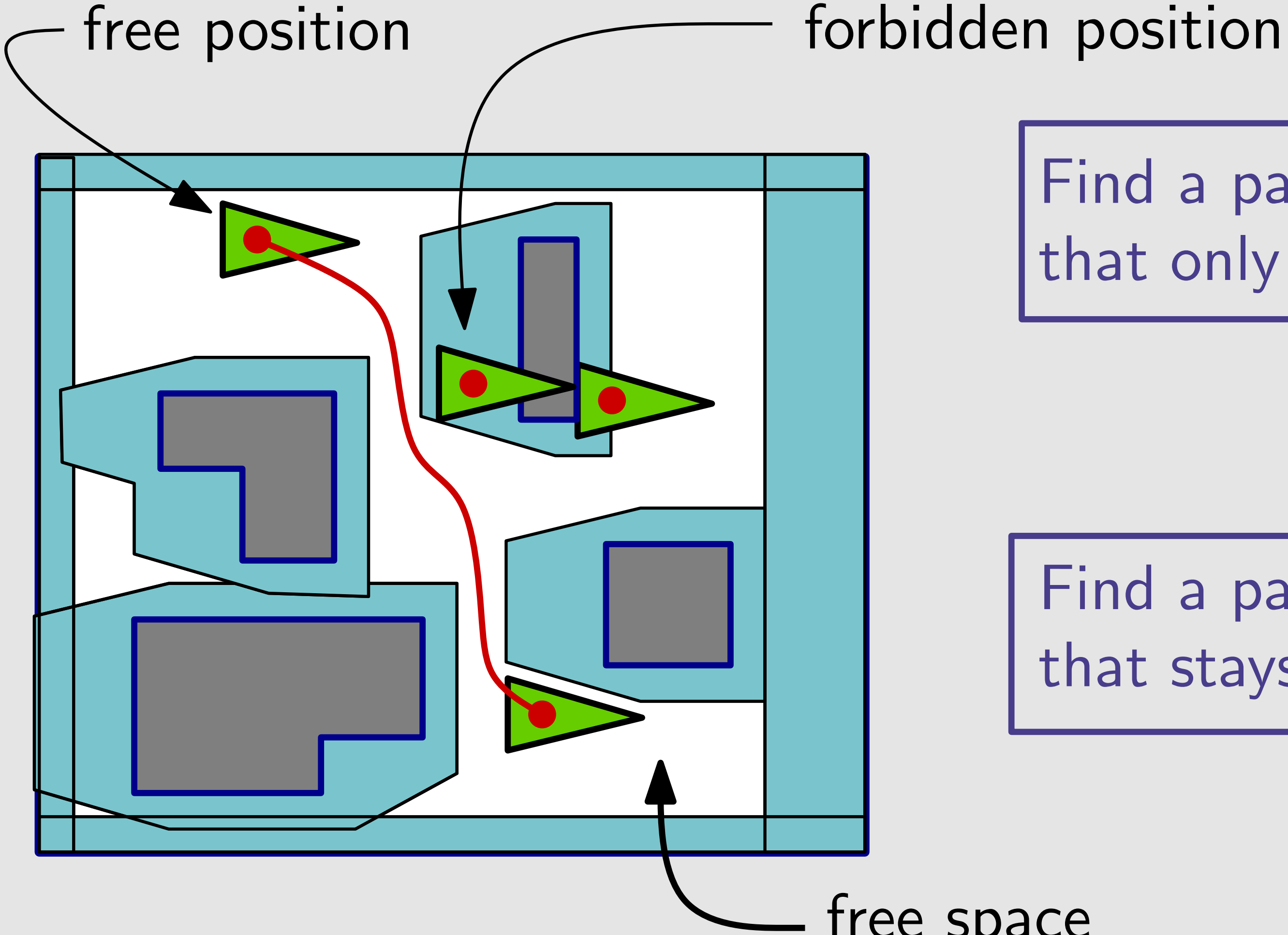
Find a path for the robot that only uses free positions.

Simple case: 2D "triangle-robot" that cannot rotate



Find a path for the robot that only uses free positions.

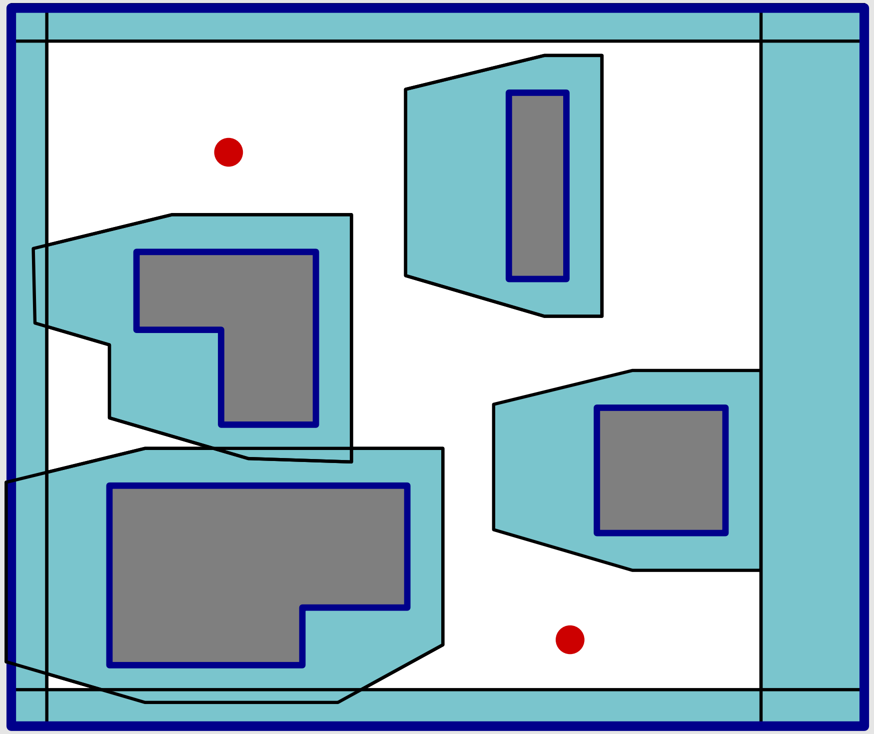
Simple case: 2D "triangle-robot" that cannot rotate



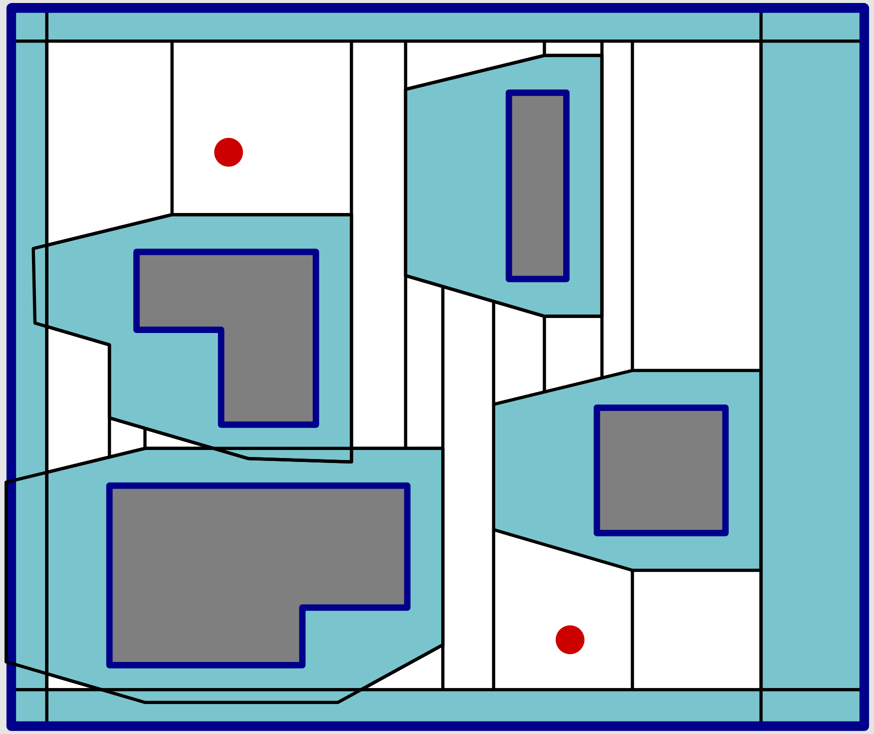
Find a path for the robot that only uses free positions.

↓

Find a path for a point that stays in the free space.

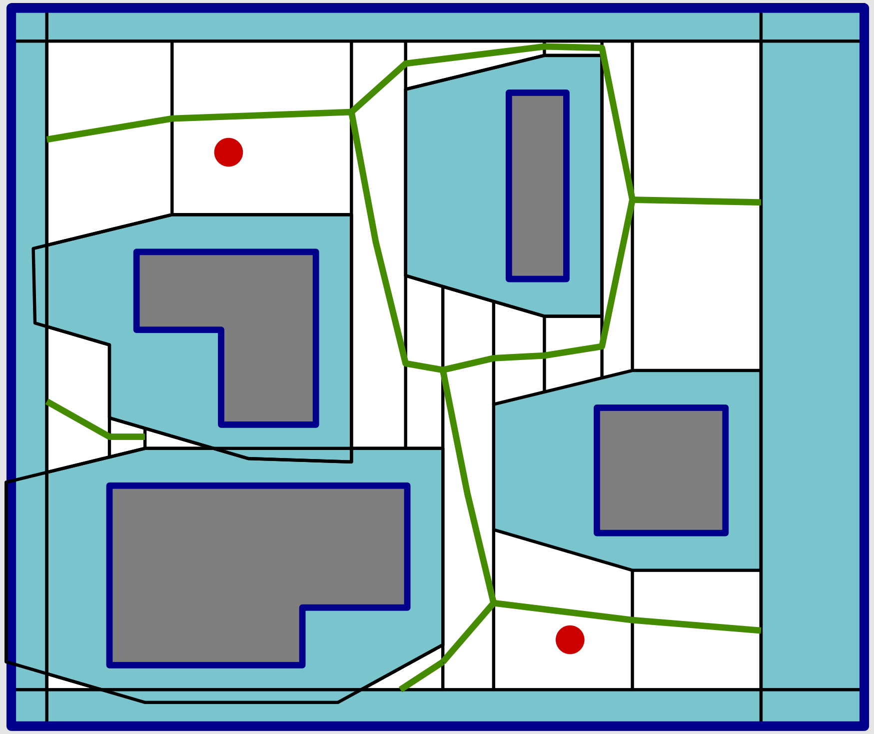


Find a path for a point that stays in the free space.



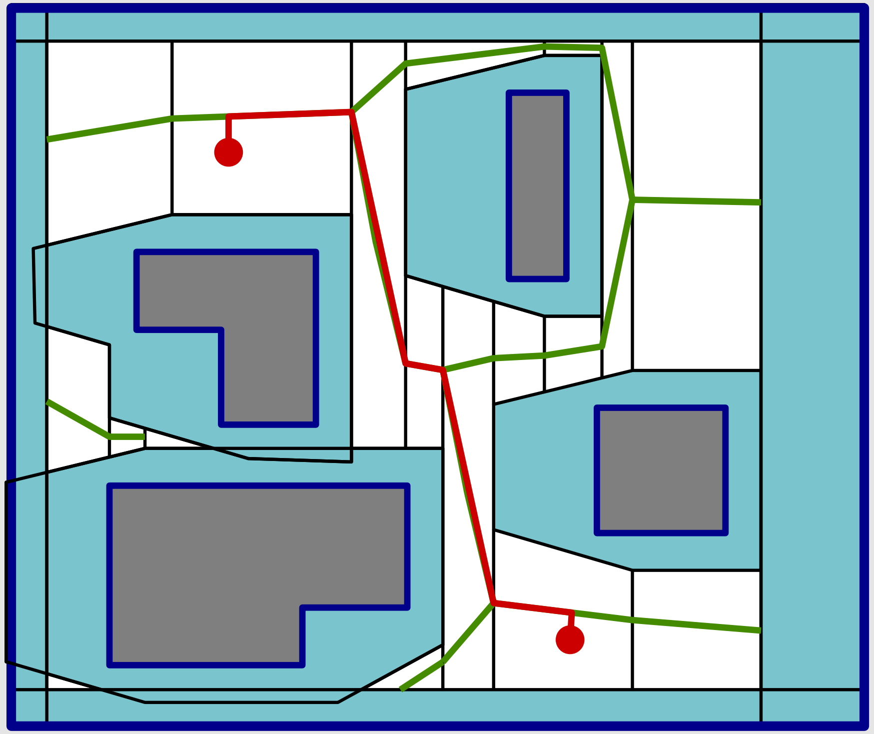
Find a path for a point
that stays in the free space.

1. Decompose free space into simple cells
2. Compute a "road network" based on these cells
3. Find path in network



Find a path for a point that stays in the free space.

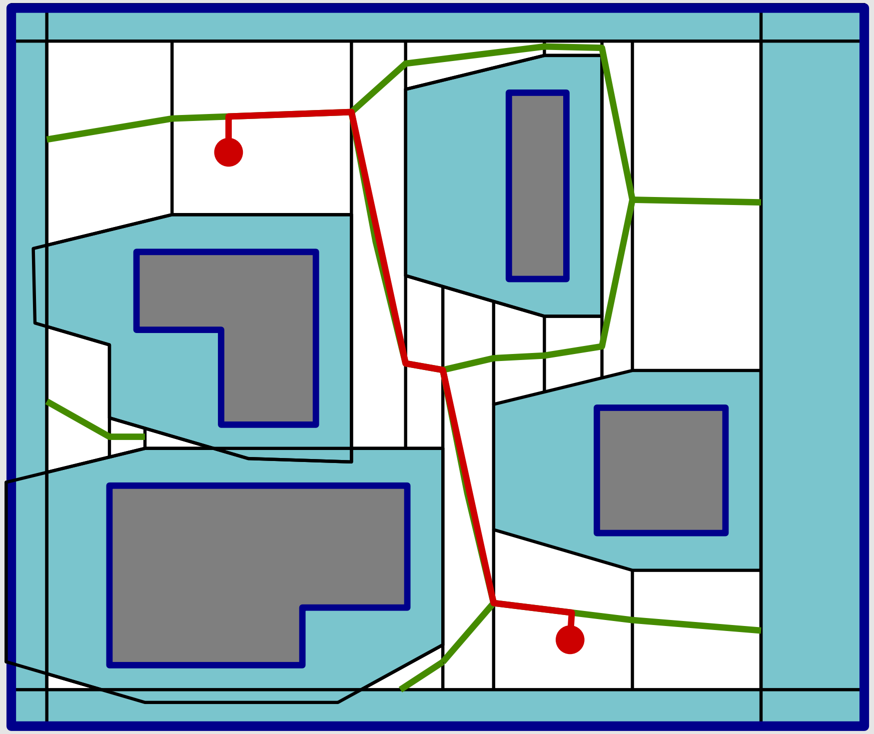
1. Decompose free space into simple cells
2. Compute a "road network" based on these cells
3. Find path in network



Find a path for a point
that stays in the free space.

1. Decompose free space into simple cells
2. Compute a "road network" based on these cells
3. Find path in network

Motion planning (cont'd)



Find a path for a point that stays in the free space.

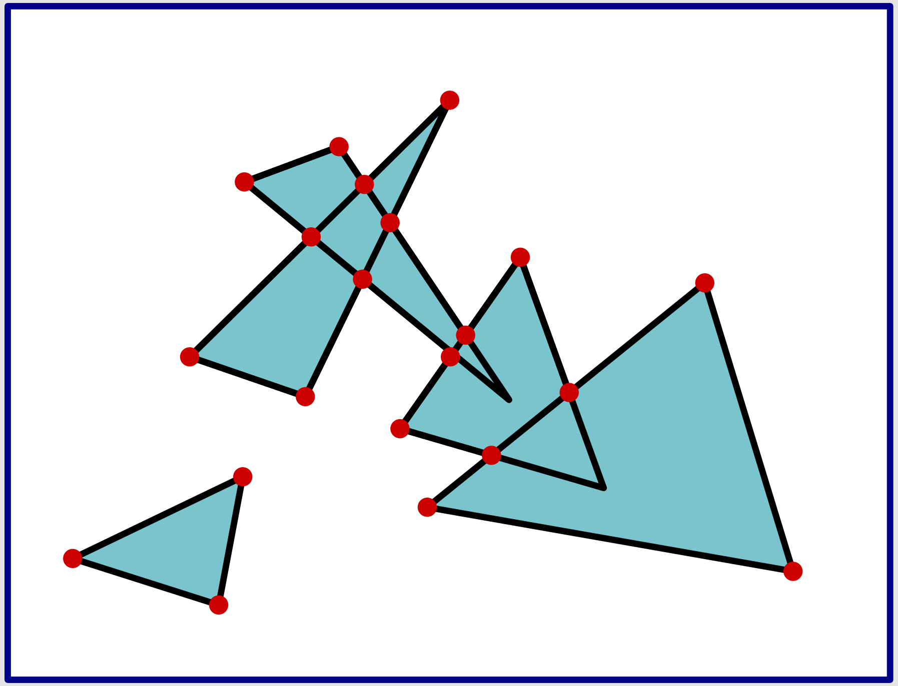
1. Decompose free space into simple cells
2. Compute a "road network" based on these cells
3. Find path in network

Computation time depends on number of cells.

Number of cells depends on how complicated the free space is.

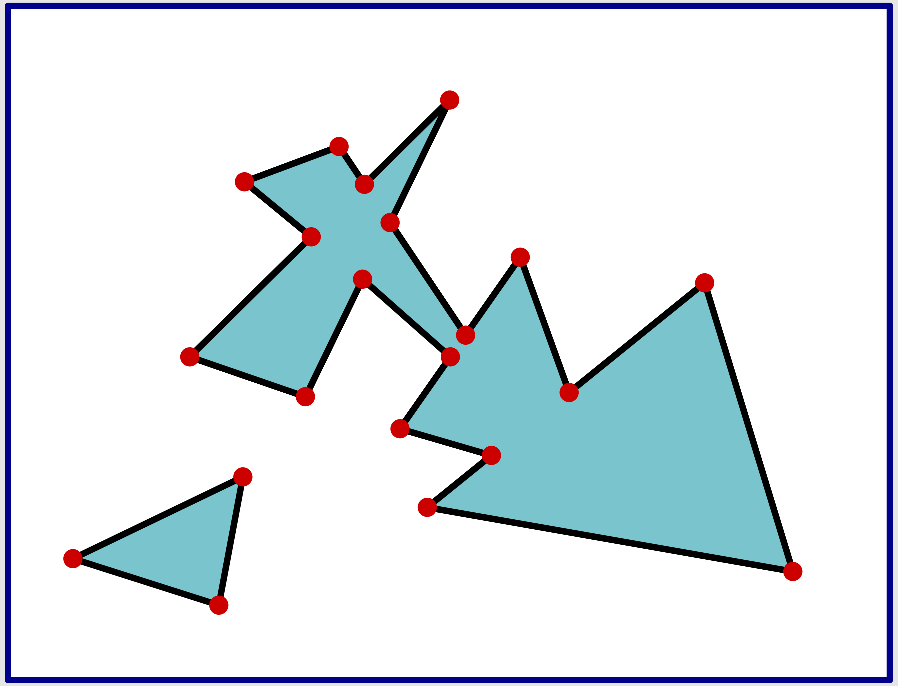
complexity of free space $=$ complexity of union of the obstacles

How high can the complexity be if we have n “simple” obstacles (triangles, rectangles, disks, . . .)



complexity of free space $=$ complexity of union of the obstacles

How high can the complexity be if we have n "simple" obstacles (triangles, rectangles, disks, ...)

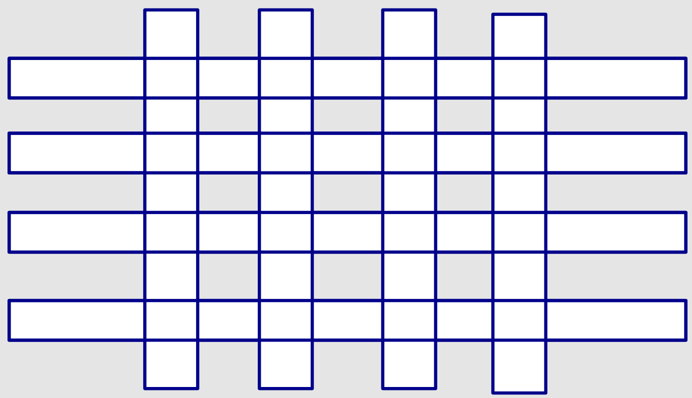


Combinatorial question:

what is the maximum number of vertices of the union of a collection of n objects of a certain type?

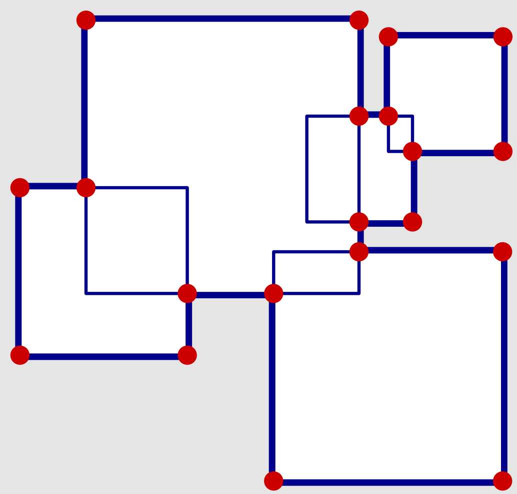
type of objects	maximum union complexity (n = number of objects)
rectangles	
squares	
disks	
triangles	
equilateral triangles	

type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	
disks	
triangles	
equilateral triangles	



type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	
disks	
triangles	
equilateral triangles	

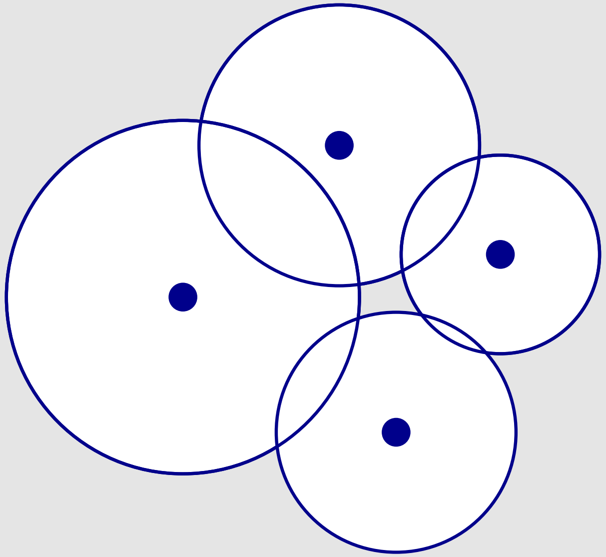
type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	$4n$
disks	
triangles	
equilateral triangles	



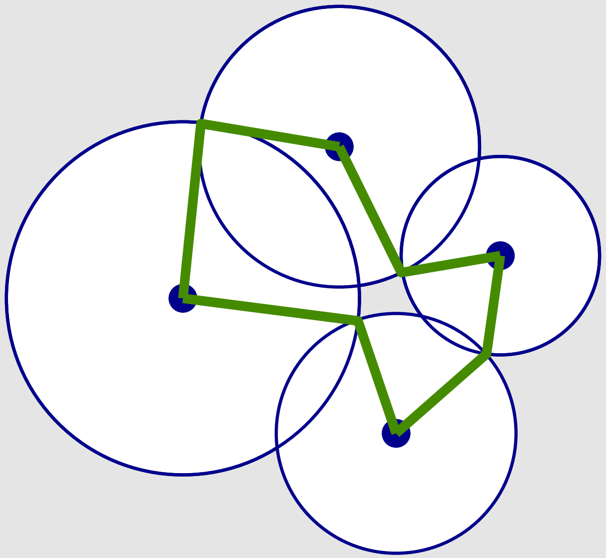
type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	$4n$
disks	
triangles	
equilateral triangles	

The union complexity of simple objects

type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	$4n$
disks	
triangles	
equilateral triangles	



type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	$4n$
disks	
triangles	
equilateral triangles	

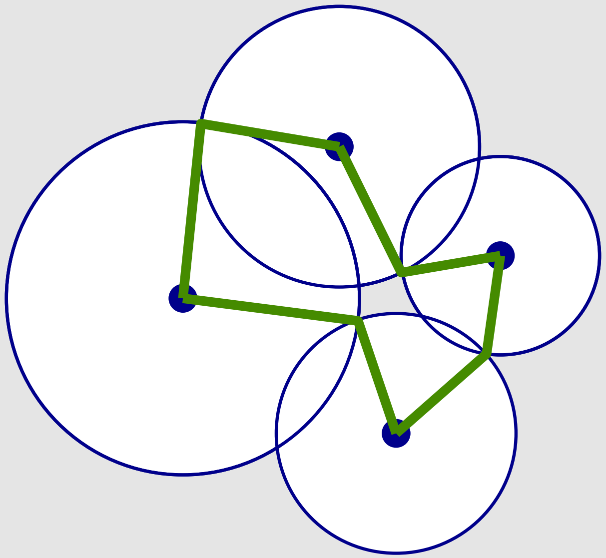


construct graph:

- nodes = circle centers
- one edge for every pair of disks defining a union vertex

graph is planar !

type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	$4n$
disks	$6n - 12$ (for $n > 2$)
triangles	
equilateral triangles	



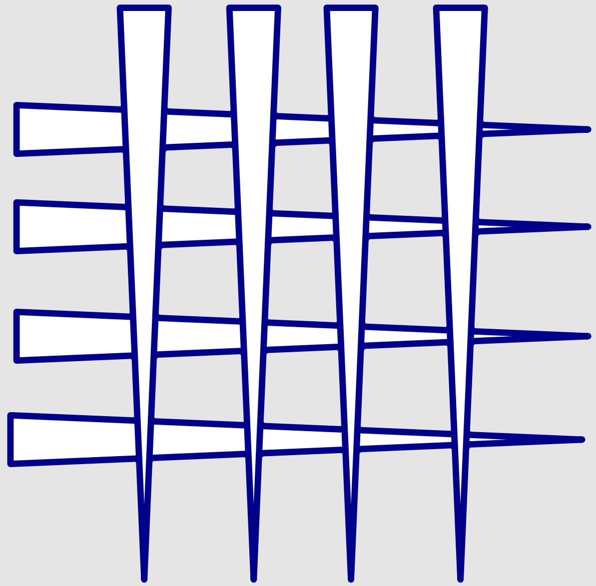
construct graph:

- nodes = circle centers
- one edge for every pair of disks defining a union vertex

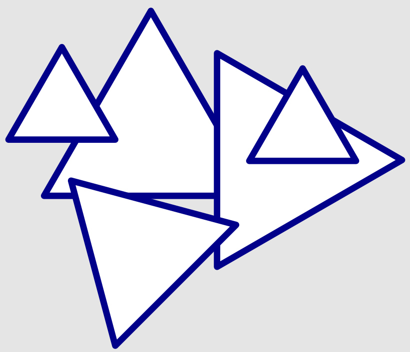
graph is planar !

type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	$4n$
disks	$6n - 12$ (for $n > 2$)
triangles	
equilateral triangles	

type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	$4n$
disks	$6n - 12$ (for $n > 2$)
triangles	$\sim n^2$
equilateral triangles	

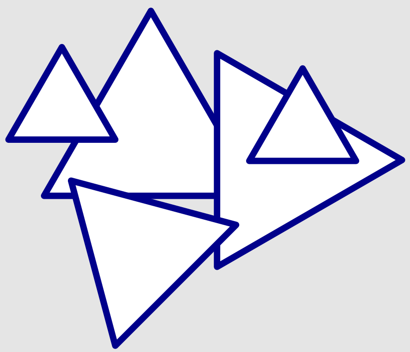


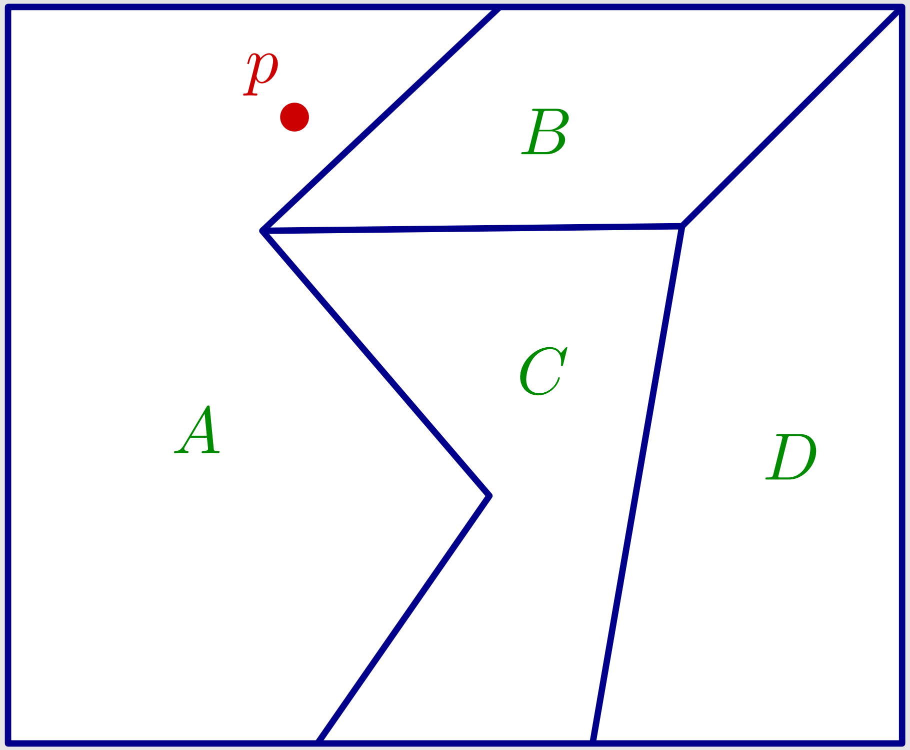
type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	$4n$
disks	$6n - 12$ (for $n > 2$)
triangles	$\sim n^2$
equilateral triangles	



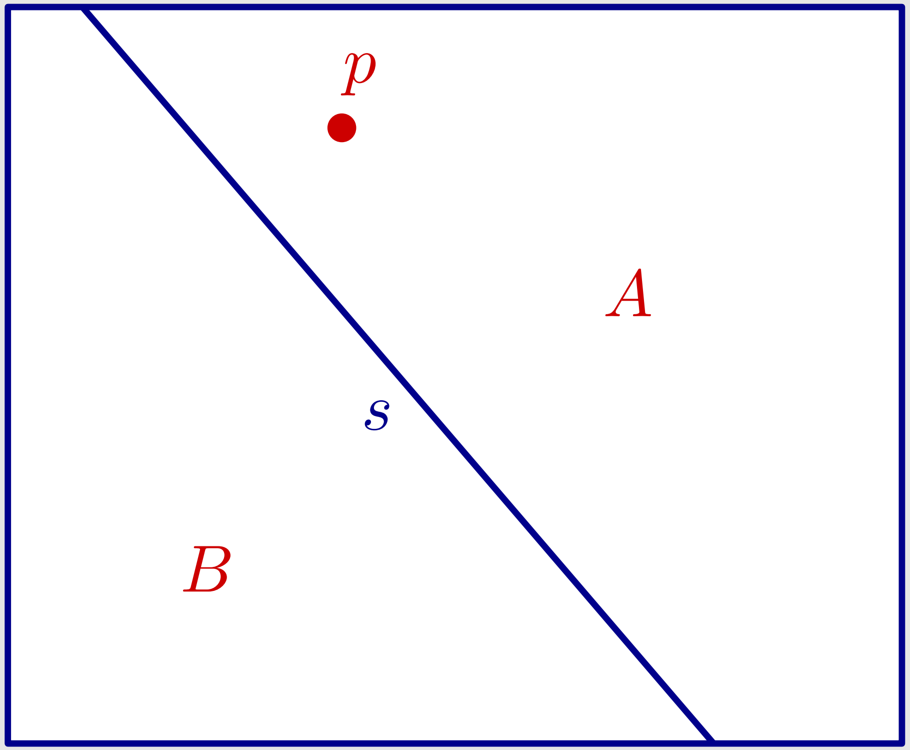
type of objects	maximum union complexity (n = number of objects)
rectangles	$\sim n^2$
squares	$4n$
disks	$6n - 12$ (for $n > 2$)
triangles	$\sim n^2$
equilateral triangles	We don't know!

It is more than linear, it is less than quadratic, but we do not know the exact answer. (We only know it is "very close to linear".)

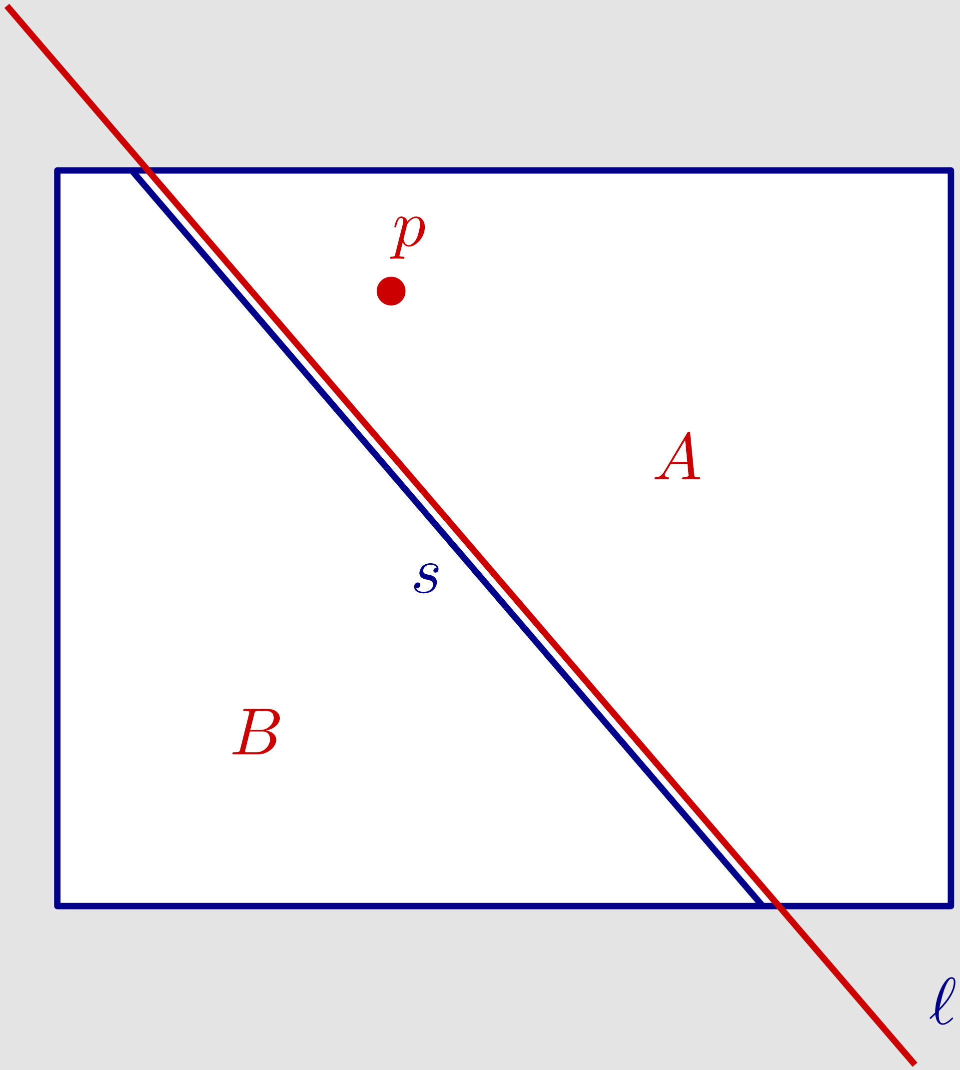




How can we compute which region contains the point p?



p
A
s
B

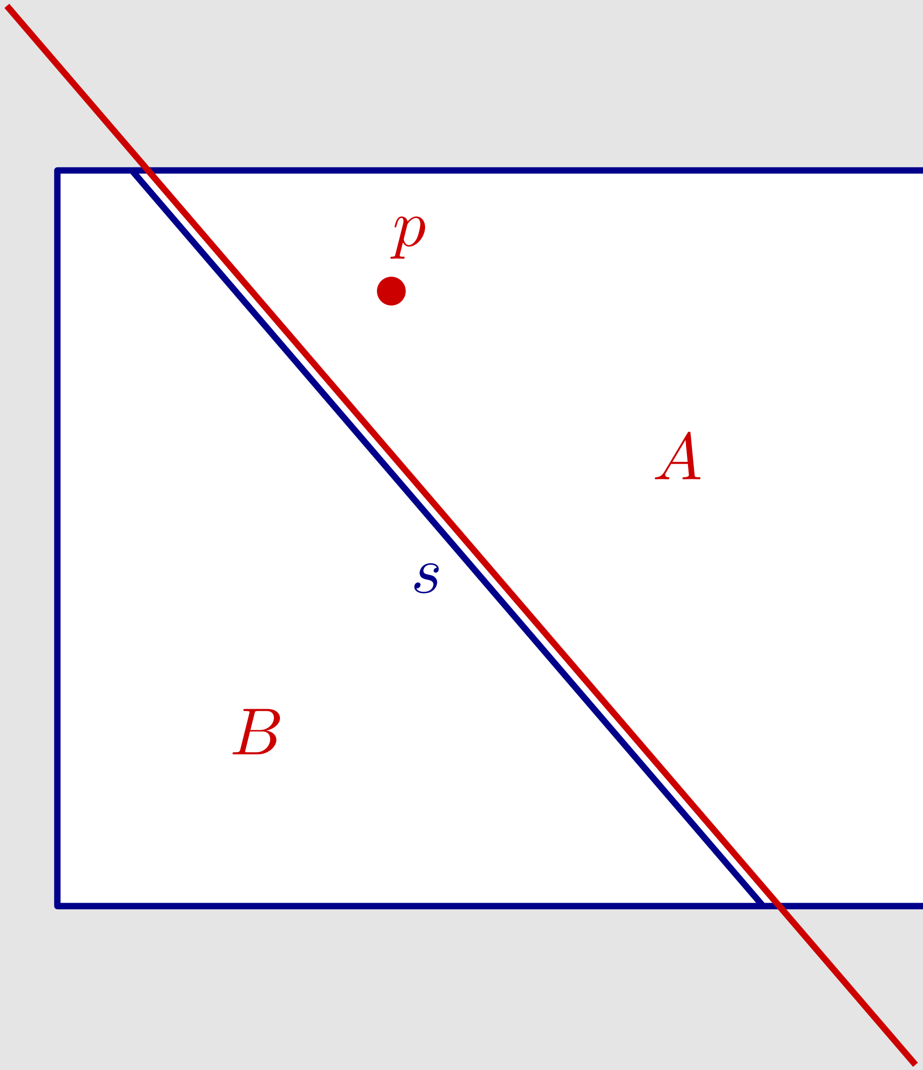


p above $\ell(s)$?

- yes: $p \in A$
- no: $p \notin A$

$\ell(s)$ = line containing s

Point location (cont'd)



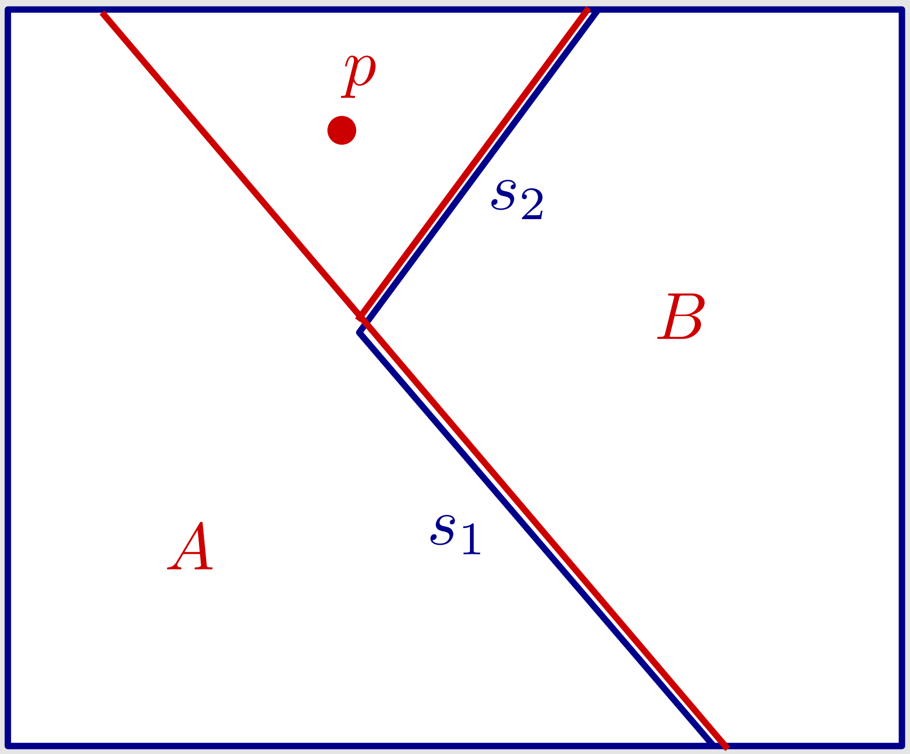
p above $\ell(s)$? $p_y > a \cdot p_x + b$?

- yes: $p \in A$
- no: $p \notin A$

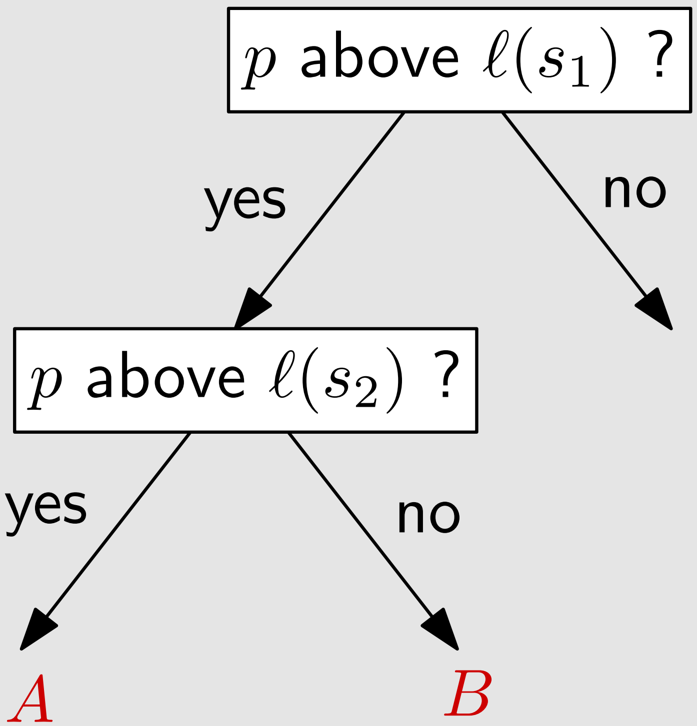
$\ell(s)$ = line containing s

$\ell(s)$: $y = ax + b$

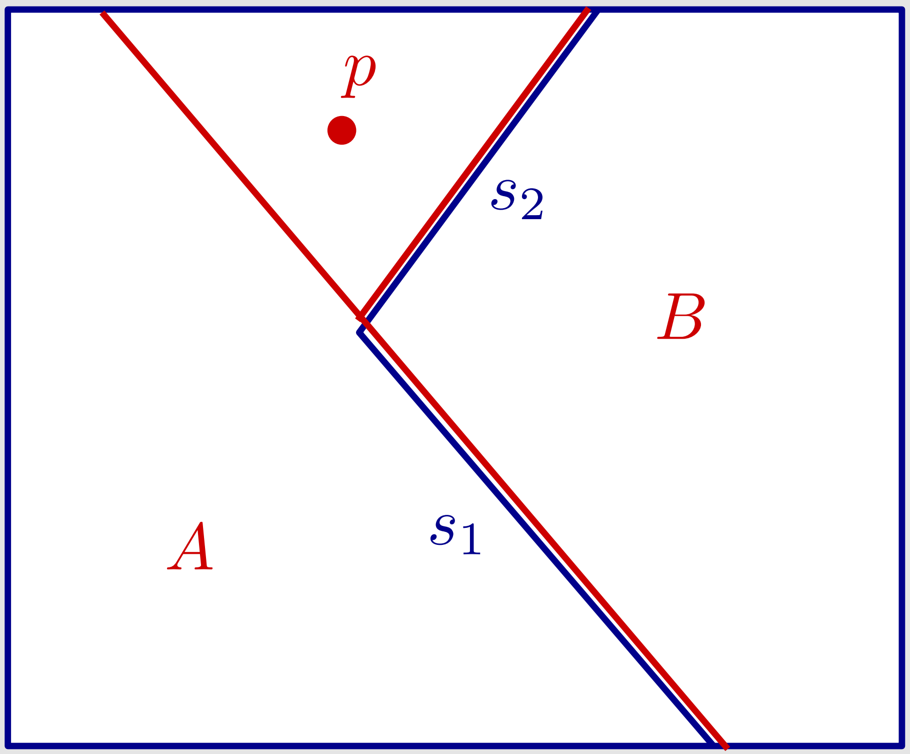
Point location (cont'd)



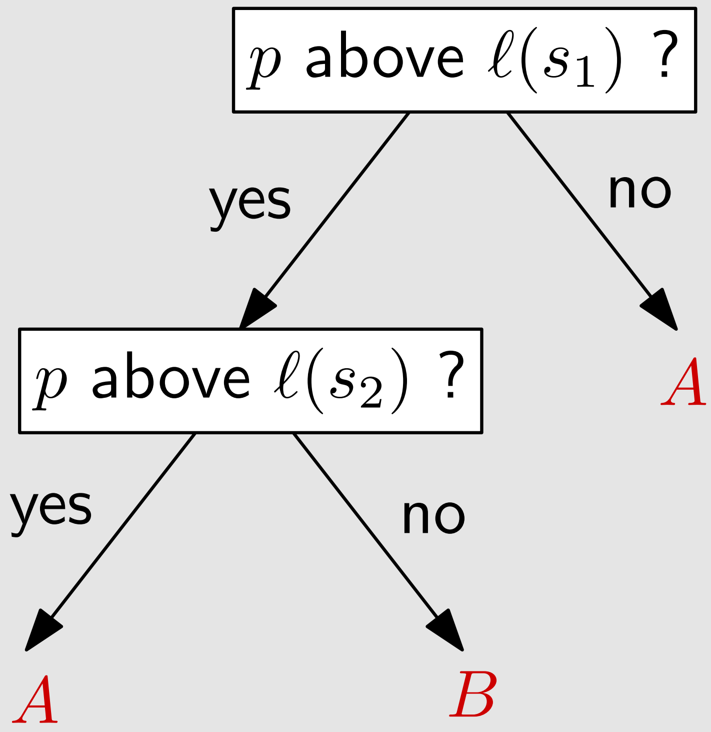
decision tree



Point location (cont'd)

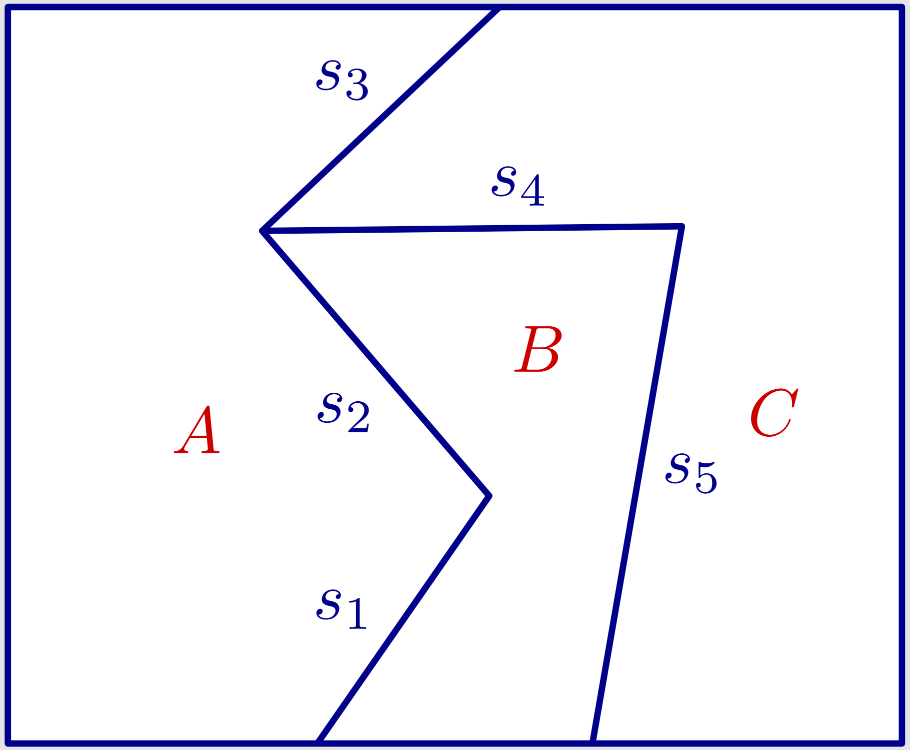


decision tree

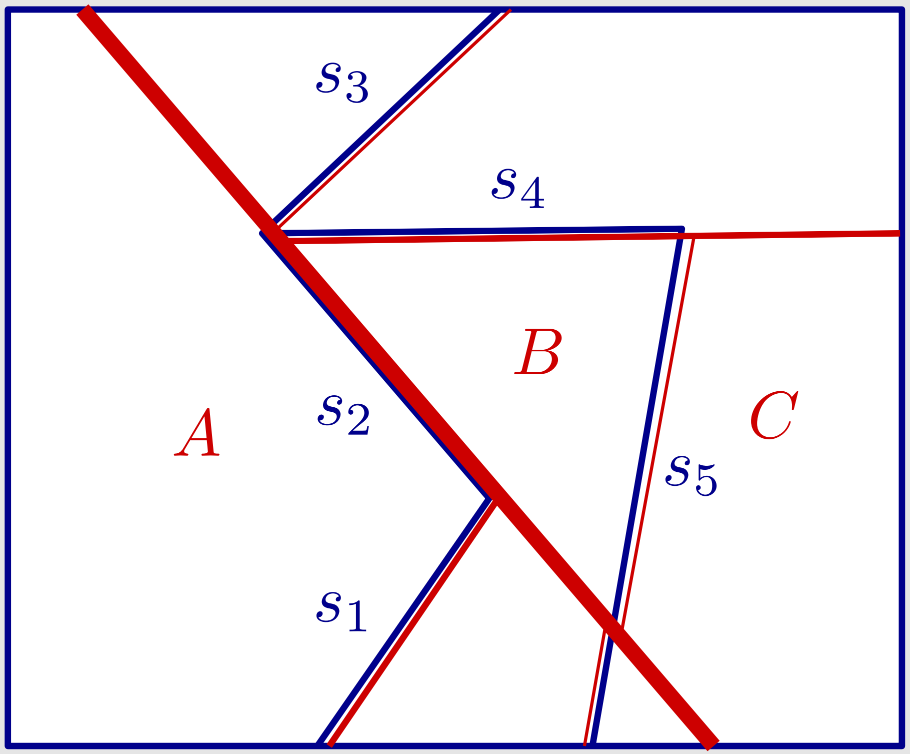


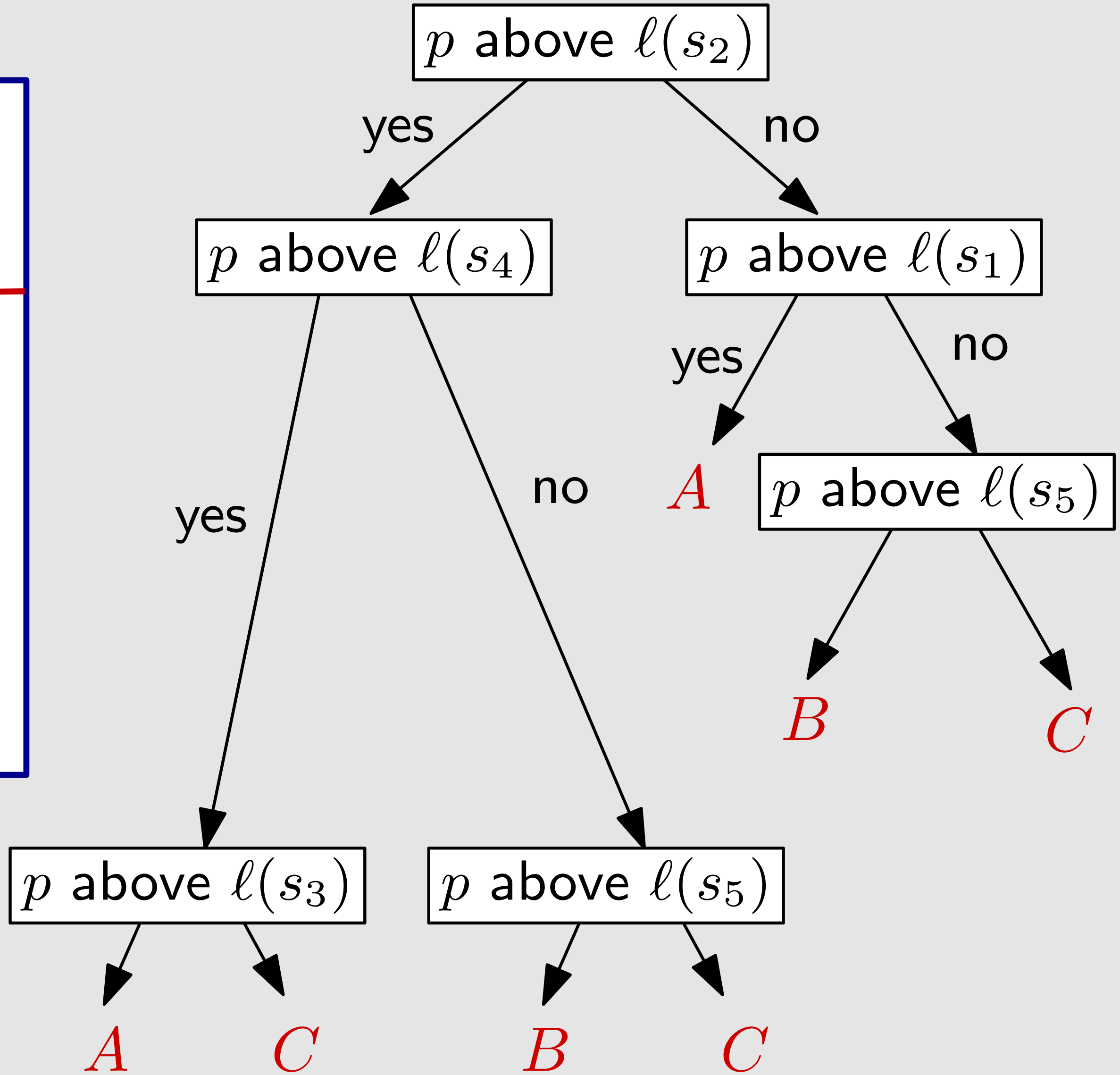
Compute decision tree once, use it to answer many queries.

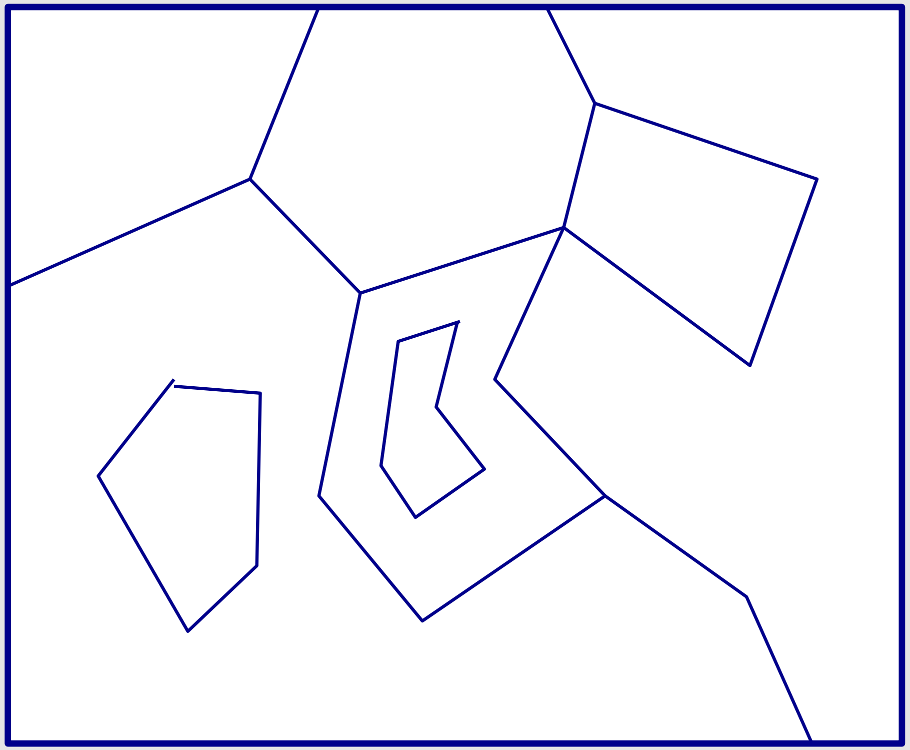
Point location (cont'd)



Point location (cont'd)



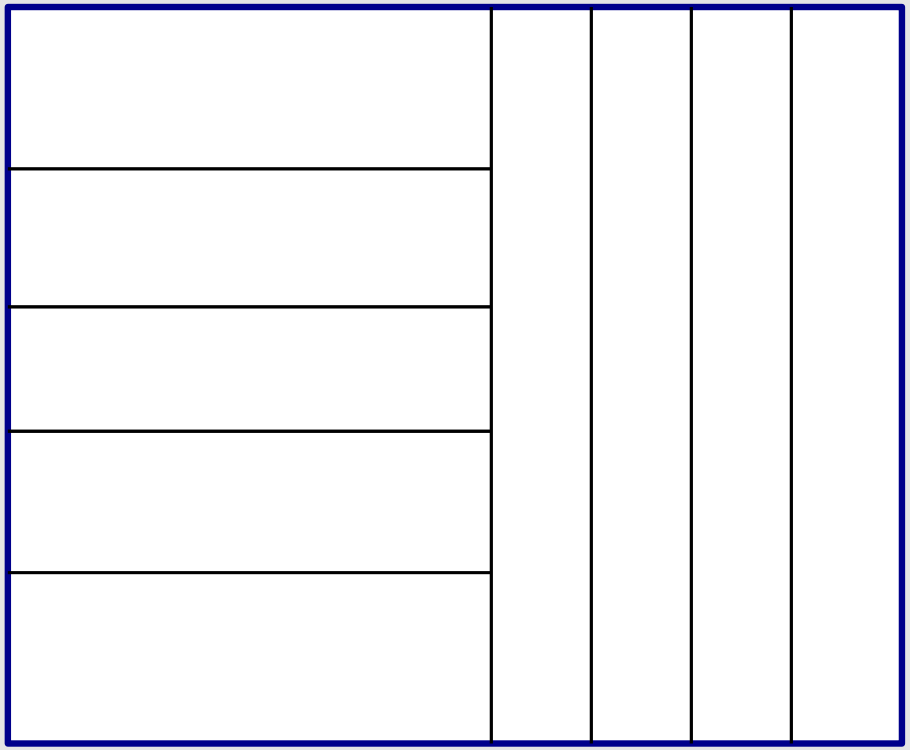




Combinatorial question:

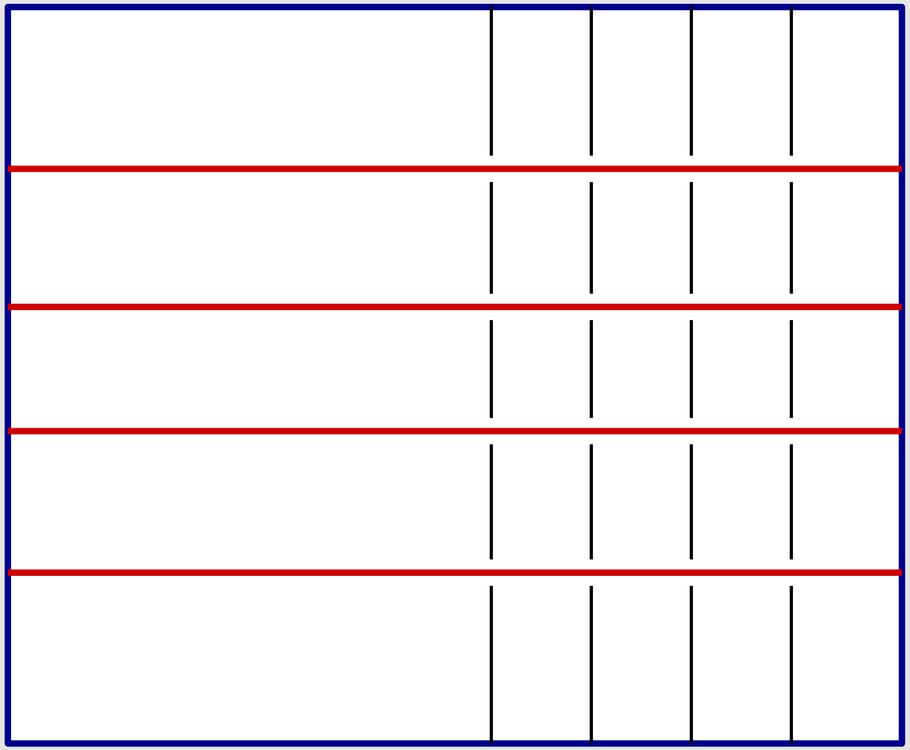
How small can we keep the decision tree for a subdivision with n segments?

Note: size of decision tree = number of fragments into which edges are cut

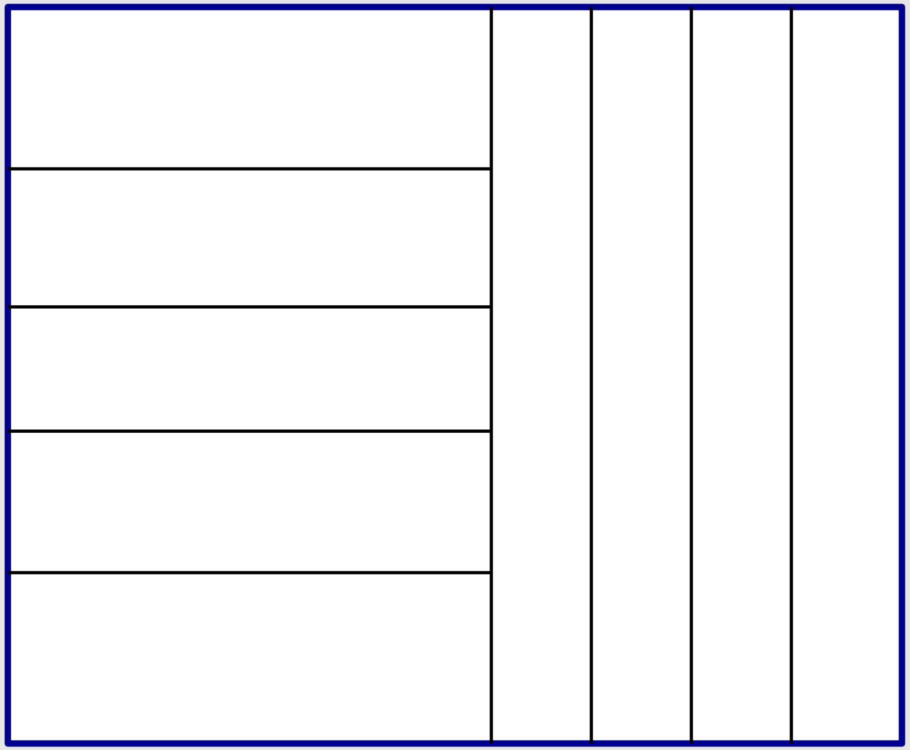


Point location (cont'd)

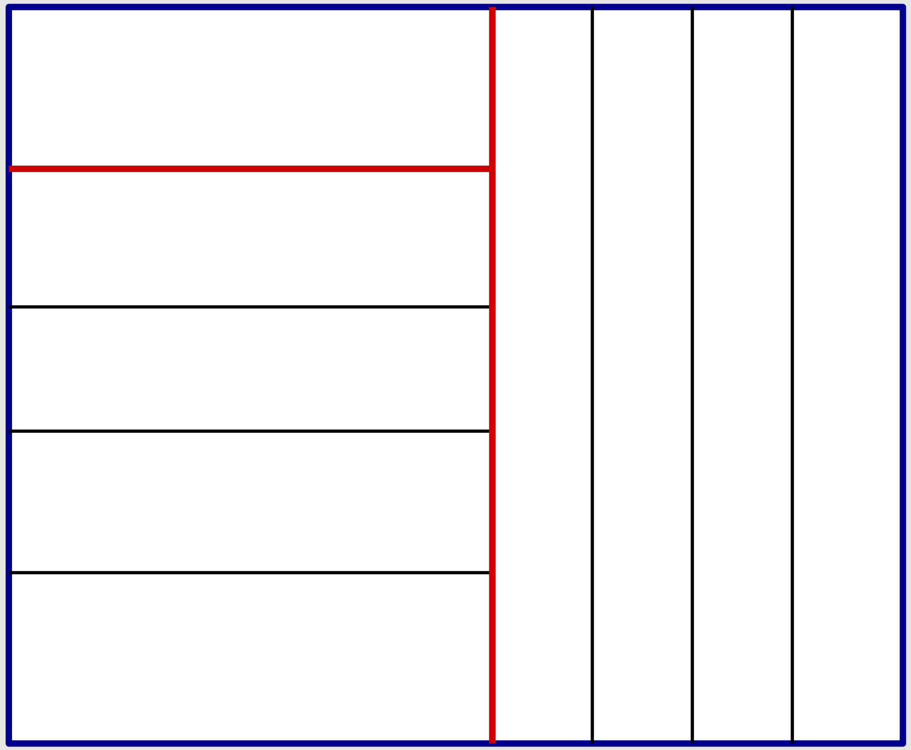
size can be $\sim n^2$

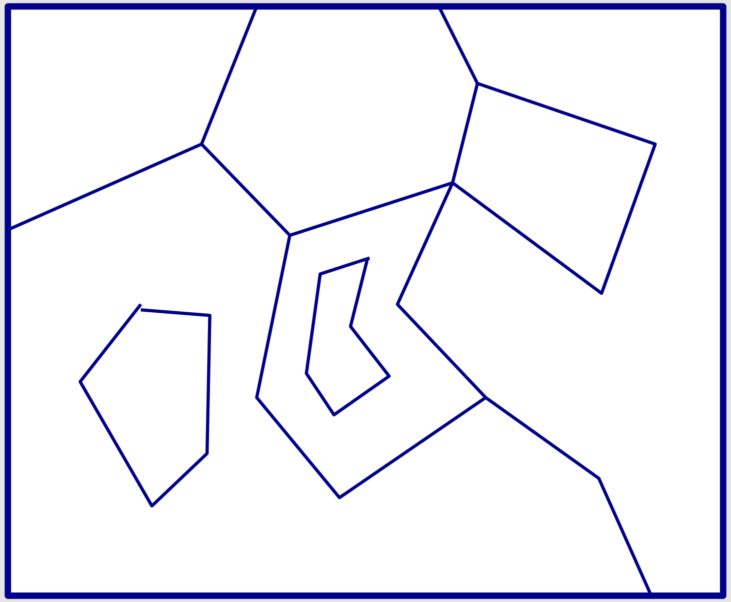


size can be $\sim n^2$



$\ldots$ or $\sim n$





Combinatorial question:

How small can we keep the decision tree for a subdivision with n segments?

Note: size of decision tree $=$ number of fragments into which edges are cut

Can we find, for any subdivision with n edges, an order for creating the decision tree such that the size of the tree is $\sim n$?

Answer: no, this is impossible, but we can get $\sim n \ln n$.

Algorithm for autopartitions: randomized version

Trick: use a *random* order on the segments!

Algorithm for autopartitions: randomized version

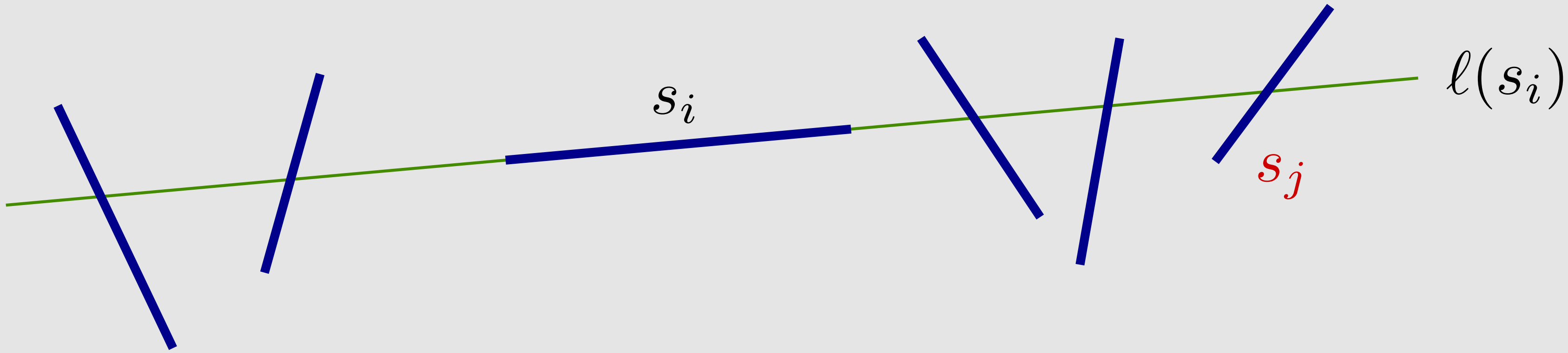
Trick: use a *random* order on the segments!

Theorem: Expected number of cuts is at most $2n \ln n$.

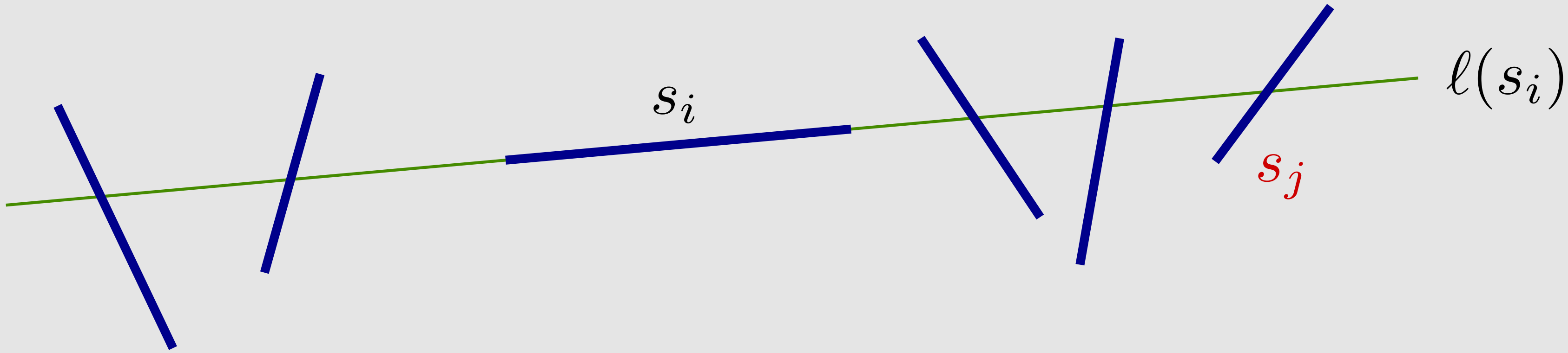
Proof:

$$
\begin{aligned}
E[\text{ number of cuts }] &= E[\textstyle\sum_{i=1}^{n} \text{(number of cuts made by } \ell(s_i))\ ] \\
&= \textstyle\sum_{i=1}^{n} E[\text{ number of cuts made by } \ell(s_i)\ ]
\end{aligned}
$$

How many cuts do we expect $\ell(s_i)$ to make?

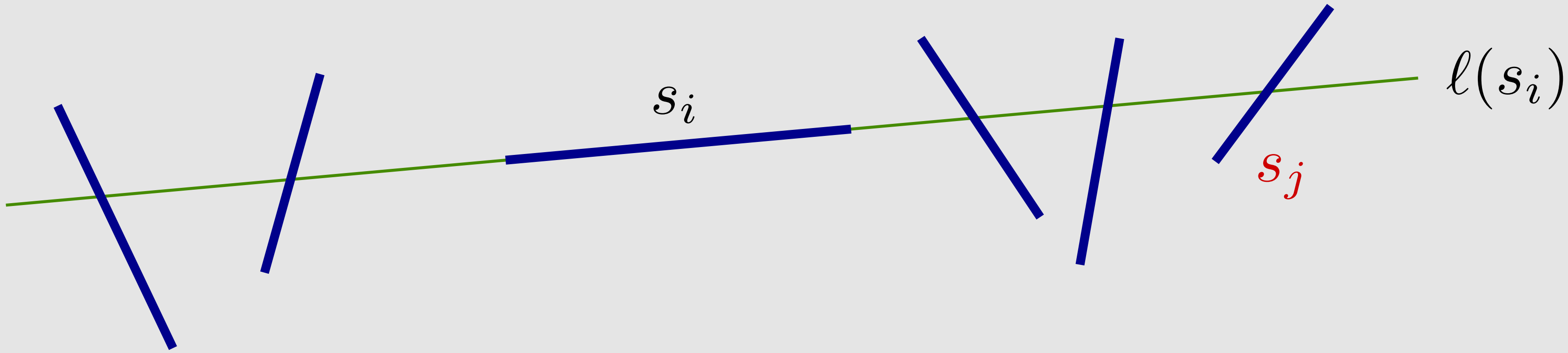


How many cuts do we expect $\ell(s_i)$ to make?



Pr[$\ell(s_i)$ cuts s_j] $=$

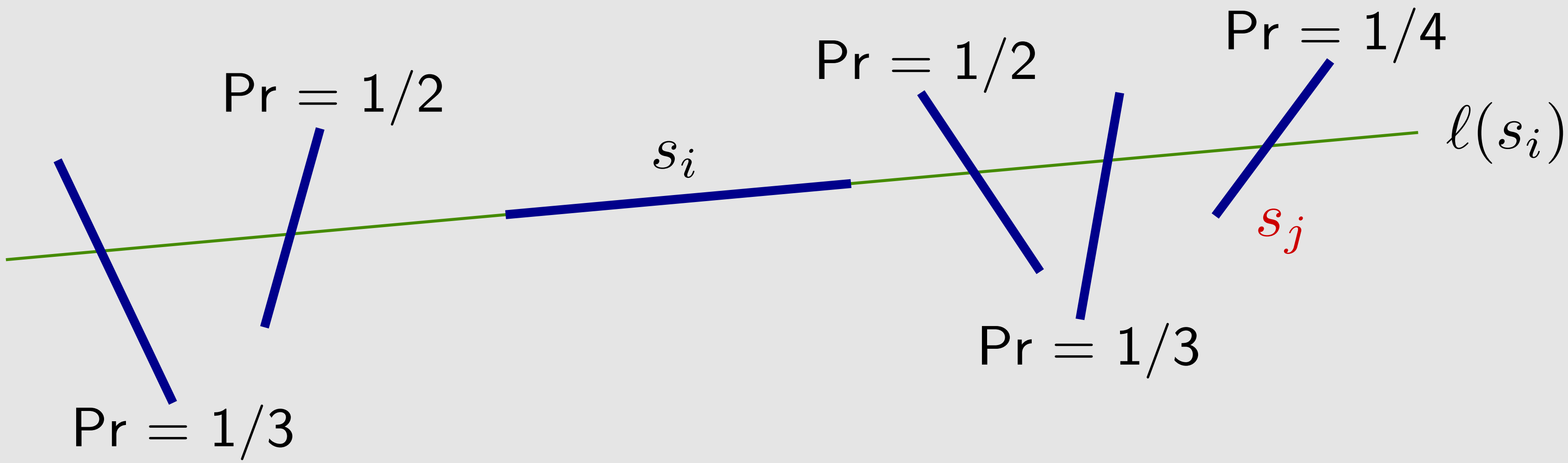
How many cuts do we expect $\ell(s_i)$ to make?



Pr[$\ell(s_i)$ cuts s_j] $= 1/4$

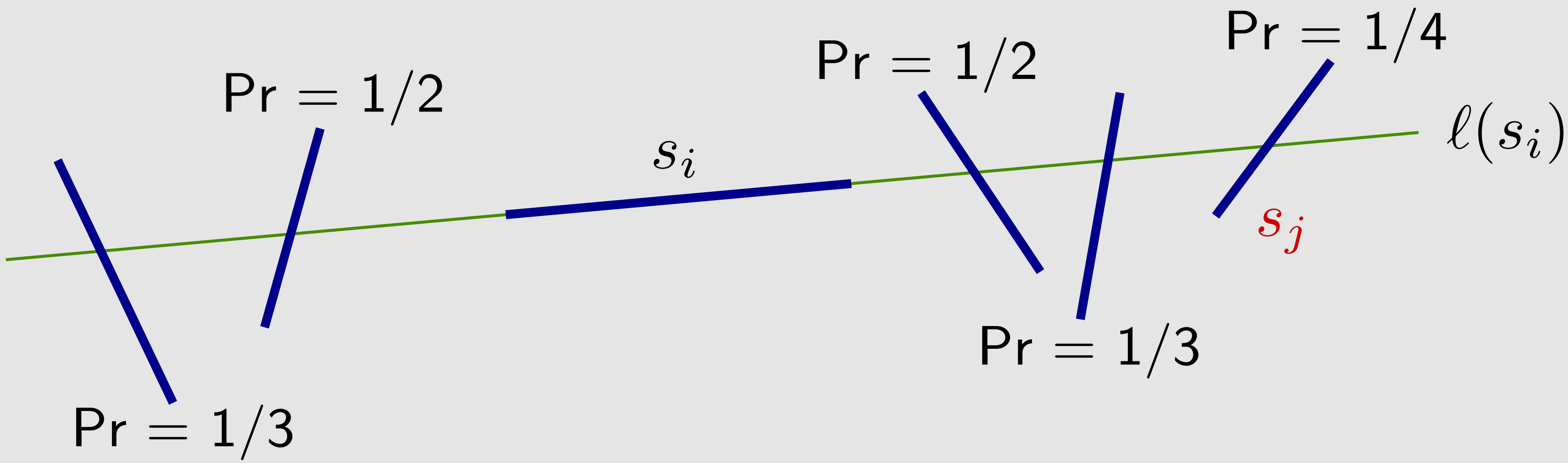
Algorithm for autopartitions: randomized version

How many cuts do we expect $\ell(s_i)$ to make?



Algorithm for autopartitions: randomized version

How many cuts do we expect $\ell(s_i)$ to make?



$$
\begin{aligned}
E[\text{ number of cuts made by } \ell(s_i)\ ] \quad &= \quad \textstyle\sum_j \Pr[\ell(s_i) \text{ cuts } s_j] \\
&\leqslant \quad 2 \cdot \left(\tfrac{1}{2} + \tfrac{1}{3} + \cdots + \tfrac{1}{n/2}\right) \\
&\approx \quad 2 \ln(n/2)
\end{aligned}
$$

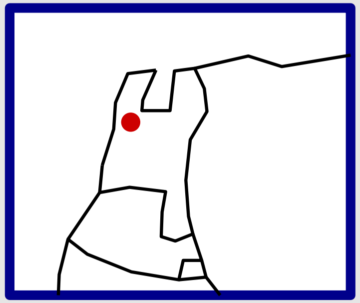
Theorem: Expected number of cuts is at most $2n \ln n$.

Proof:

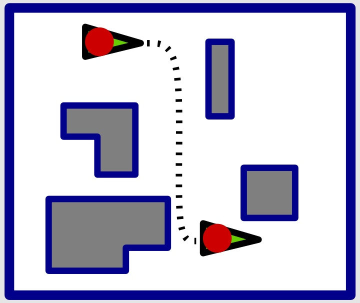
$$
\begin{aligned}
E[\text{ number of cuts }] &= E[\textstyle\sum_{i=1}^{n} (\text{number of cuts made by } \ell(s_i)) \,] \\
&= \textstyle\sum_{i=1}^{n} E[\text{ number of cuts made by } \ell(s_i) \,] \\
&\leqslant \textstyle\sum_{i=1}^{n} 2\ln(n/2) \\
&< 2n \ln n
\end{aligned}
$$

$\blacksquare$

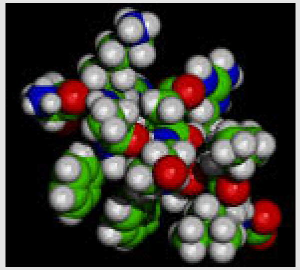
Computational geometry: algorithms for spatial data.



point location

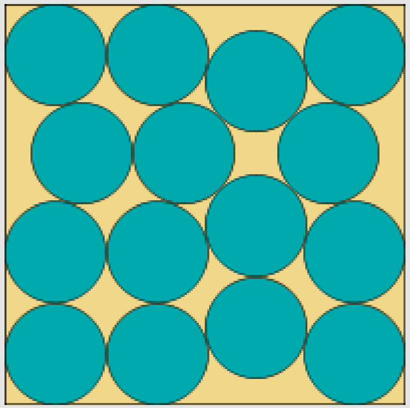


motion planning

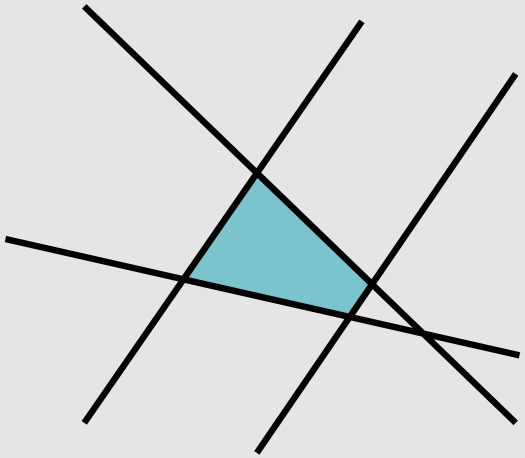


docking

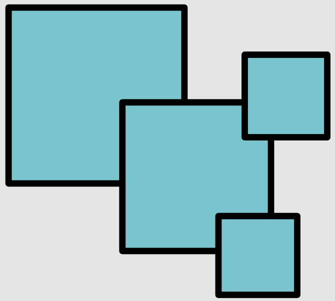
Combinatorial geometry: combinatorics for spatial data.



circle packings



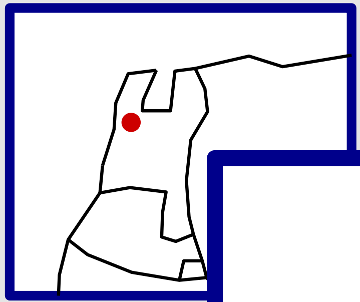
arrangements



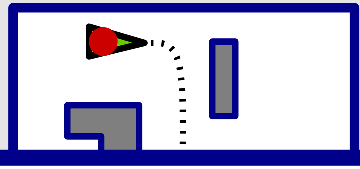
union complexity

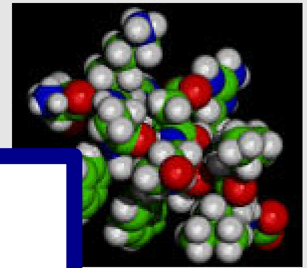
TU/e

Computational geometry: algorithms for spatial data.



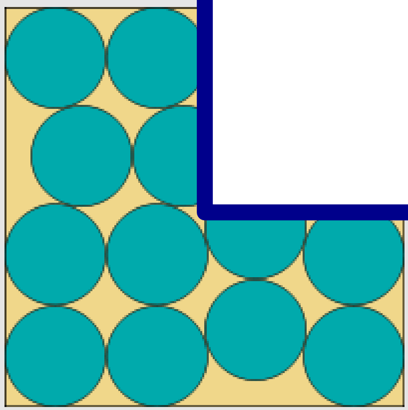
point lo



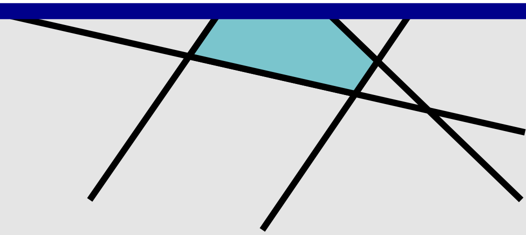


locking

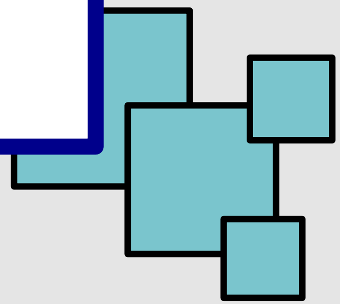
Combinatorial



circle packings



arrangements



union complexity

Many nice geometric puzzles . . .

. . . that are (sometimes) even useful !