RECITATION 1

Introducing CGAL, OMPL
Halfplanes' intersection

Michal Kleinbort

CGAL

- The Computational Geometry Algorithms Library (CGAL) is an open source C++ library providing implementations for many algorithms and data structures in computational geometry
- Implemented algorithms are efficient and reliable
- Allows for exact computation (avoiding roundoff errors)
- Several packages of CGAL have Python bindings

https://www.cgal.org/index.html

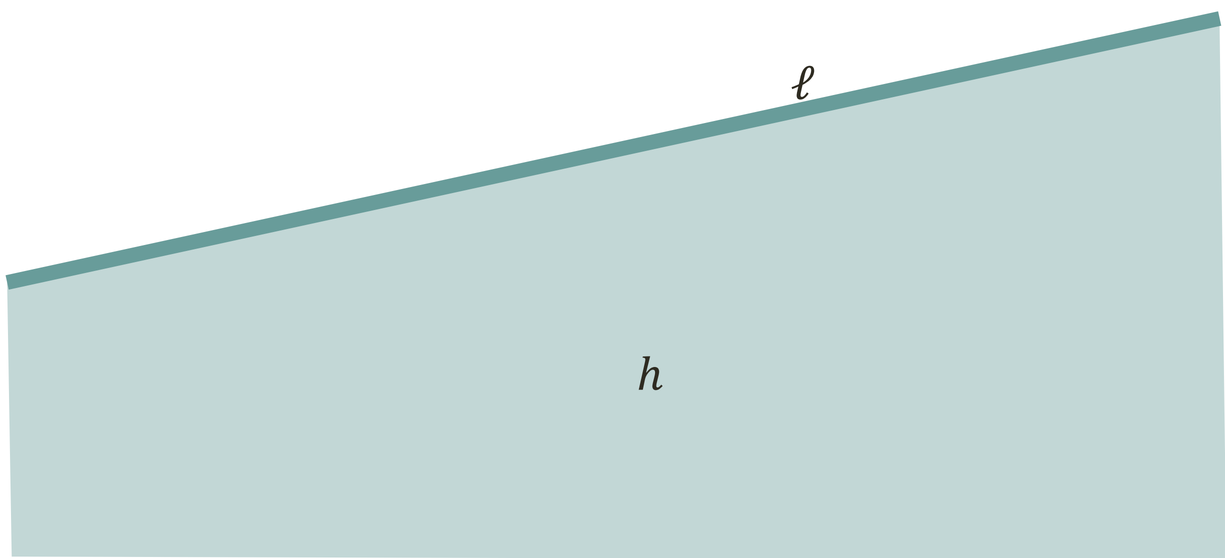
OMPL

- The Open Motion Planning Library (OMPL) is an open source C++ library providing implementations to many state-of-the-art sampling-based motion planning algorithms.
- OMPL.app builds upon OMPL and specifies geometric representation for the robot and its environment. It makes use of open-source collision checking libraries.
- Has also been integrated with ROS (Robot Operating System), which is a collection of frameworks for robot software development.

- https://ompl.kavrakilab.org/

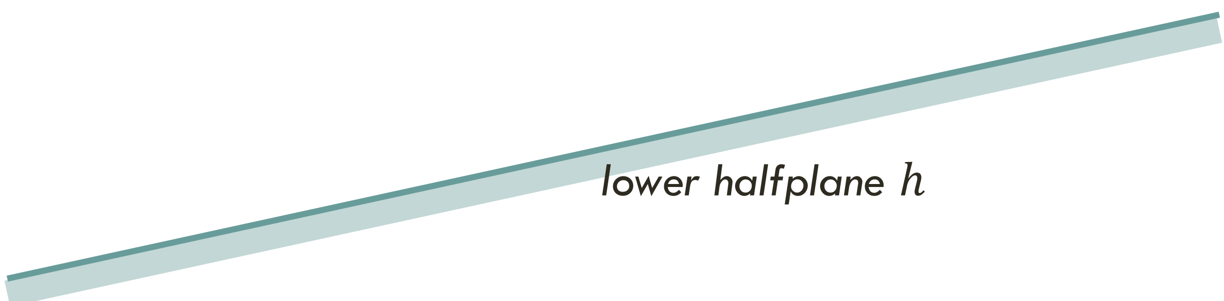
HALFPLANE (DEFINITION)

- a planar region h consisting of all points on **one side** of an (infinite) line ℓ
- ***lower halfplane*** h *is represented as* $y \leq ax + b$



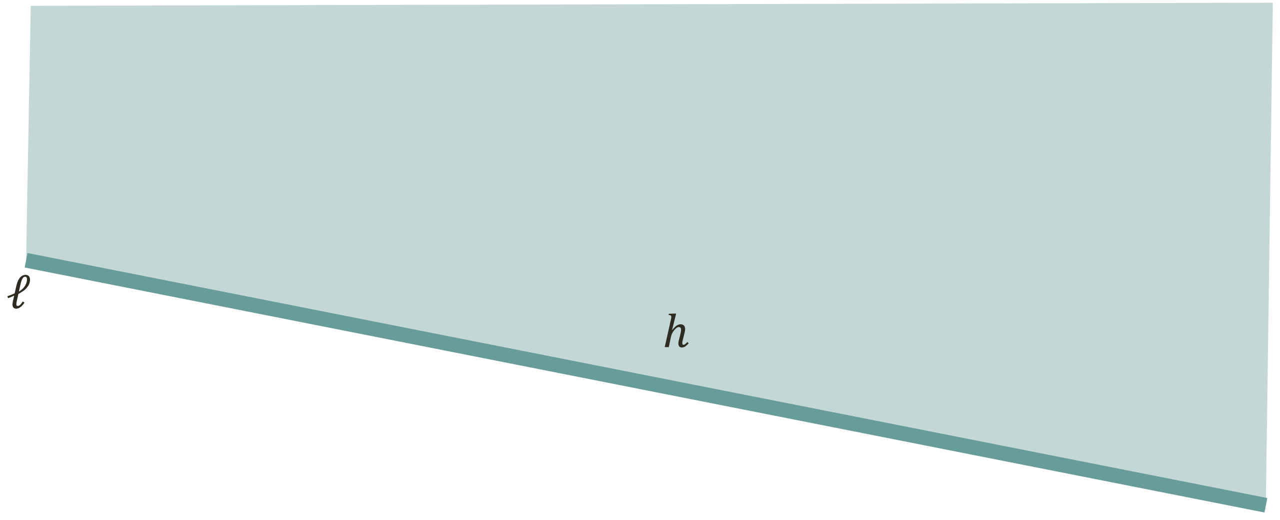
HALFPLANE (DEFINITION)

- a planar region h consisting of all points on one side of an (infinite) line ℓ
- ***lower halfplane*** h *is represented as* $y \leq ax + b$



HALFPLANE (DEFINITION)

- a planar region h consisting of all points on **one side** of an (infinite) line ℓ
- ***upper halfplane*** h *is represented as* $y \geq ax + b$

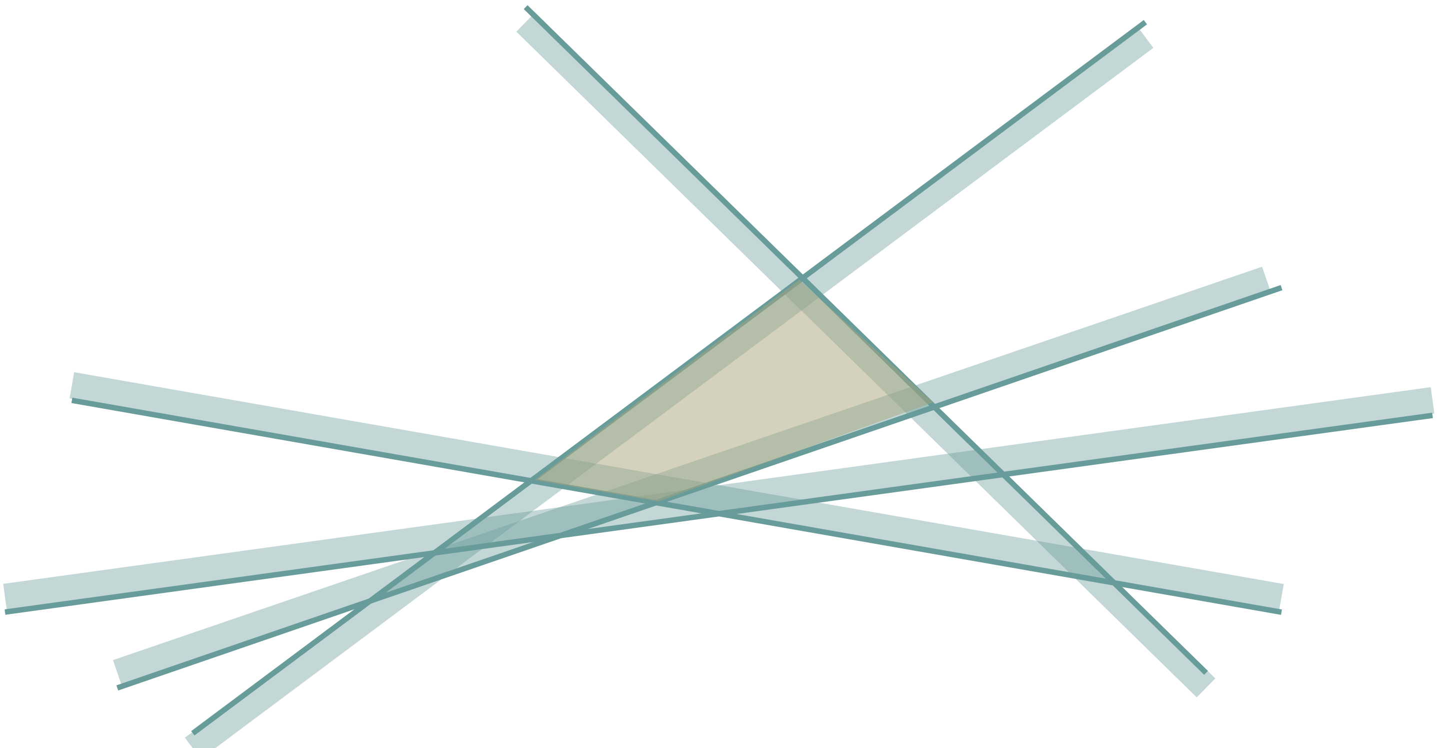


HALFPLANE (DEFINITION)

- a planar region h consisting of all points on one side of an (infinite) line ℓ
- ***upper halfplane*** h *is represented as* $y \geq ax + b$

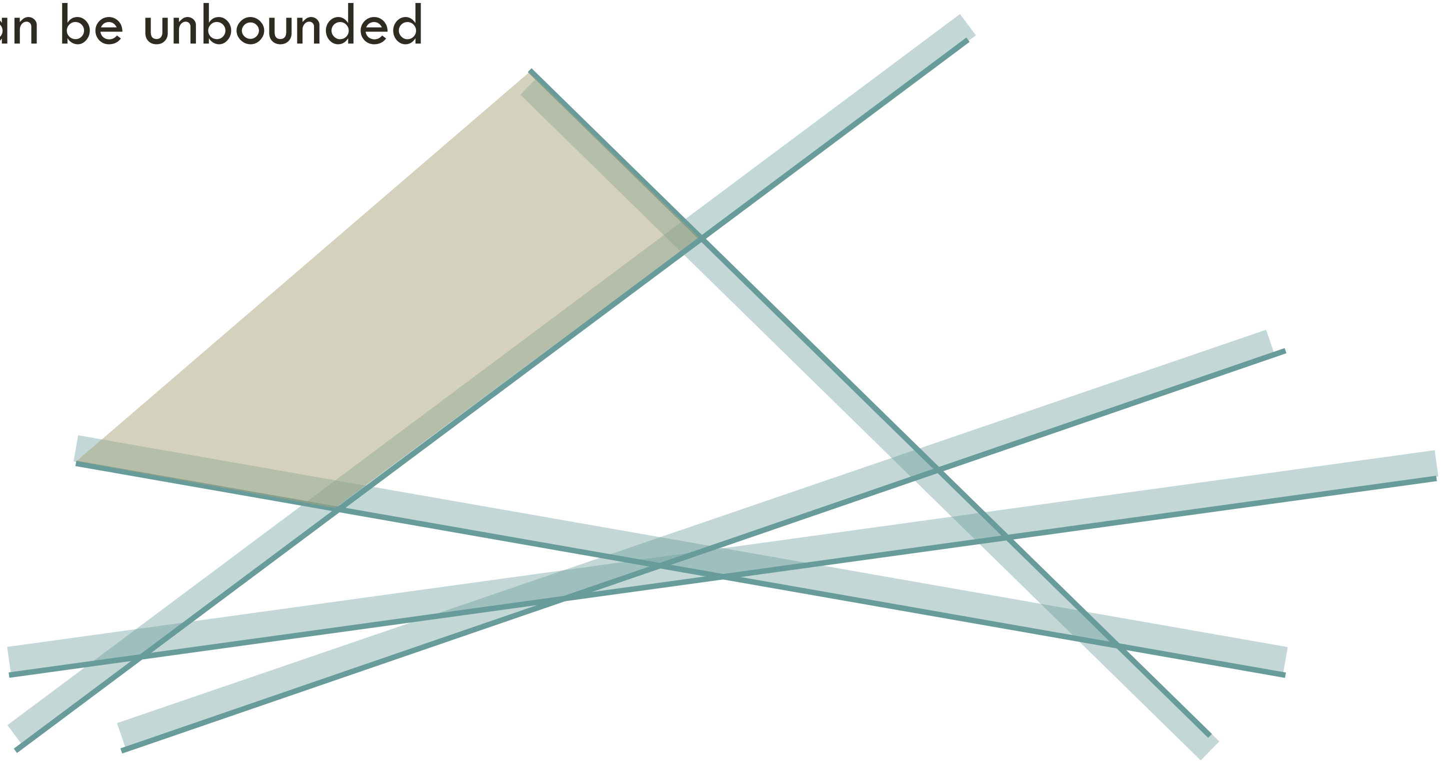
INTERSECTION OF HALFPLANES

is always convex (why?)



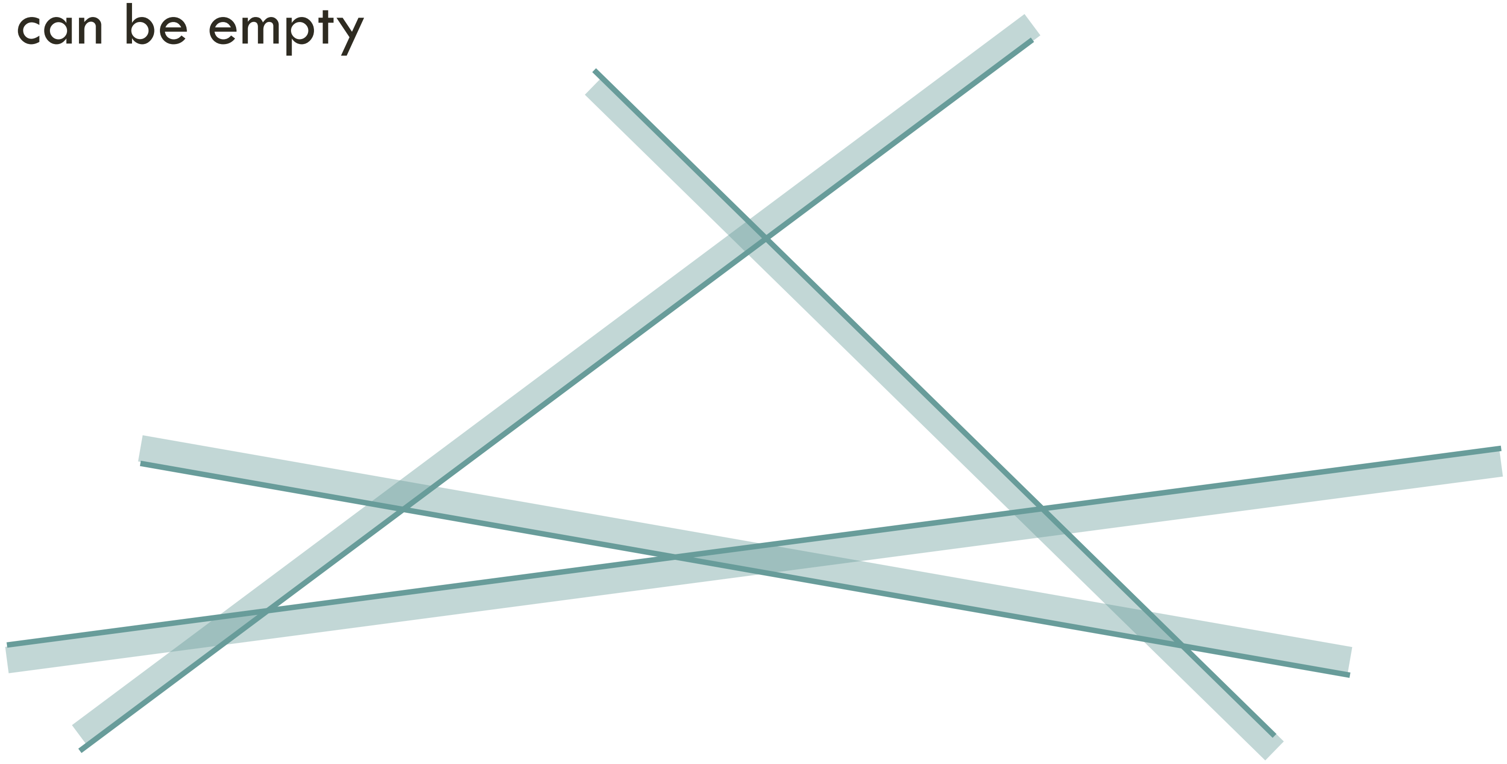
INTERSECTION OF HALFPLANES

can be unbounded



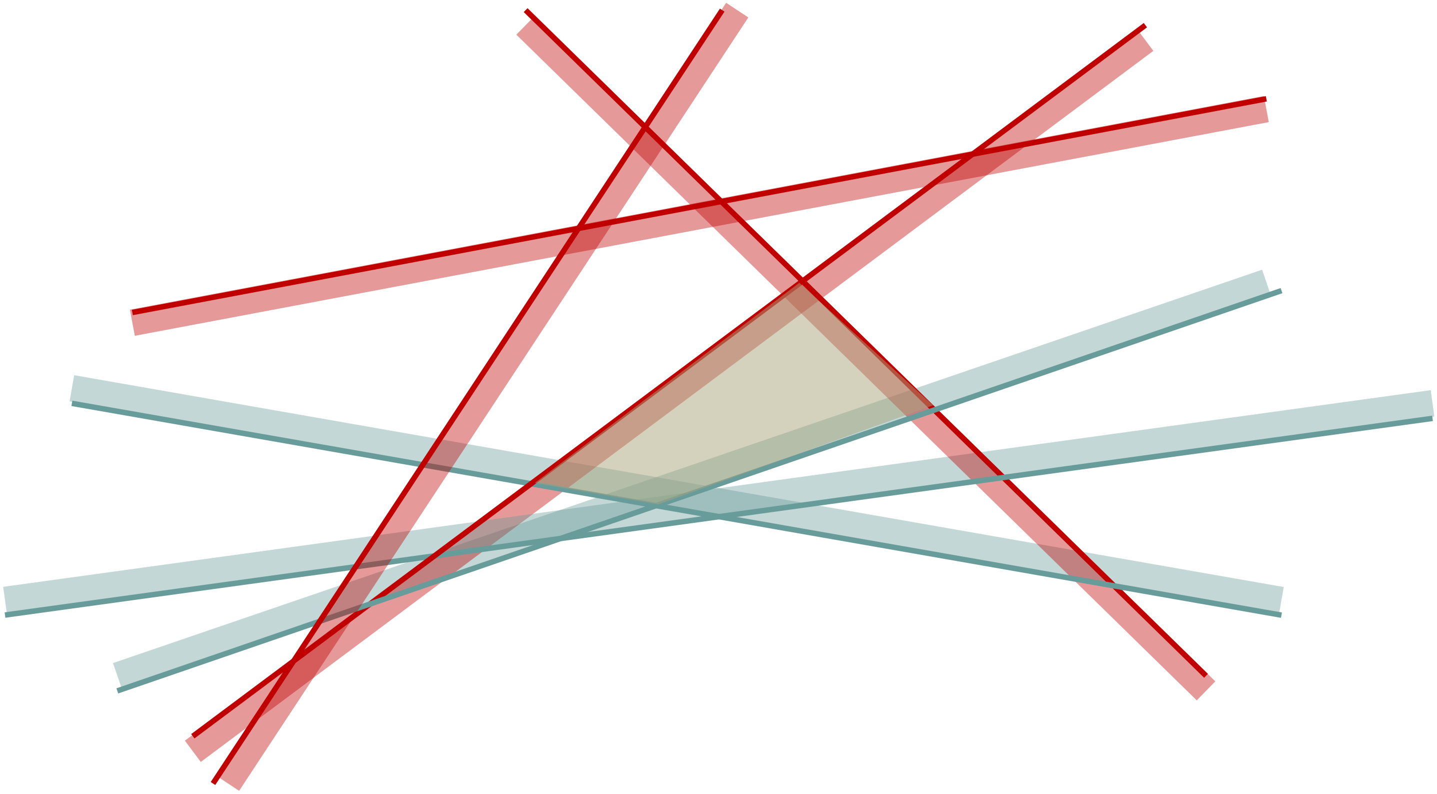
INTERSECTION OF HALFPLANES

can be empty



INTERSECTION OF HALFPLANES

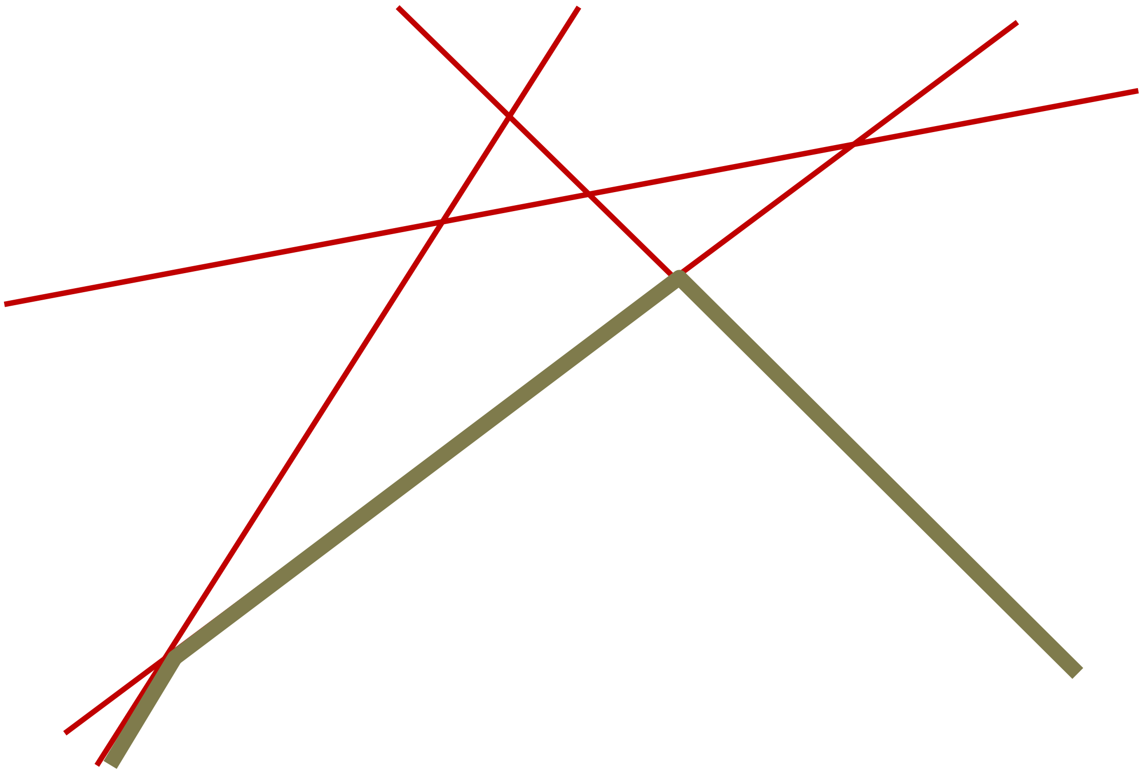
Splitting to upper and lower halfplanes



THE LOWER ENVELOPE OF LINES

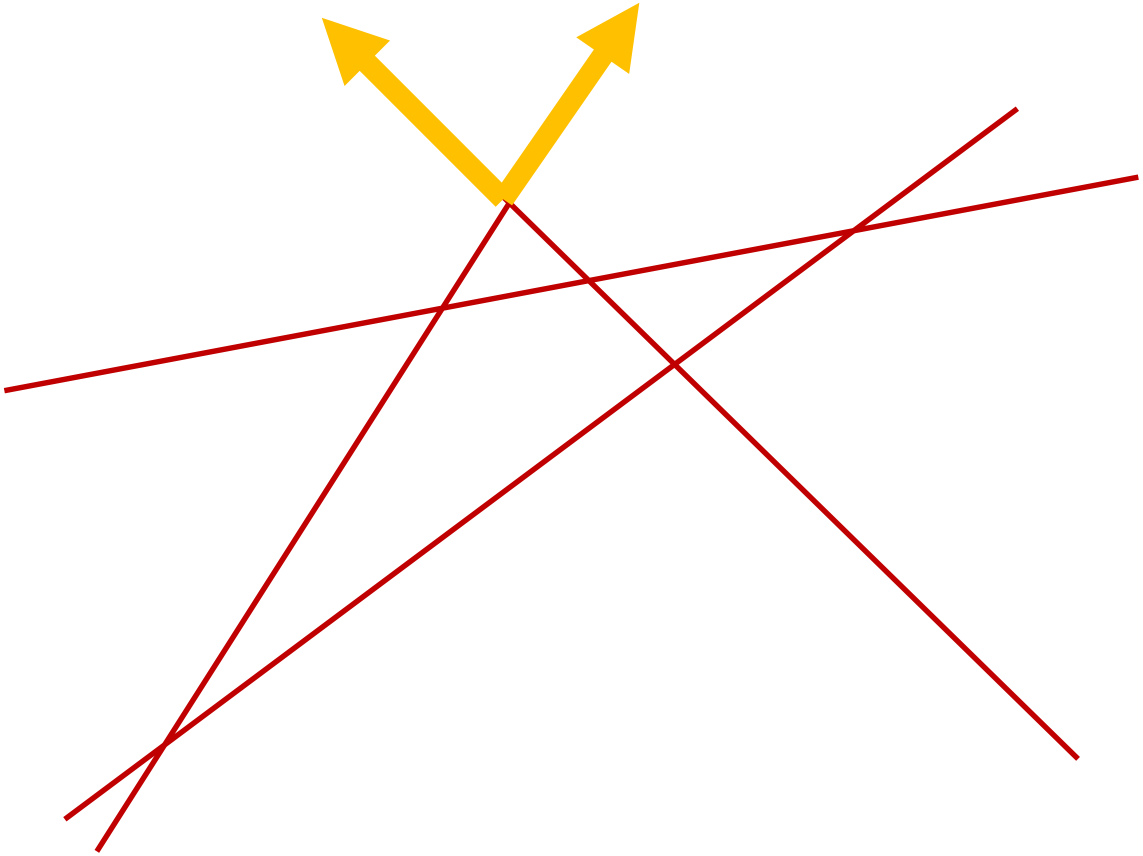
Let L be a set of lines

The *lower envelope* of L is $f(x) = min_{\ell \in L}\, \ell(x)$



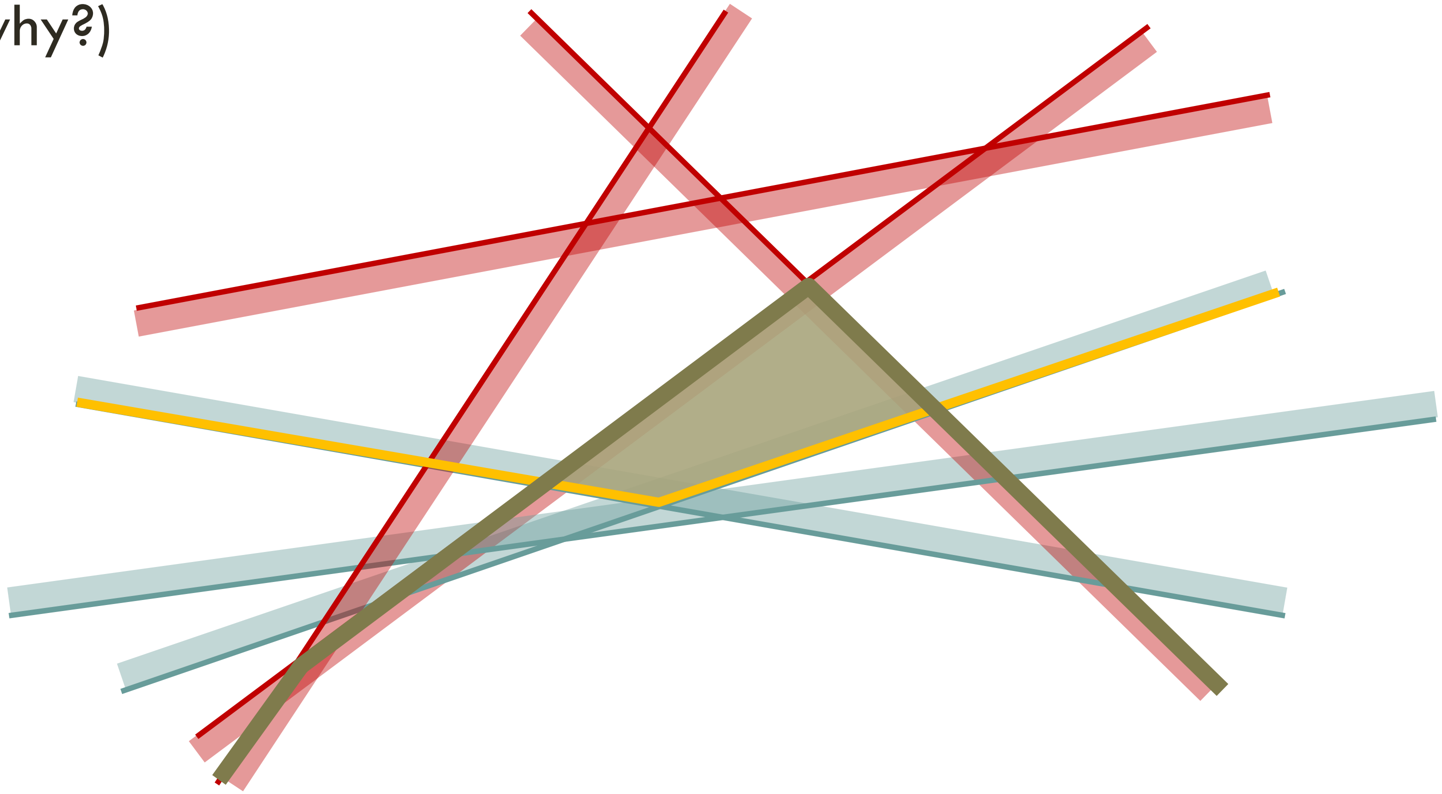
THE UPPER ENVELOPE OF LINES

The *upper envelope* is defined similarly



INTERSECTION OF HALFPLANES

The region below the lower envelope of the lower halfplanes and above the upper envelope of the upper halfplanes (why?)



ALGORITHM FOR COMPUTING THE INTERSECTION

Given a set of halfplanes H

1) Split H into 3 subsets:
   - H_L the lower halfplanes
   - H_U the upper halfplanes
   - H_{Vert} the vertical halfplanes
2) Compute the E_L - the lower envelope of H_L
3) Compute the E_U- the upper envelope of H_U
4) Compute the region bounded between the two envelopes and intersect it with the rightmost right-halfplane and leftmost left-halfplane in H_{Vert}, if such exist

ALGORITHM FOR PART (4): COMPUTING THE BOUNDED REGION

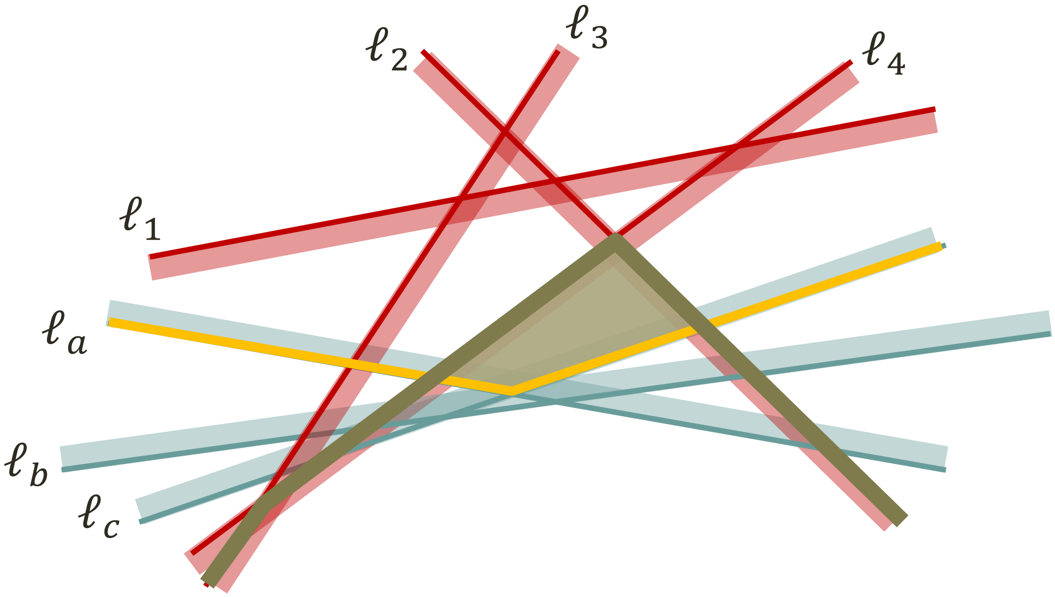
Given two envelopes E_U, E_L (upper and lower) represented as ordered lists of lines.

The goal is to compute the bounded region below E_L and above E_U.

The output should be two sub-lists of E_L and E_U, representing the upper and lower boundary of the region, respectively.

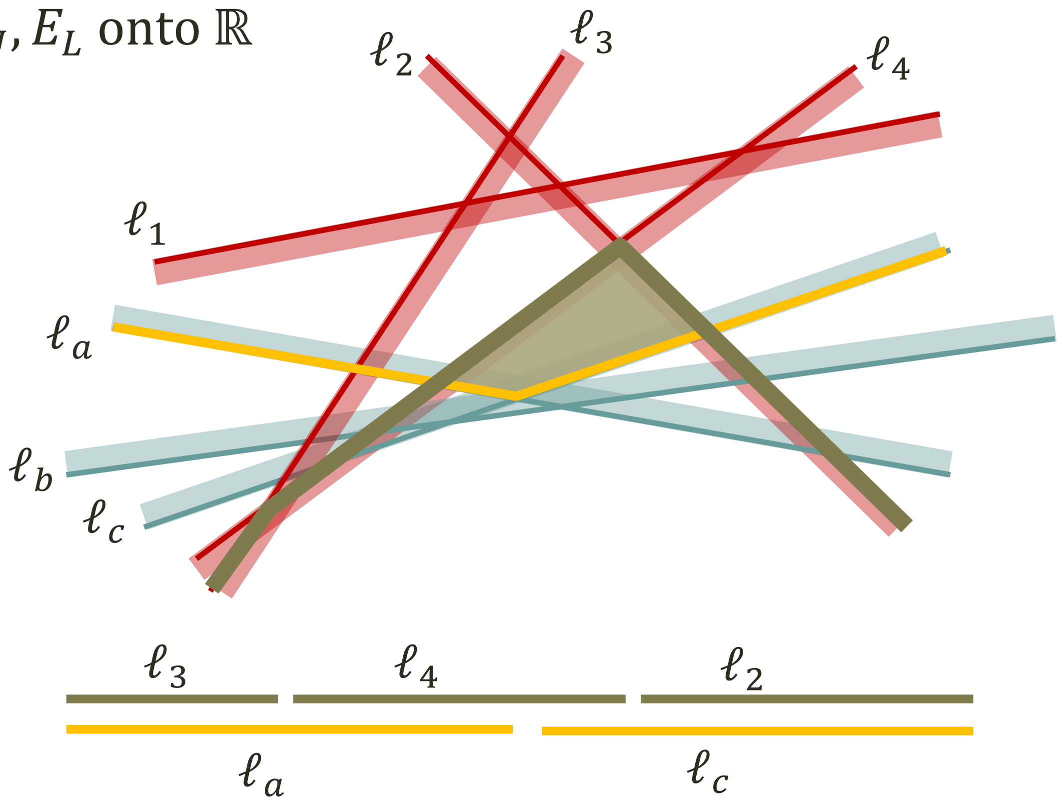
ALGORITHM FOR PART (4): COMPUTING THE BOUNDED REGION

$E_U = [\ell_a, \ell_c]$ $\quad$ $E_L = [\ell_3, \ell_4, \ell_2]$



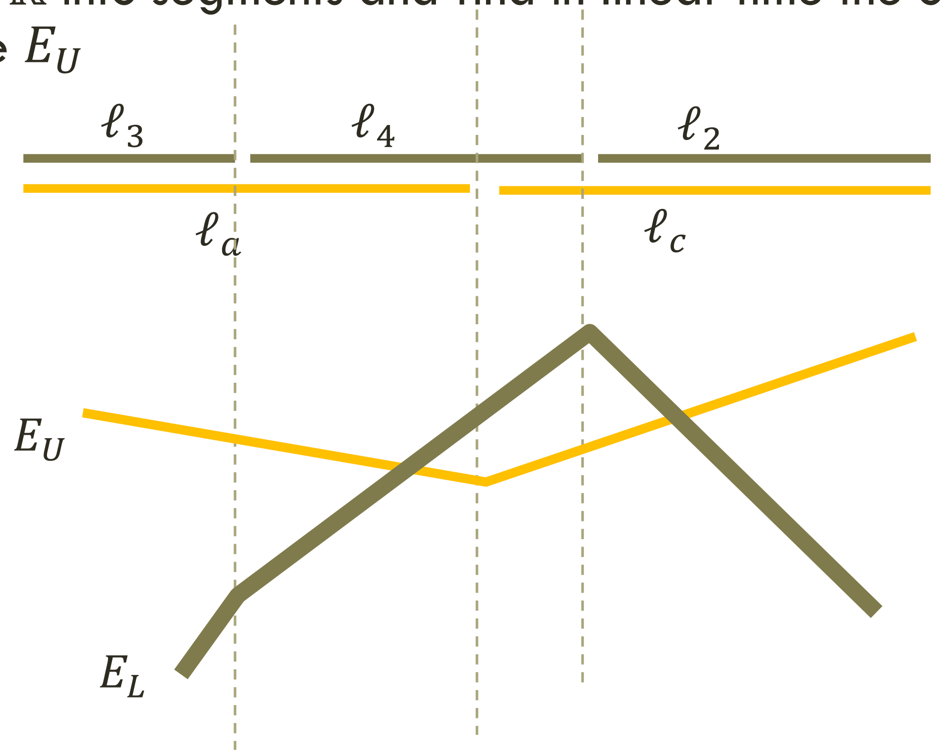
ALGORITHM FOR PART (4): COMPUTING THE BOUNDED REGION

Project E_U, E_L onto $\mathbb{R}$



ALGORITHM FOR PART (4): COMPUTING THE BOUNDED REGION

We partition $\mathbb{R}$ into segments and find in linear time the ones where E_L lies above E_U



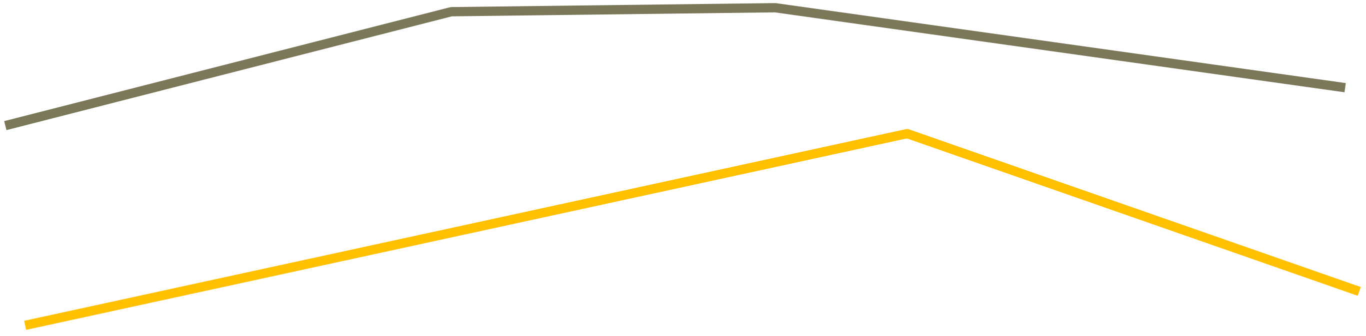
COMPUTING THE LOWER ENVELOPE

Divide and Conquer algorithm for a given set of n lines L:

1) Divide L into two subsets A, B of size $n/2$ each
2) Run the D&C alg on A, B separately returning $E^A$and E^B
3) Merge $E^A$and E^B into a new lower envelope E
4) Return E

PART (3): MERGING TWO ENVELOPES

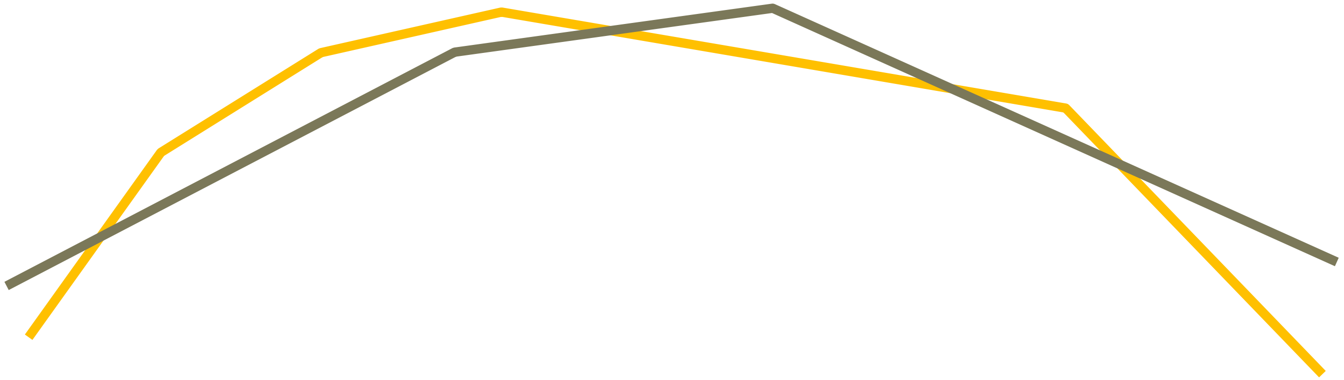
Case 1 (simple): the envelopes do not intersect



return the lower one

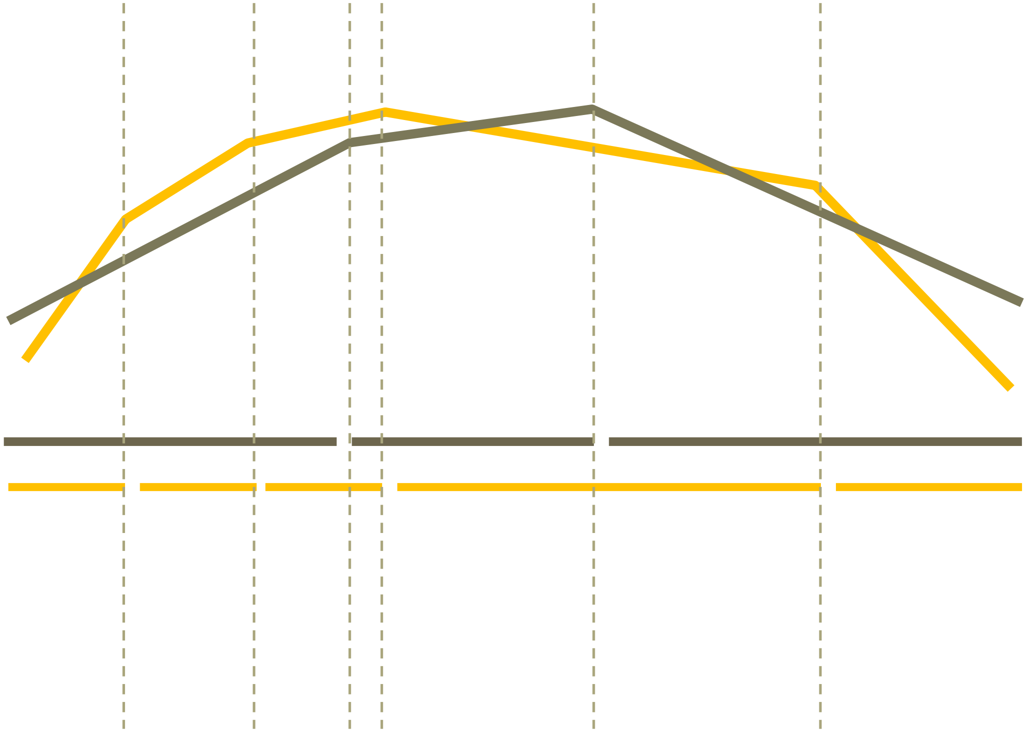
PART (3): MERGING TWO ENVELOPES

Case 2: the envelopes intersect



Project E_A, E_B onto $\mathbb{R}$

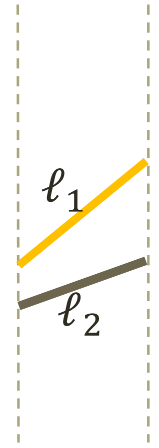
PART (3): MERGING TWO ENVELOPES



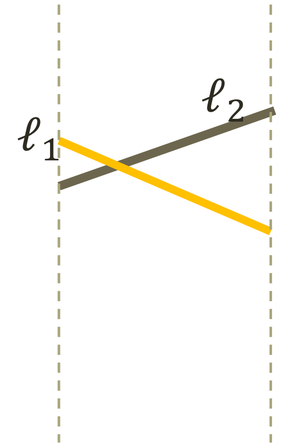
We partition $\mathbb{R}$ into segments, each segment is defined by two line segments: one from each envelope

PART (3): MERGING TWO ENVELOPES

Two options for segments:



Add ℓ_2 to the resulting envelope



Add ℓ_2 to the resulting envelope and then ℓ_1

COMPLEXITY

- Merging two envelopes takes $O(n)$ where n is the size of the longer envelope
- Note that if the initial set of lines is of size n then the lower envelope is of size $O(n)$ (each line can appear at most once on the envelope)
- Complexity of the D&C algorithm for computing the lower envelope is $O(n \log n)$
  - This is optimal in the worst-case
  - Output sensitive algorithms, which may perform better for certain inputs, exist as well
- Complexity of the algorithm for computing the intersection of halfspaces is $O(n \log n)$