LEMONADE STAND

- YOU HAVE ONE HOUR TO SELL YOUR PRODUCT
- ASSUME EVERYTHING YOU MAKE WILL BE SOLD
- 2 PRODUCTS : REGULAR LEMONADE & SPECIAL LEMONADE
- EVERYTHING IS FRESH : YOU SQUEEZE LEMONS ON THE SPOT
  - ↳ IT TAKES TIME TO MAKE LEMONADE
- YOU HAVE A FIXED AMOUNT OF INGREDIENTS
- YOU WANT TO MAXIMIZE PROFIT DURING THIS HOUR
  - ↳ DON'T CARE ABOUT LEFTOVER INVENTORY

	REGULAR	SPECIAL
PROFIT	1	2

	REGULAR	SPECIAL
PROFIT	1	2
SUPPLIES → used ↓ available		
TIME 60	0.25	0.5

	REGULAR	SPECIAL
PROFIT	1	2

SUPPLIES → used ↓ available		
TIME 60	0.25	0.5
LEMONS 200	1	1

		REGULAR	SPECIAL
PROFIT		1	2
SUPPLIES → used ↓ available			
TIME	60	0.25	0.5
LEMONS	200	1	1
SUGAR	250	2	1.25

		REGULAR	SPECIAL
PROFIT		1	2
SUPPLIES → used ↓ available			
TIME	60	0.25	0.5
LEMONS	200	1	1
SUGAR	250	2	1.25
WATER	240	0.5	0.6

This will accidentally turn into a 2 on the next slide.
It's just a typo and doesn't really matter.
I haven't corrected it because the video still shows this glitch

		REGULAR	SPECIAL
PROFIT		1	2
SUPPLIES → used ↓ available			
TIME	60	0.25	0.5
LEMONS	200	1	1
SUGAR	250	2	1.25
WATER	240	2	0.6
VODKA	50	0	0.5

		x REGULAR	y SPECIAL
PROFIT		1	2
maximize		x + 2y	
SUPPLIES → used	available		
TIME	60	0.25	0.5
LEMONS	200	1	1
SUGAR	250	2	1.25
WATER	240	2	0.6
VODKA	50	0	0.5

	x REGULAR	y SPECIAL
PROFIT	1	2

maximize $x + 2y$

SUPPLIES → used

↓

available

subject to :

TIME	60	$\geq$	$0.25x$	+	$0.5y$
LEMONS	200	$\geq$	$1x$	+	$1y$
SUGAR	250	$\geq$	$2x$	+	$1.25y$
WATER	240	$\geq$	$2x$	+	$0.6y$
VODKA	50	$\geq$	0		$0.5y$

$x, y \geq 0$

	x	y
	REGULAR	SPECIAL
PROFIT	1	2

maximize $x + 2y$

SUPPLIES → used

↓

available

subject to :

TIME	60	$\geq$	$0.25x$	$+$	$0.5y$	
LEMONS	200	$\geq$	$1x$	$+$	$1y$	← at most 200 glasses total
SUGAR	250	$\geq$	$2x$	$+$	$1.25y$	
WATER	240	$\geq$	$2x$	$+$	$0.6y$	
VODKA	50	$\geq$	0		$0.5y$	← at most 100 special glasses

$x, y \geq 0$

	REGULAR	SPECIAL
	x	y
PROFIT	1	2

maximize $x + 2y$

SUPPLIES → used

↓

available

subject to :

TIME	60	$\geq$	$0.25x$	+	$0.5y$	
LEMONS	200	$\geq$	$1x$	+	$1y$	
SUGAR	250	$\geq$	$2x$	+	$1.25y$	
WATER	240	$\geq$	$2x$	+	$0.6y$	
VODKA	50	$\geq$	0		$0.5y$	

$x, y \geq 0$

LINEAR PROGRAMMING

	x	y
	REGULAR	SPECIAL
PROFIT	1	2

maximize $x + 2y$

SUPPLIES → used

↓

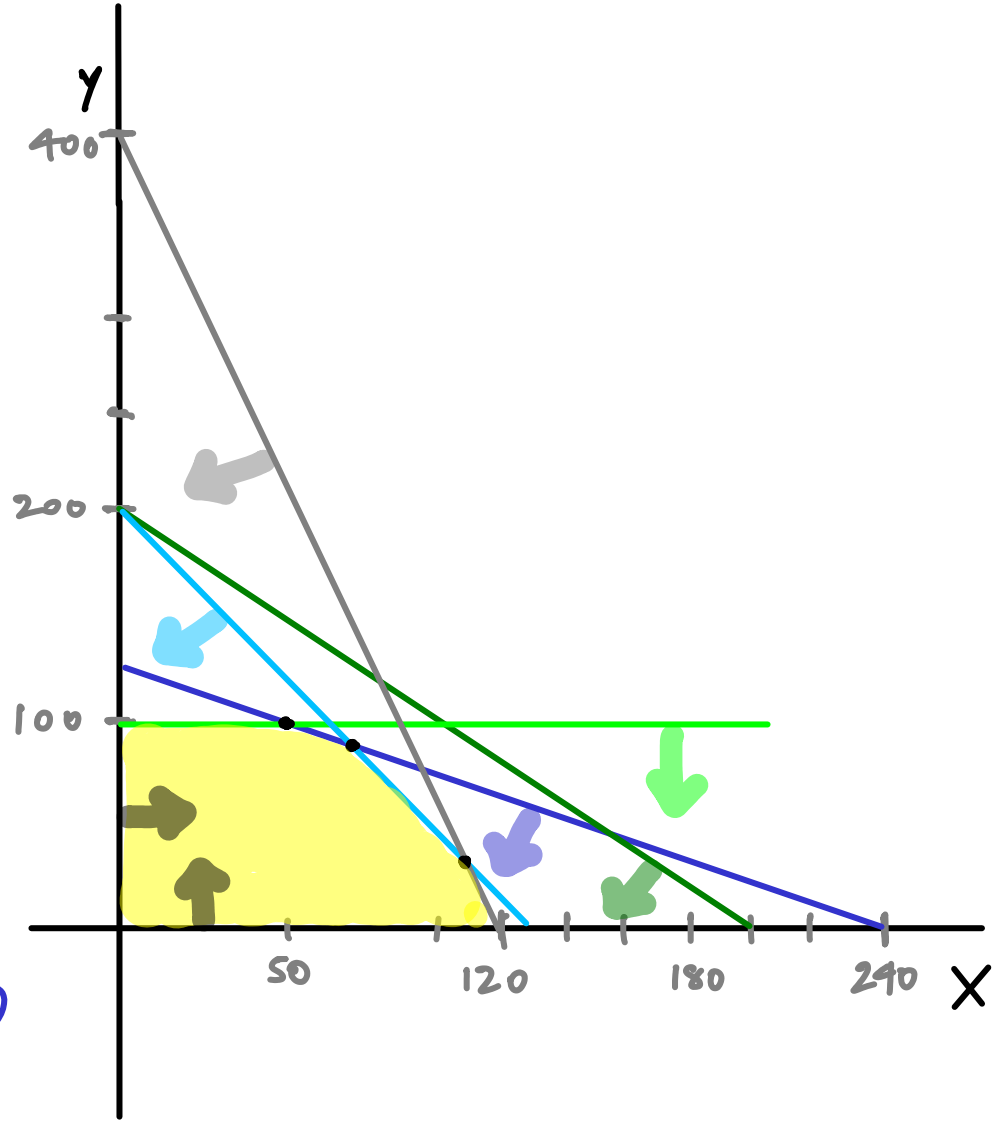
available

subject to :

TIME	60	$\geq$	$0.25x$	$+$	$0.5y$
LEMONS	200	$\geq$	$1x$	$+$	$1y$
SUGAR	250	$\geq$	$2x$	$+$	$1.25y$
WATER	240	$\geq$	$2x$	$+$	$0.6y$
VODKA	50	$\geq$	0		$0.5y$

$x, y \geq 0$

LINEAR PROGRAMMING



	x	y
	REGULAR	SPECIAL
PROFIT	1	2

maximize $x + 2y$

SUPPLIES → used

↓

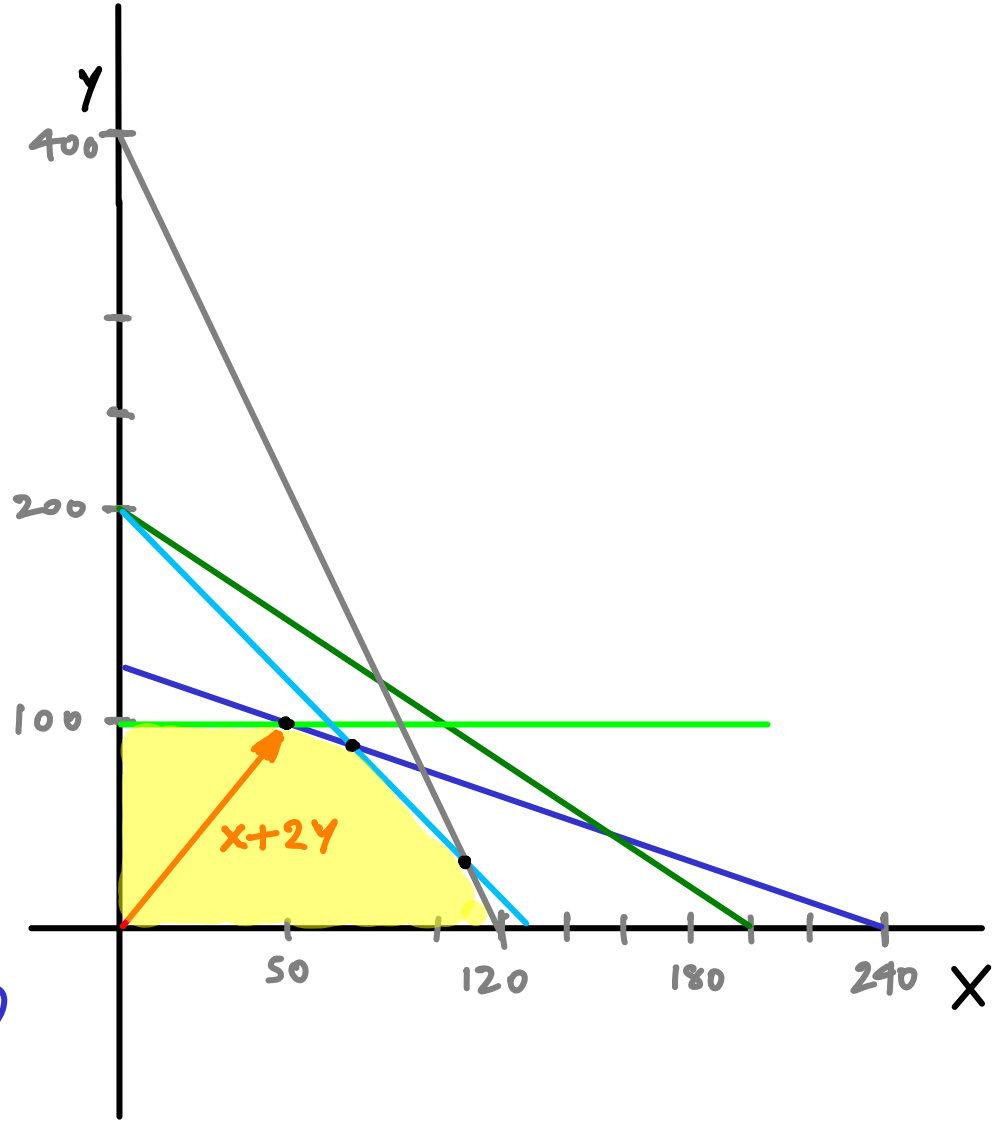
available

subject to :

TIME	60	$\geq$	$0.25x$	$+$	$0.5y$
LEMONS	200	$\geq$	$1x$	$+$	$1y$
SUGAR	250	$\geq$	$2x$	$+$	$1.25y$
WATER	240	$\geq$	$2x$	$+$	$0.6y$
VODKA	50	$\geq$	0		$0.5y$

$x, y \geq 0$

LINEAR PROGRAMMING



	x REGULAR	y SPECIAL
PROFIT	1	2

maximize $x + 2y$

SUPPLIES → used

↓

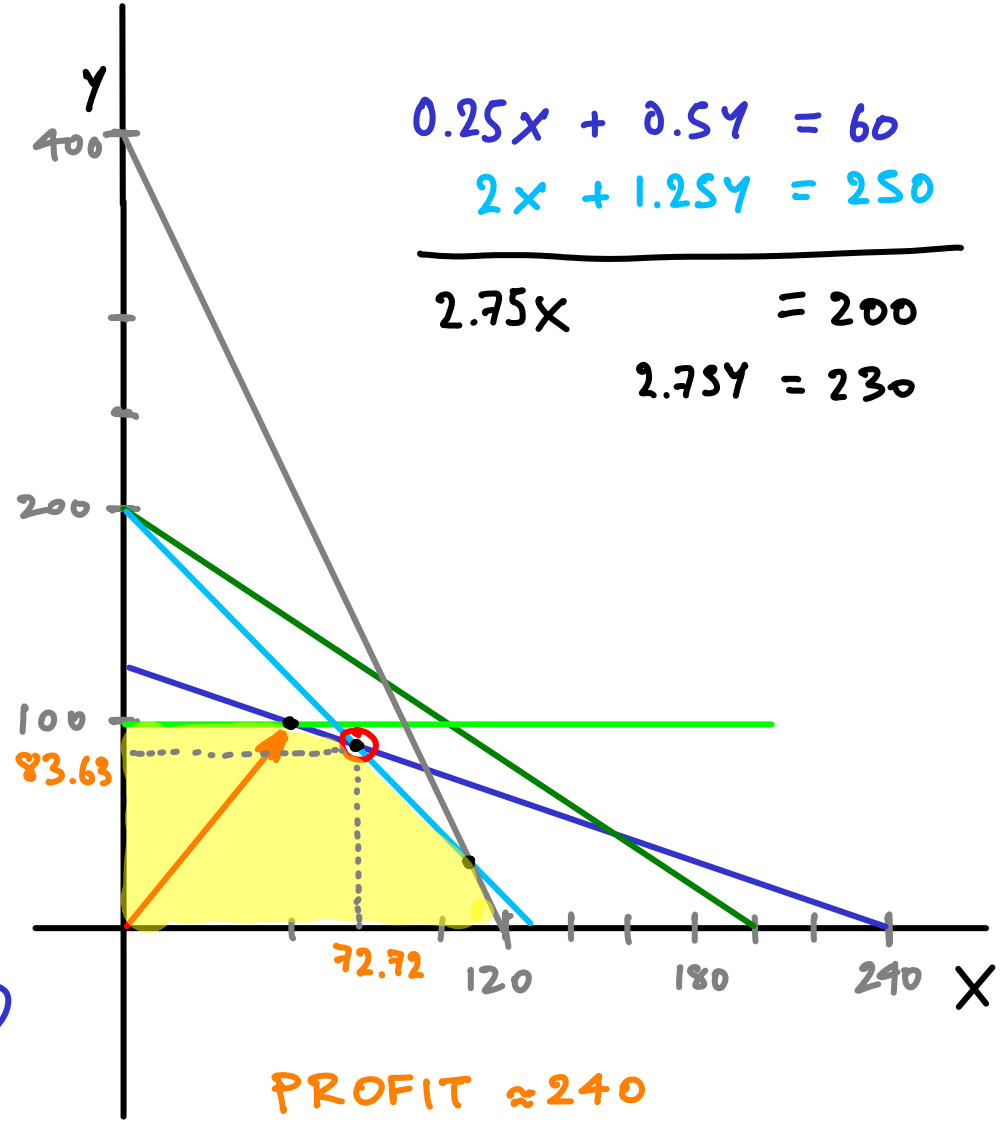
available

subject to :

* TIME	60	≥	$0.25x$	+	$0.5y$
LEMONS	200	≥	$1x$	+	$1y$
* SUGAR	250	≥	$2x$	+	$1.25y$
WATER	240	≥	$2x$	+	$0.6y$
VODKA	50	≥	0		$0.5y$

$x, y \geq 0$

LINEAR PROGRAMMING



$0.25x + 0.5y = 60$

$2x + 1.25y = 250$

$2.75x = 200$

$2.75y = 230$

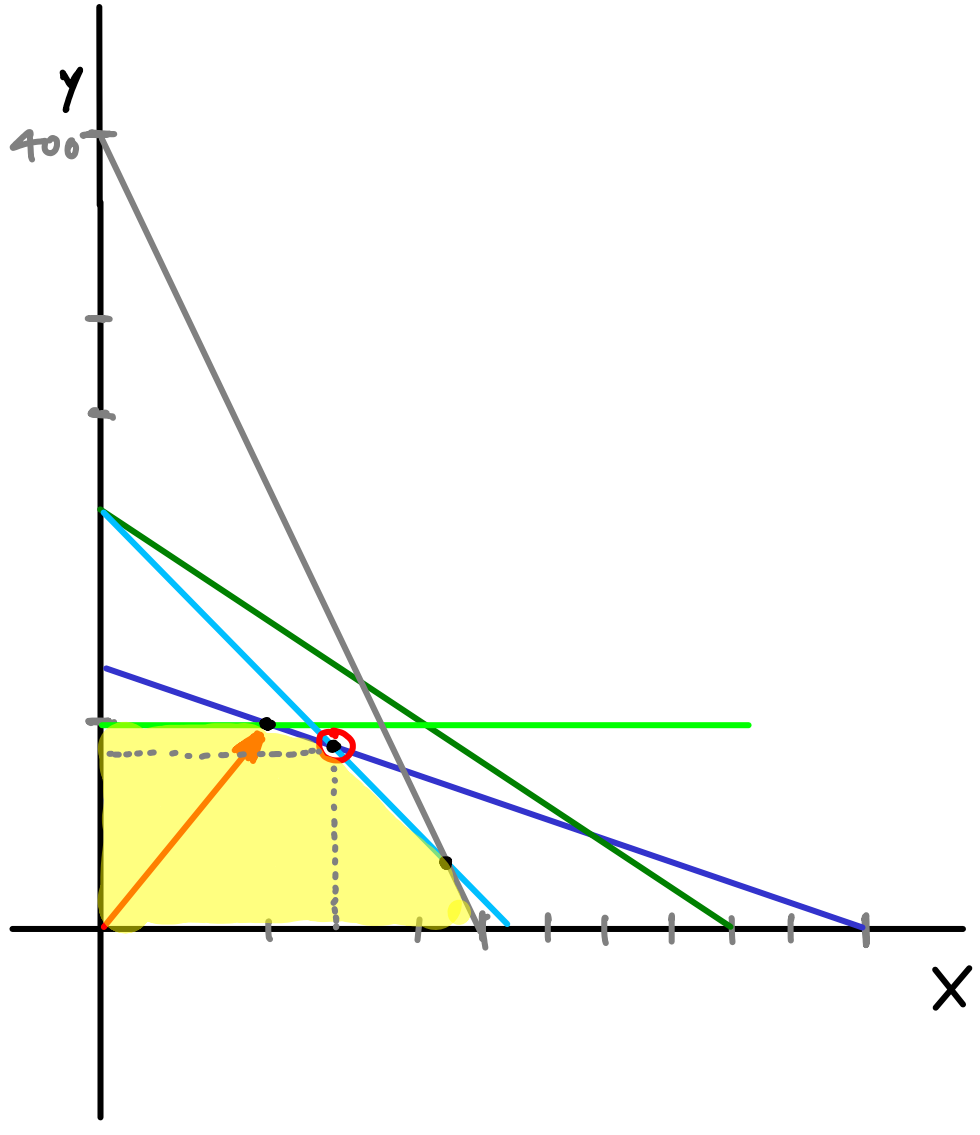
PROFIT ≈ 240

Notes

Finding an integer solution
is more difficult in general

↓

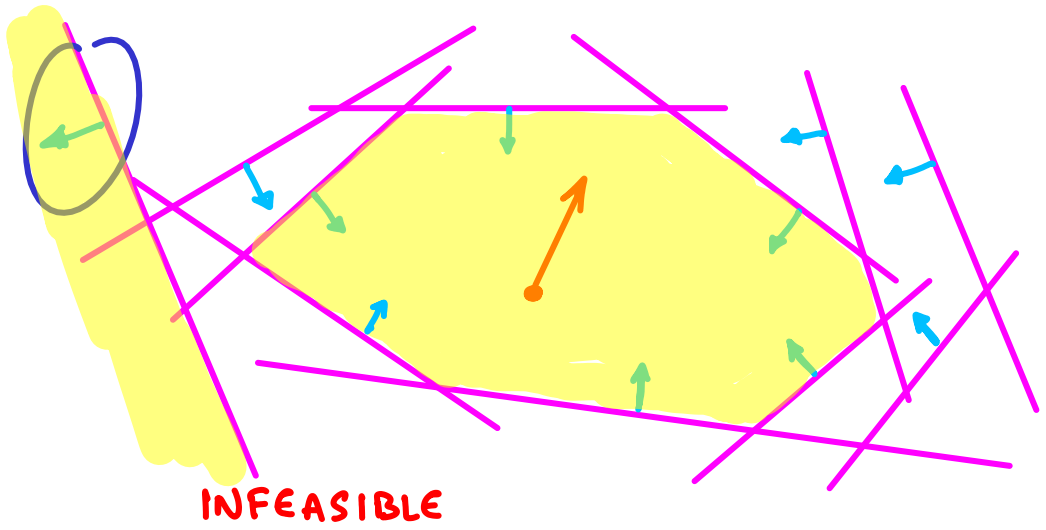
Integer programming

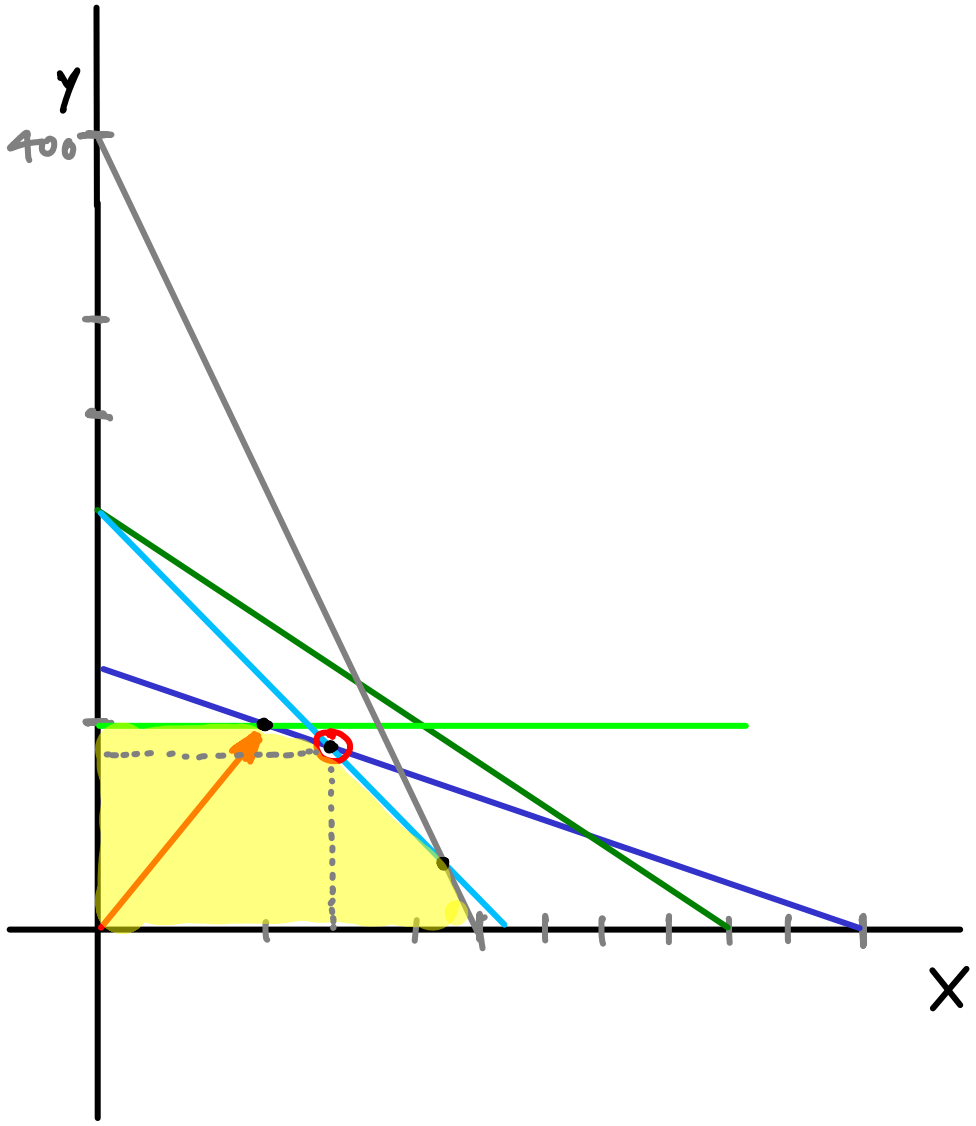


Notes

Finding an integer solution
is more difficult in general

Finding if any valid solution exists
is not really an easier problem





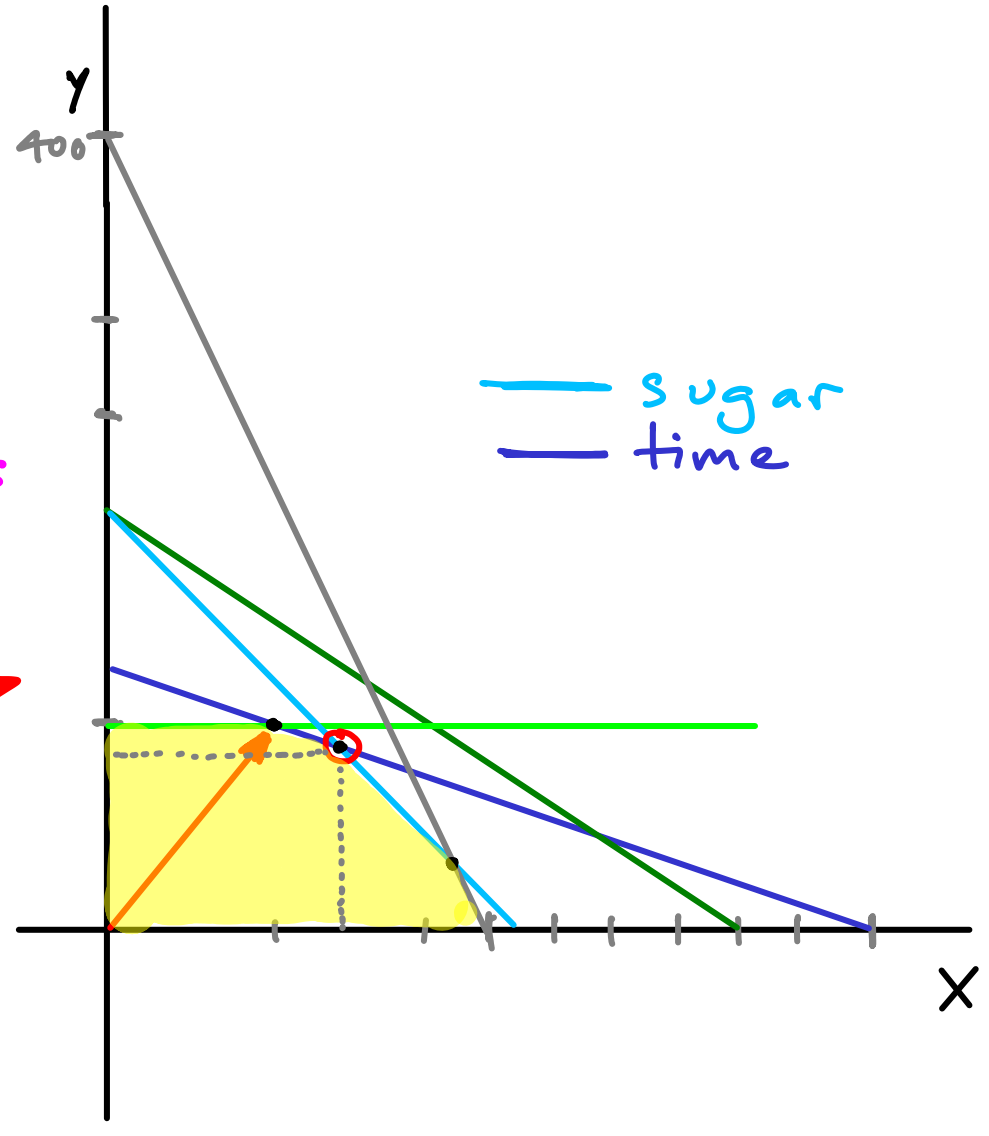
Notes

Finding an integer solution
is more difficult in general

Finding if any valid solution exists
is not really an easier problem

The solution tells us that we didn't have enough time or sugar

↳ might be worth it to hire help or make drinks less sweet



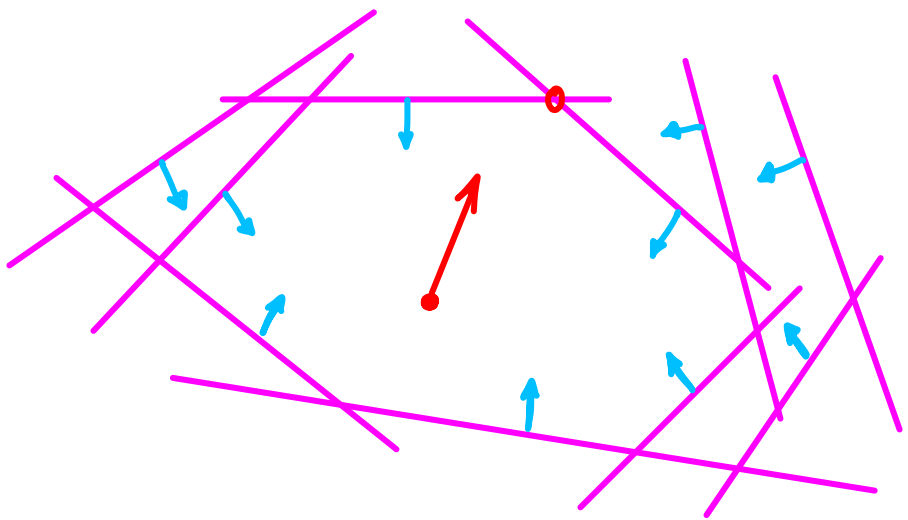
2D LP : maximize $aX + bY$

subject to $r_1 X + s_1 Y \leq k_1$

$r_2 X + s_2 Y \leq k_2$

$\vdots$

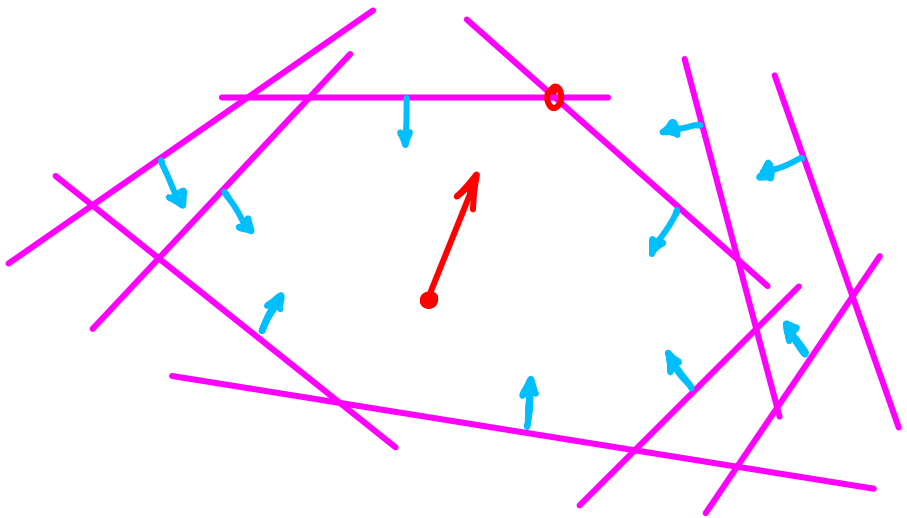
$r_n X + s_n Y \leq k_n$



LP : maximize $c_1X_1 + c_2X_2 + \cdots + c_dX_d$

subject to

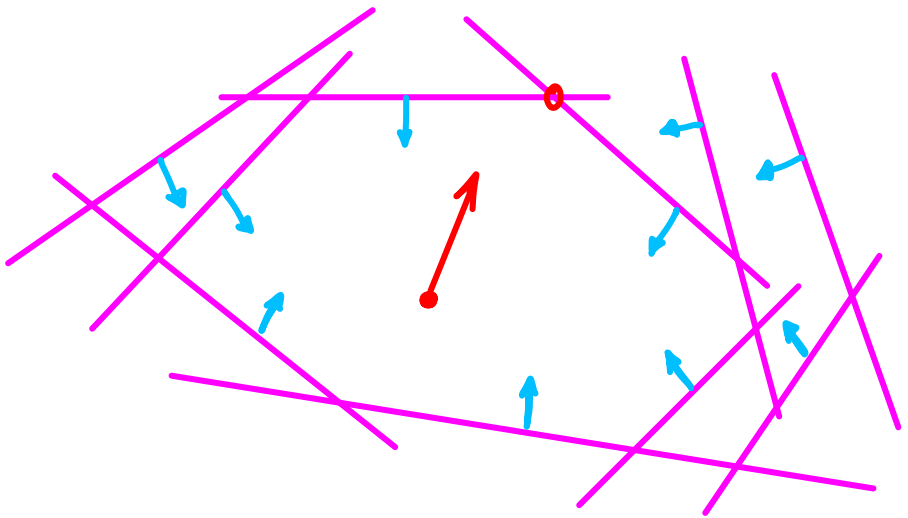
$$a_{11}X_1 + a_{21}X_2 + \cdots + a_{d1}X_d \leq b_1$$
$$a_{12}X_1 + a_{22}X_2 + \cdots + a_{d2}X_d \leq b_2$$
$$\vdots$$
$$a_{1n}X_1 + a_{2n}X_2 + \cdots + a_{dn}X_d \leq b_n$$



LP : maximize $c_1X_1 + c_2X_2 + \cdots + c_dX_d$

subject to

$$a_{11}X_1 + a_{21}X_2 + \cdots + a_{d1}X_d \leq b_1$$
$$a_{12}X_1 + a_{22}X_2 + \cdots + a_{d2}X_d \leq b_2$$
$$\vdots$$
$$a_{1n}X_1 + a_{2n}X_2 + \cdots + a_{dn}X_d \leq b_n$$



matrix form:

maximize $c^T X$

$$[c_1 \ldots c_d] \cdot \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$$

subject to $A\vec{x} \leq \vec{b}$

$$\begin{bmatrix} a_{11} \cdots a_{d1} \\ \vdots \\ a_{1n} \cdots a_{dn} \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} \leq \begin{bmatrix} k_1 \\ \vdots \\ k_n \end{bmatrix}$$

Classic methods to solve LP

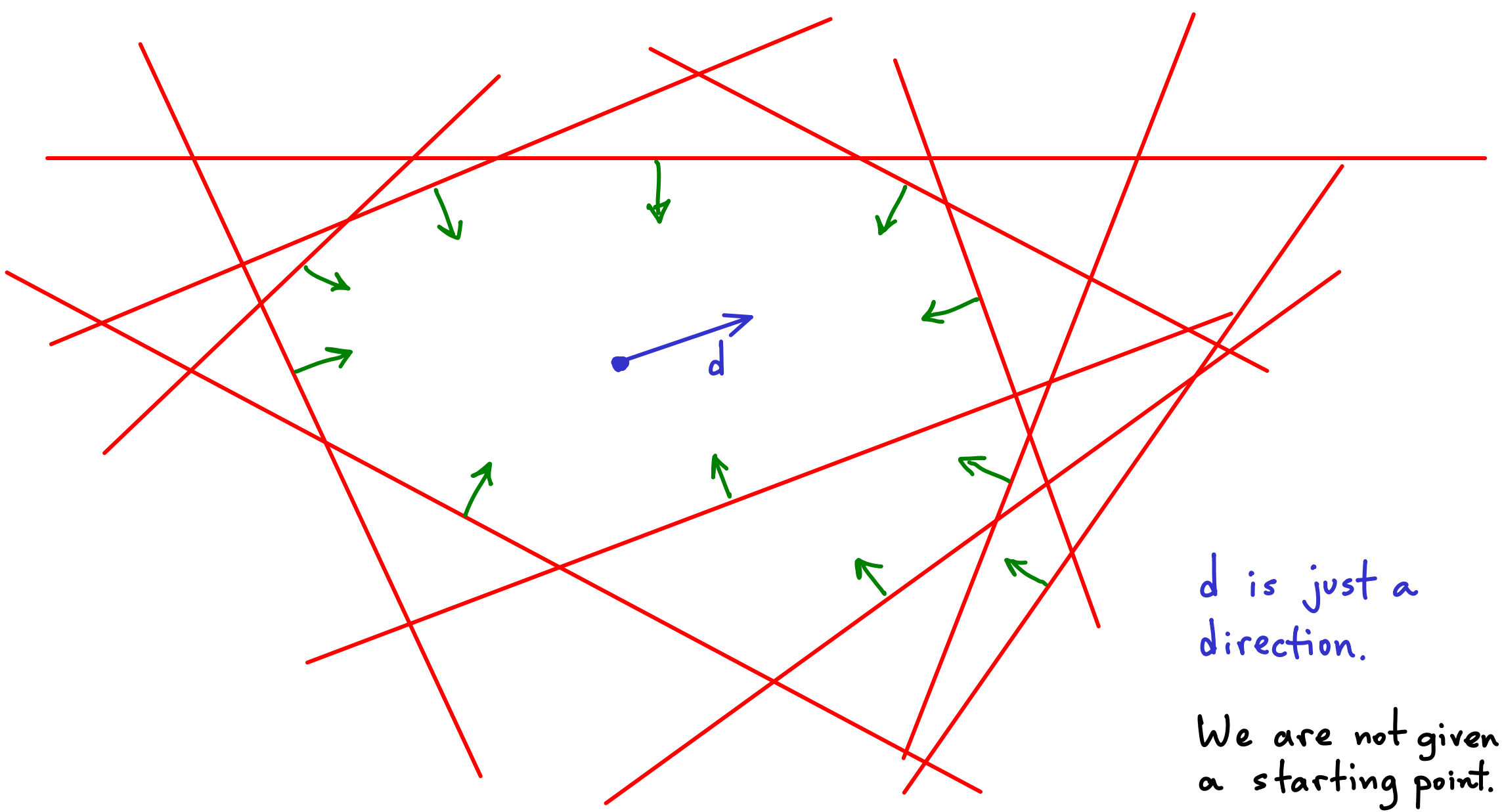
- simplex algorithm
- interior point methods

fast in practice, bad worst case

Coming up: geometric method for 2D

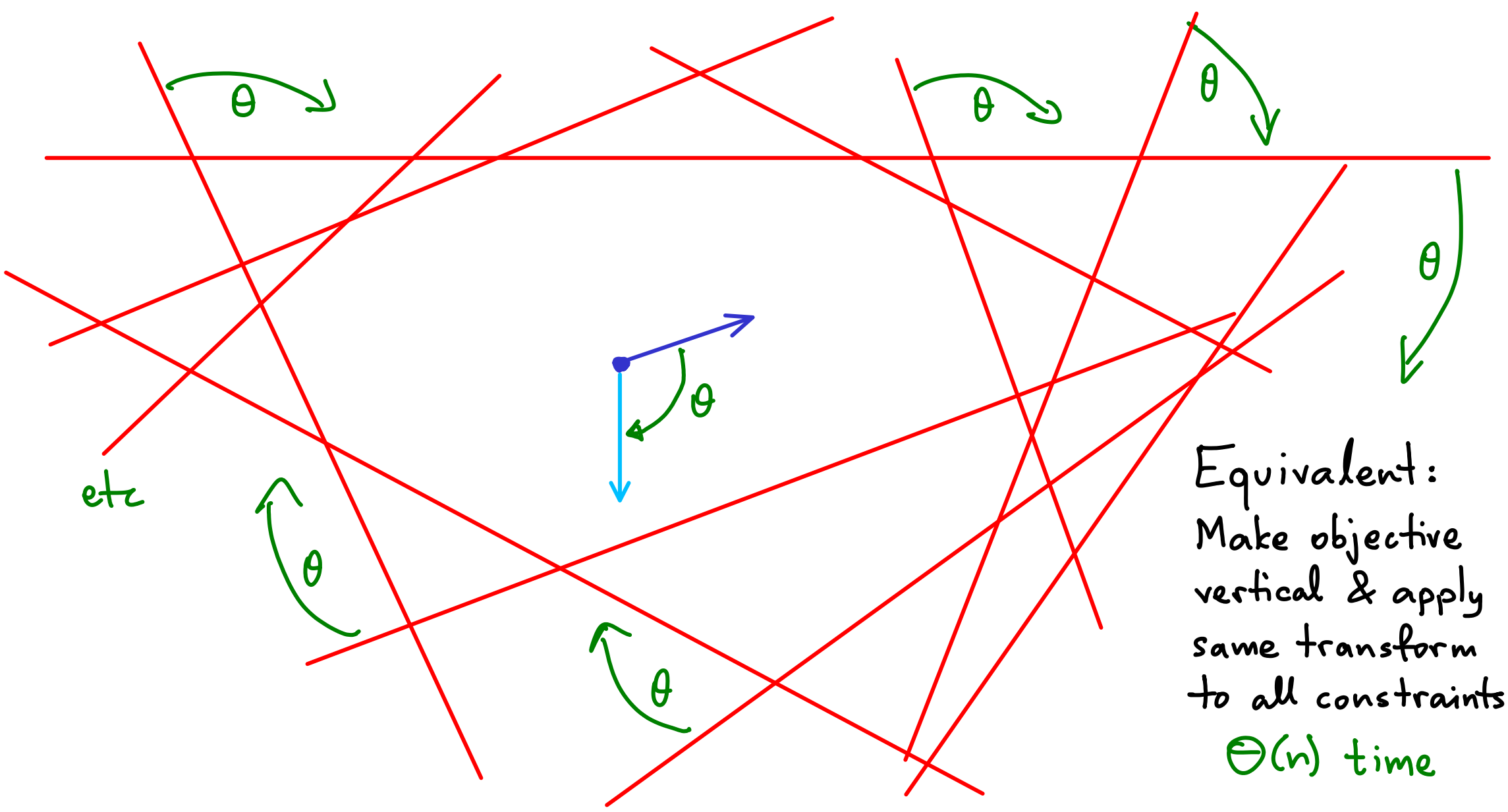
MAXIMIZING a LINEAR OBJECTIVE FUNCTION SUBJECT TO LINEAR CONSTRAINTS (INEQUALITIES)

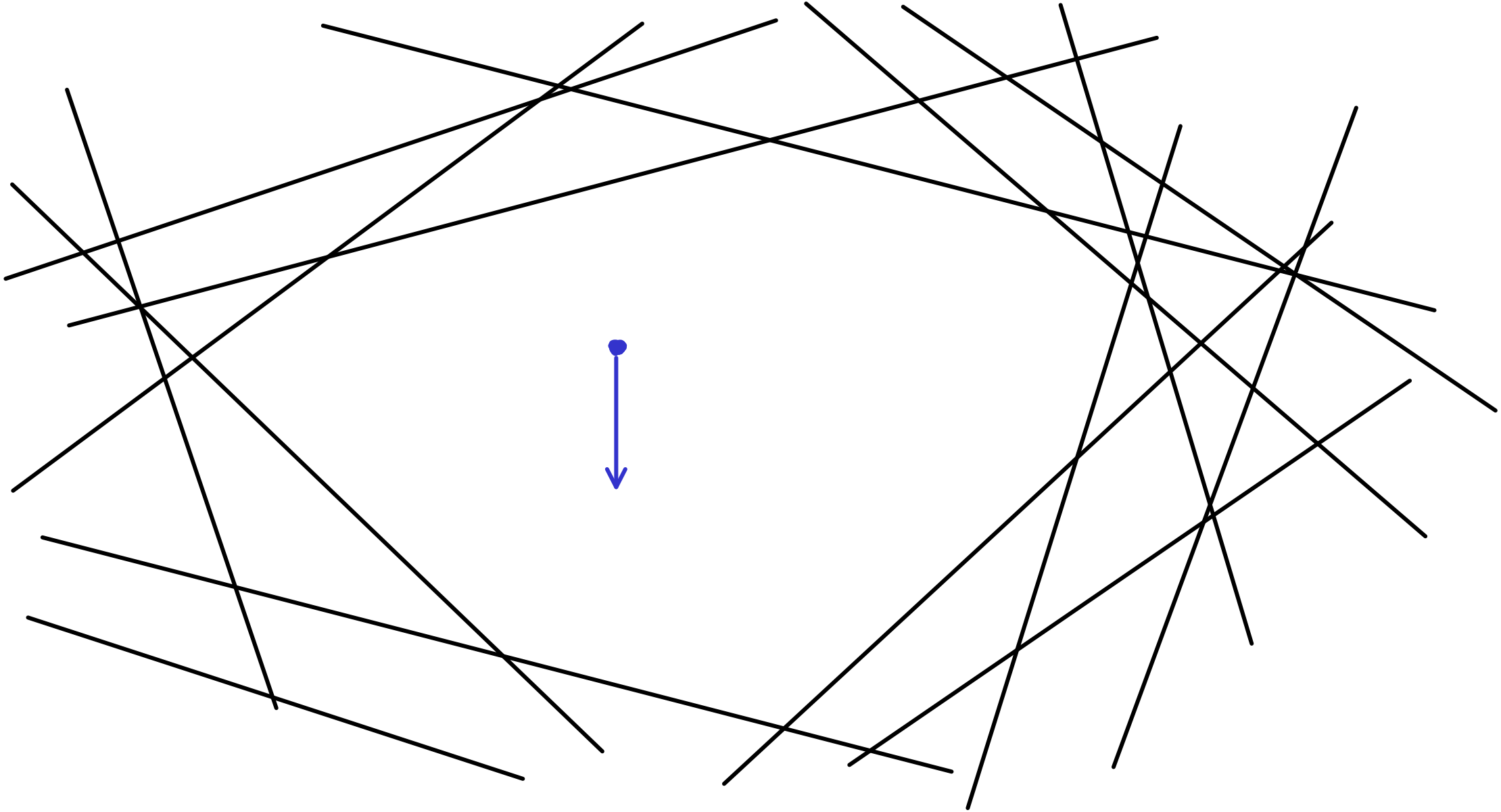
Megiddo's algorithm

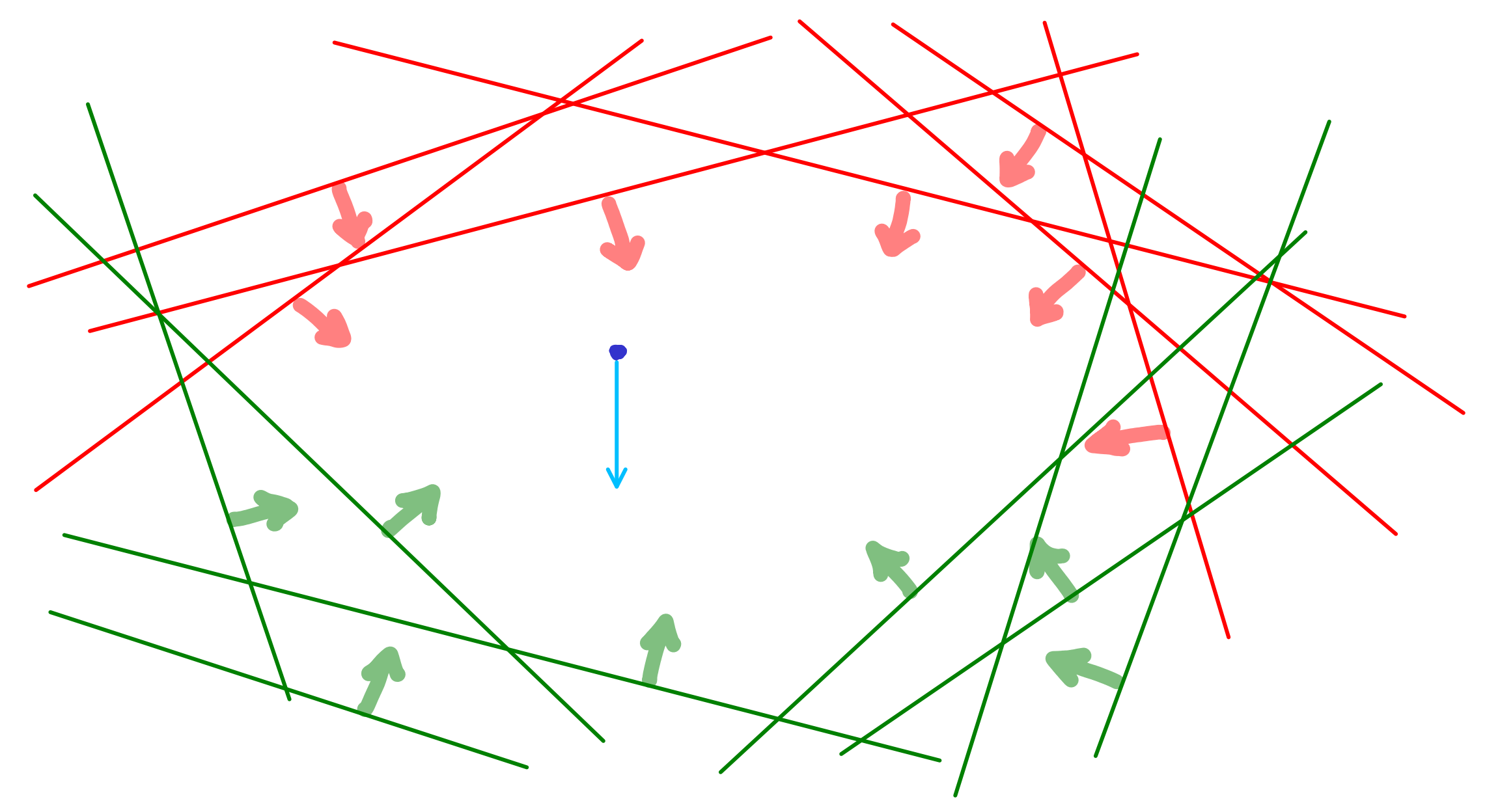


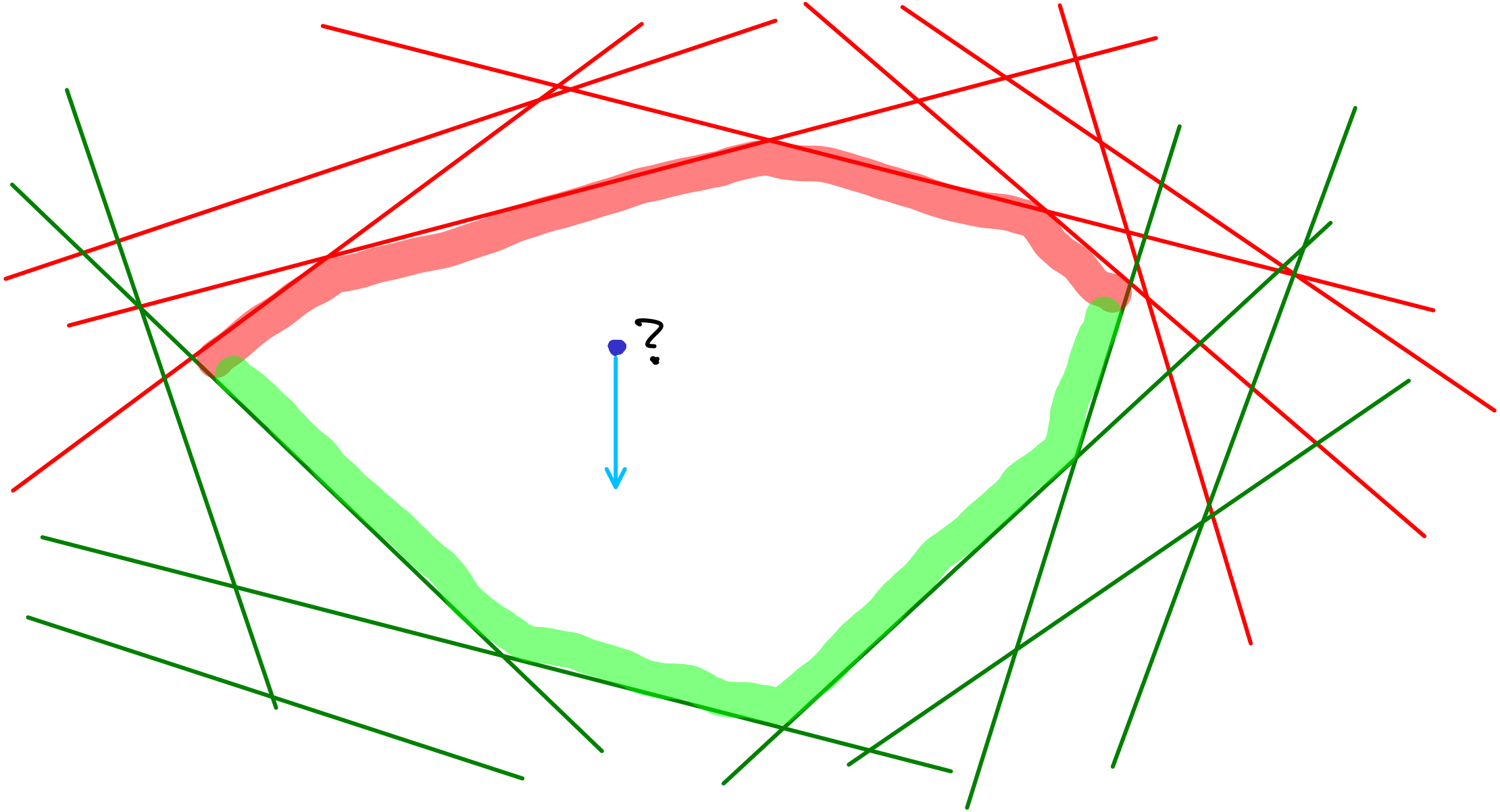
d is just a direction.

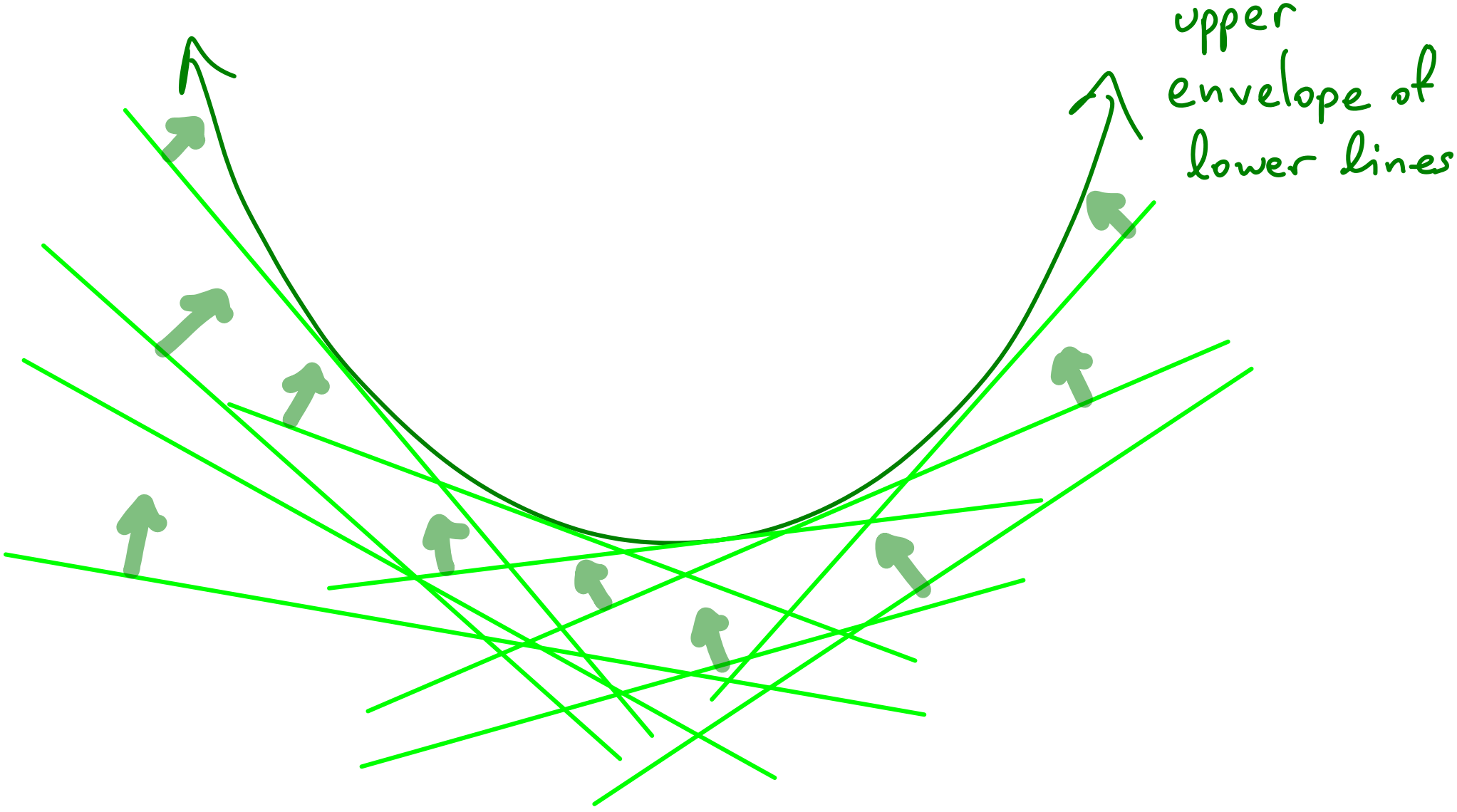
We are not given a starting point.

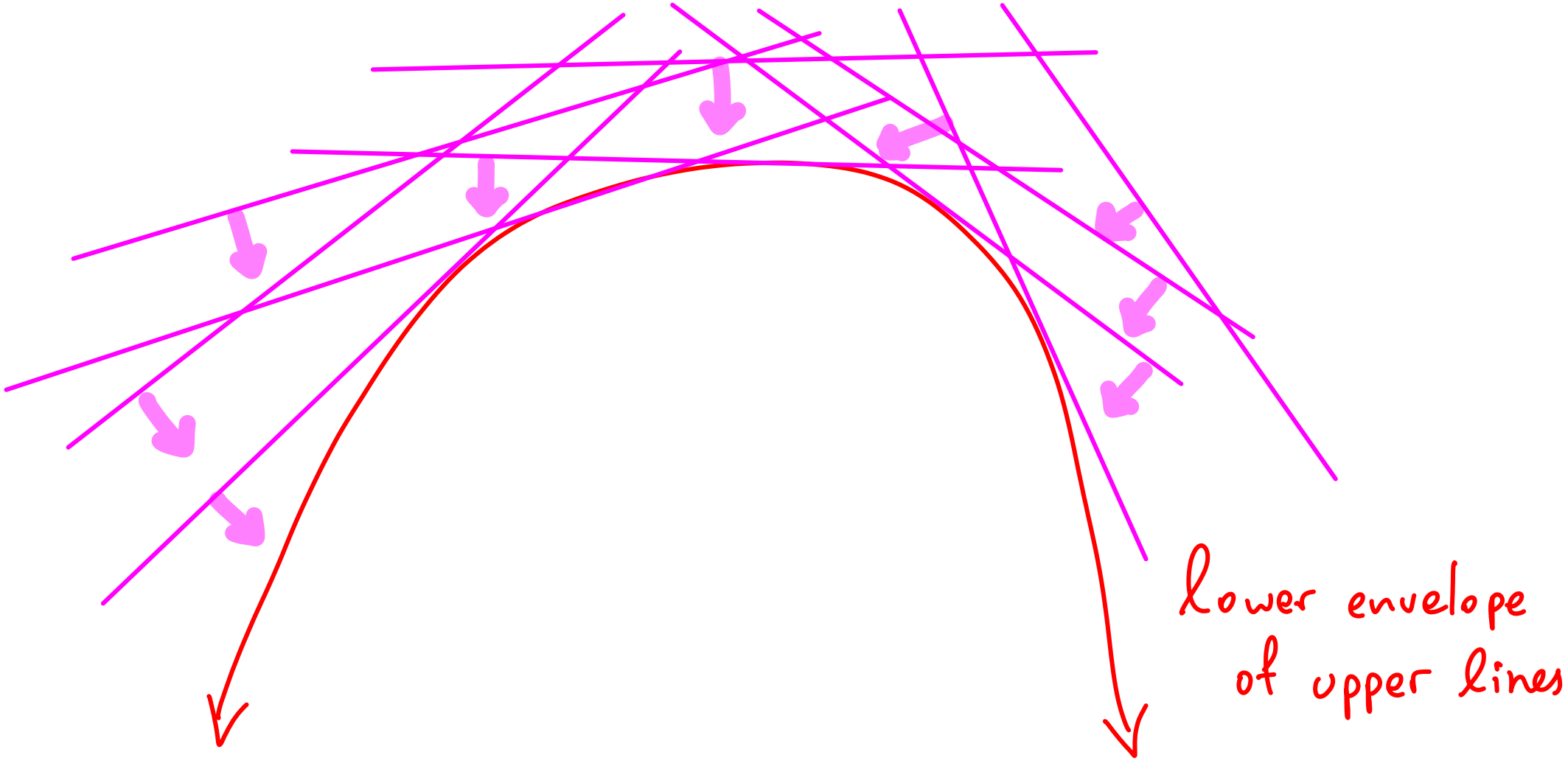

θ
θ
θ
θ
θ
θ
θ
etc
Equivalent:
Make objective
vertical & apply
same transform
to all constraints
Θ(n) time

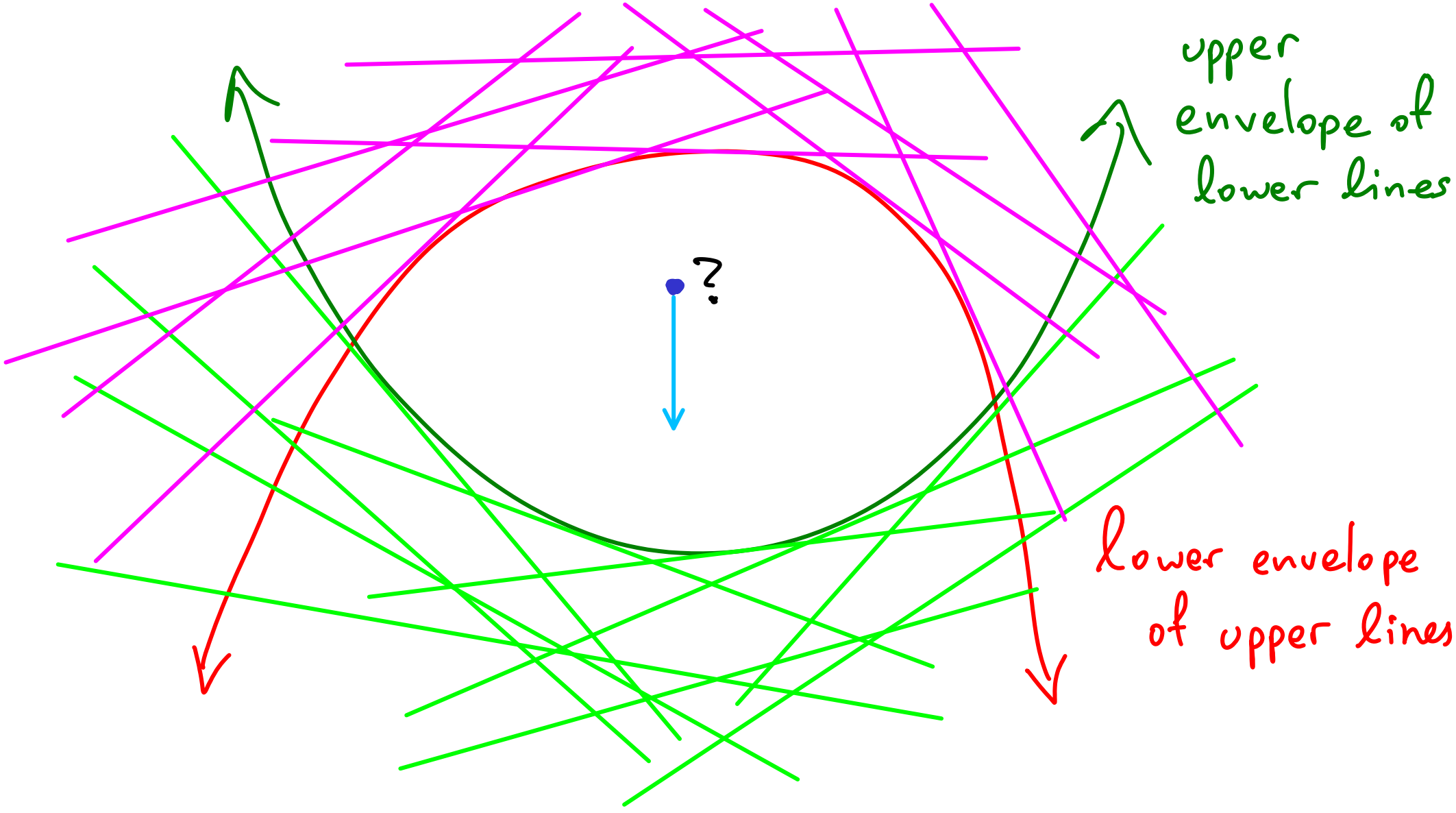




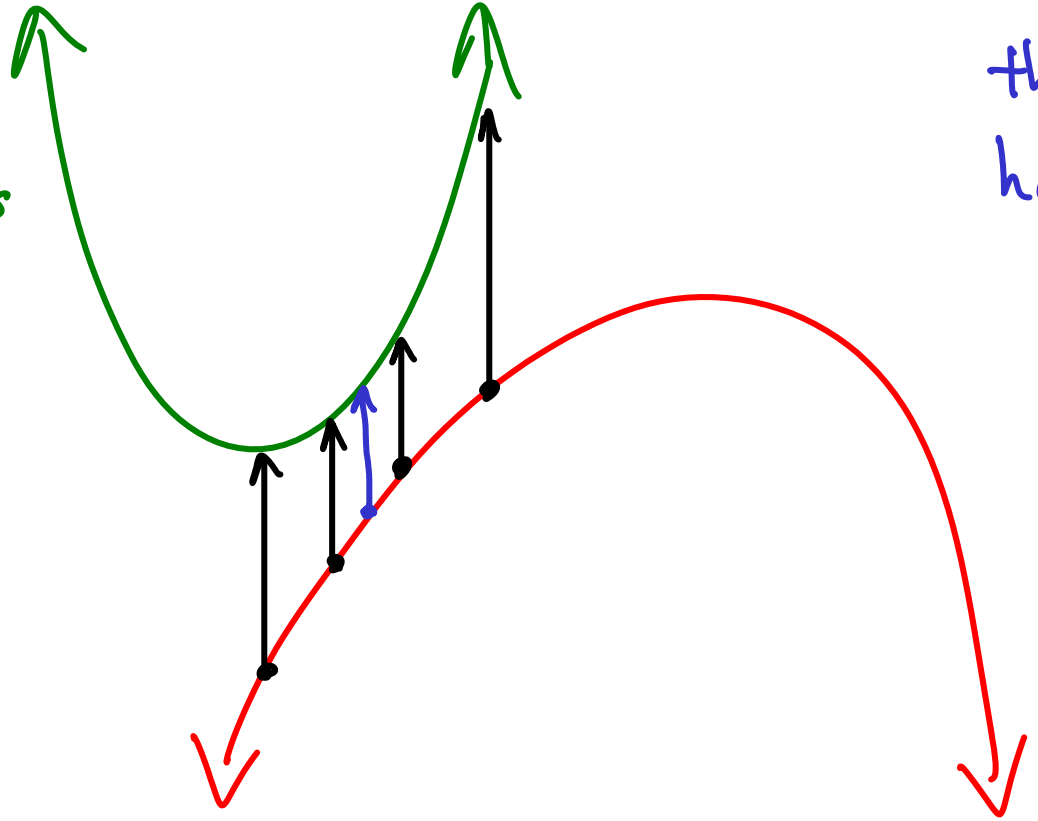



upper
envelope of
lower lines


lower envelope
of upper lines


upper
envelope of
lower lines
?
lower envelope
of upper lines

upper envelope of lower lines



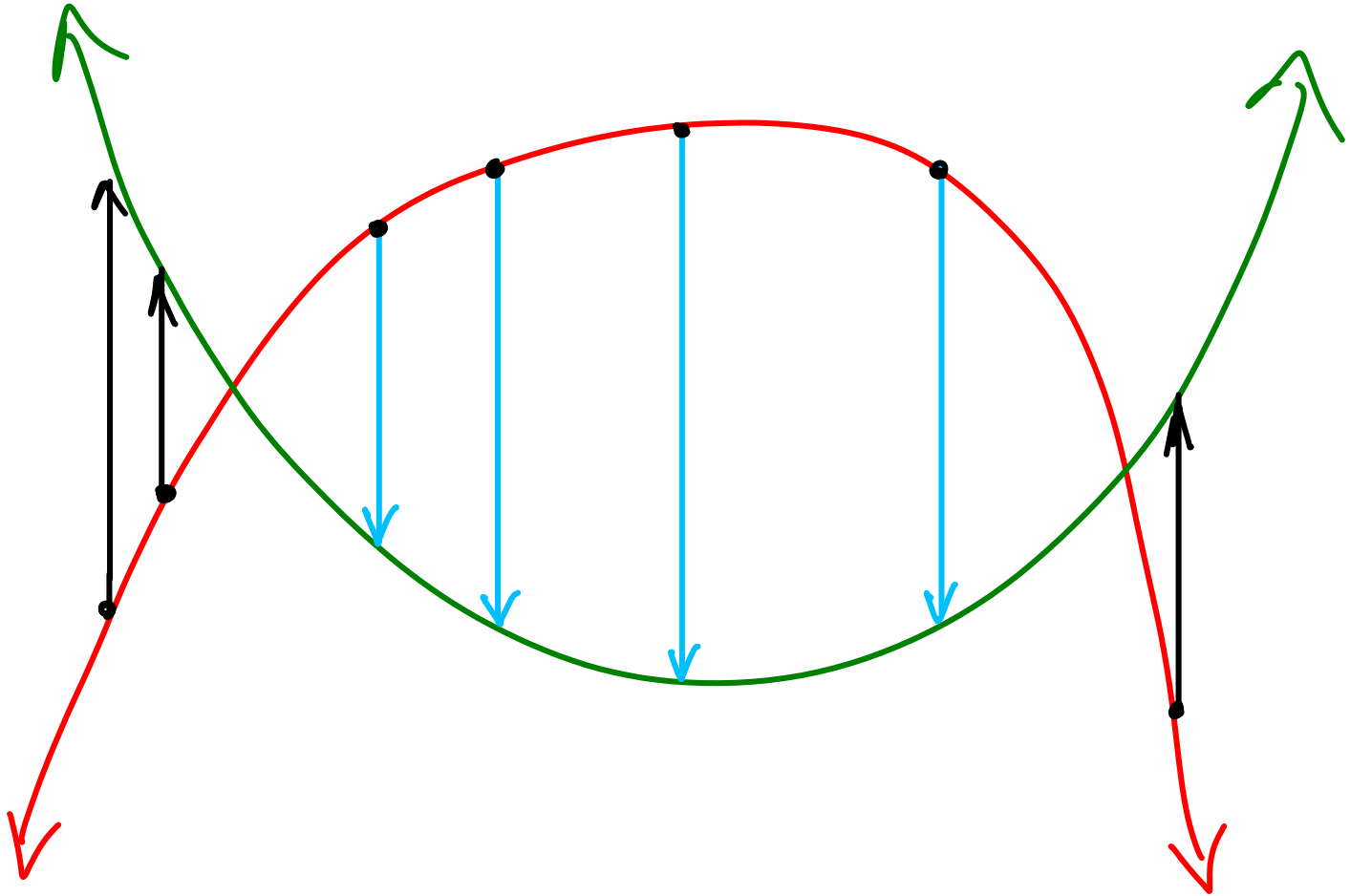
if no intersection then envelopes have some vertical separation that has a local min.

lower envelope of upper lines

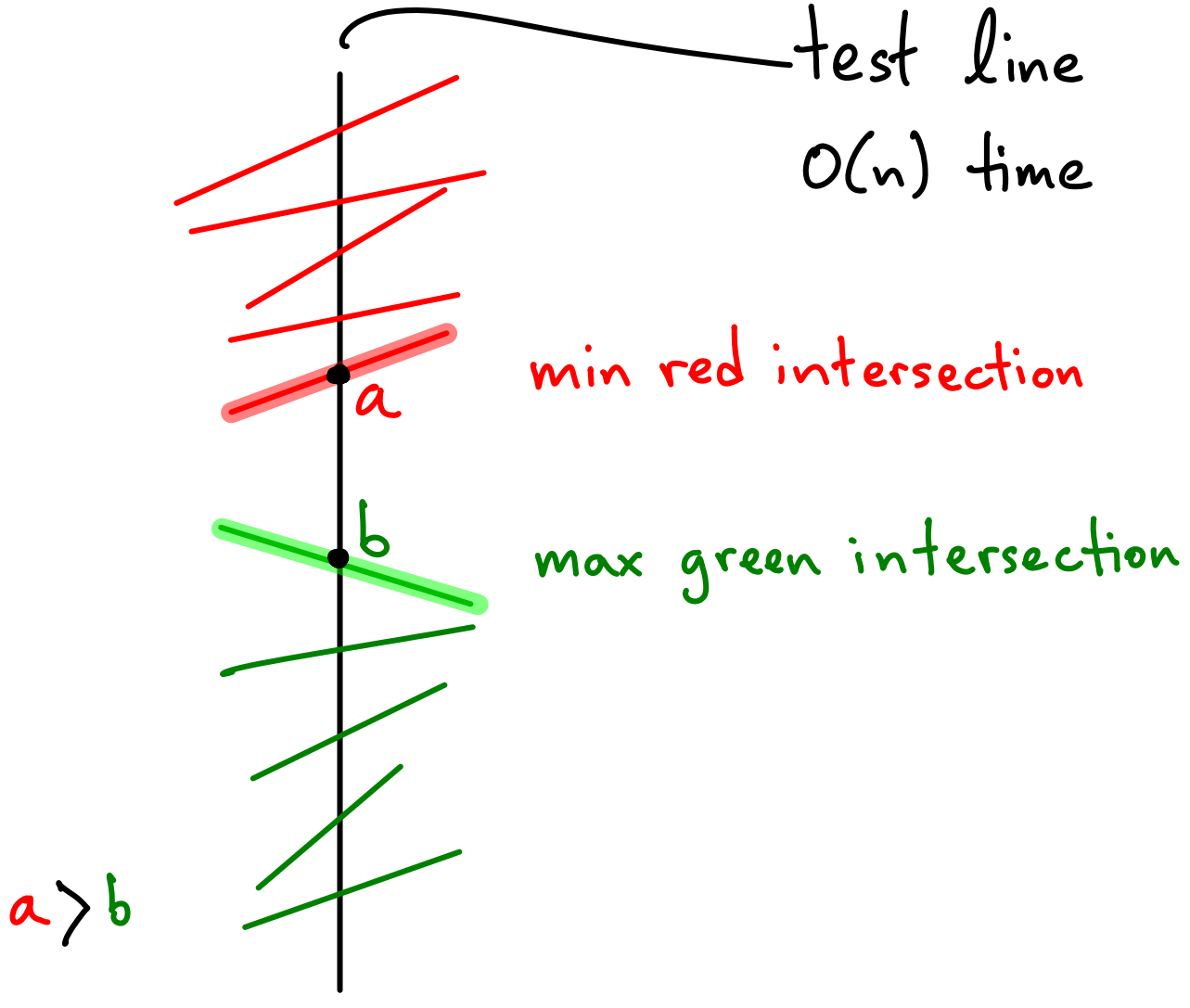
For every x-coord, green is above red

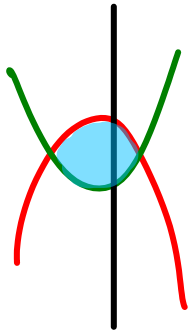
If there is an intersection, then for some x-coord we have

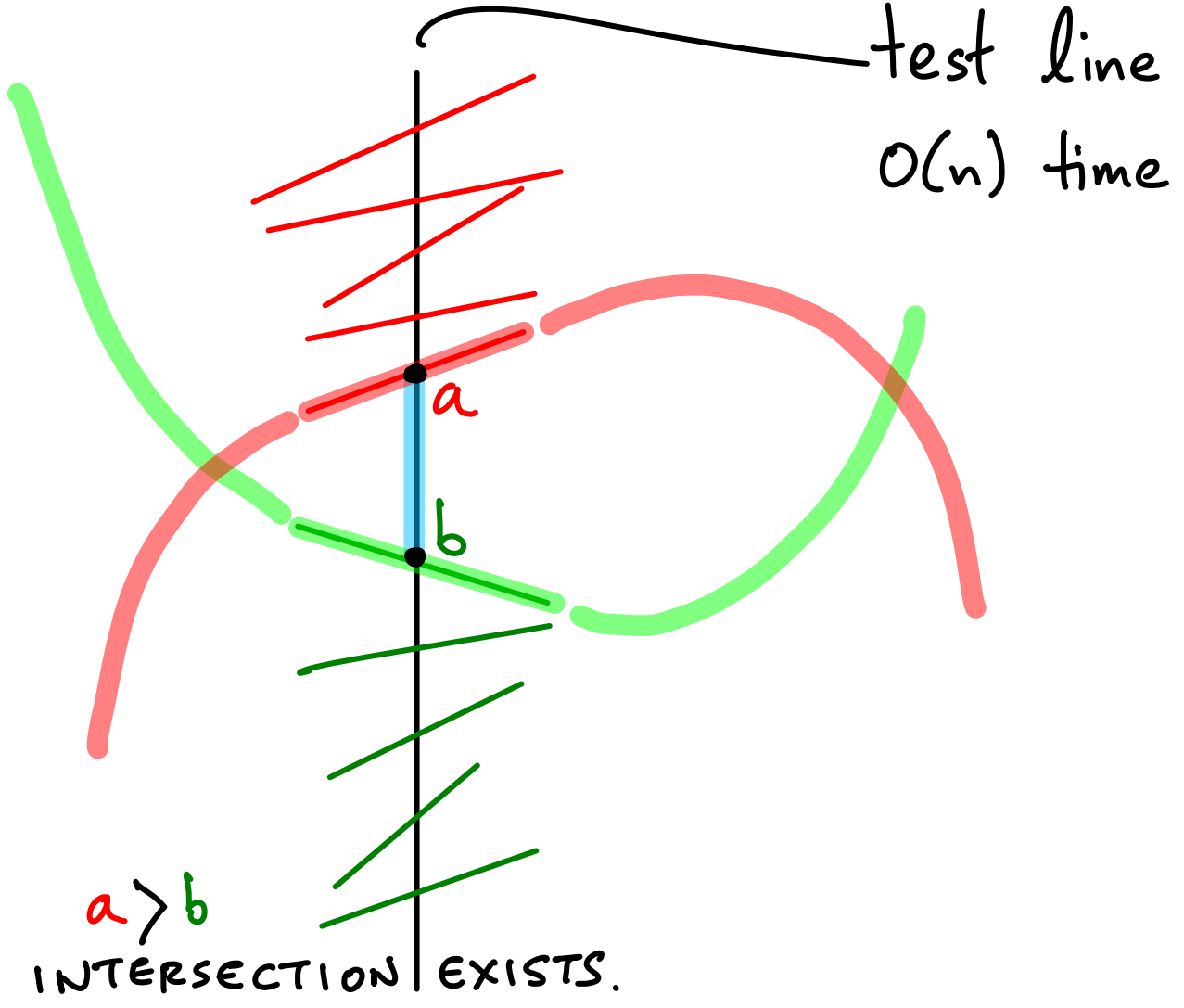
red above green

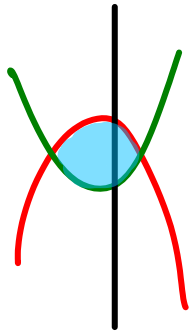


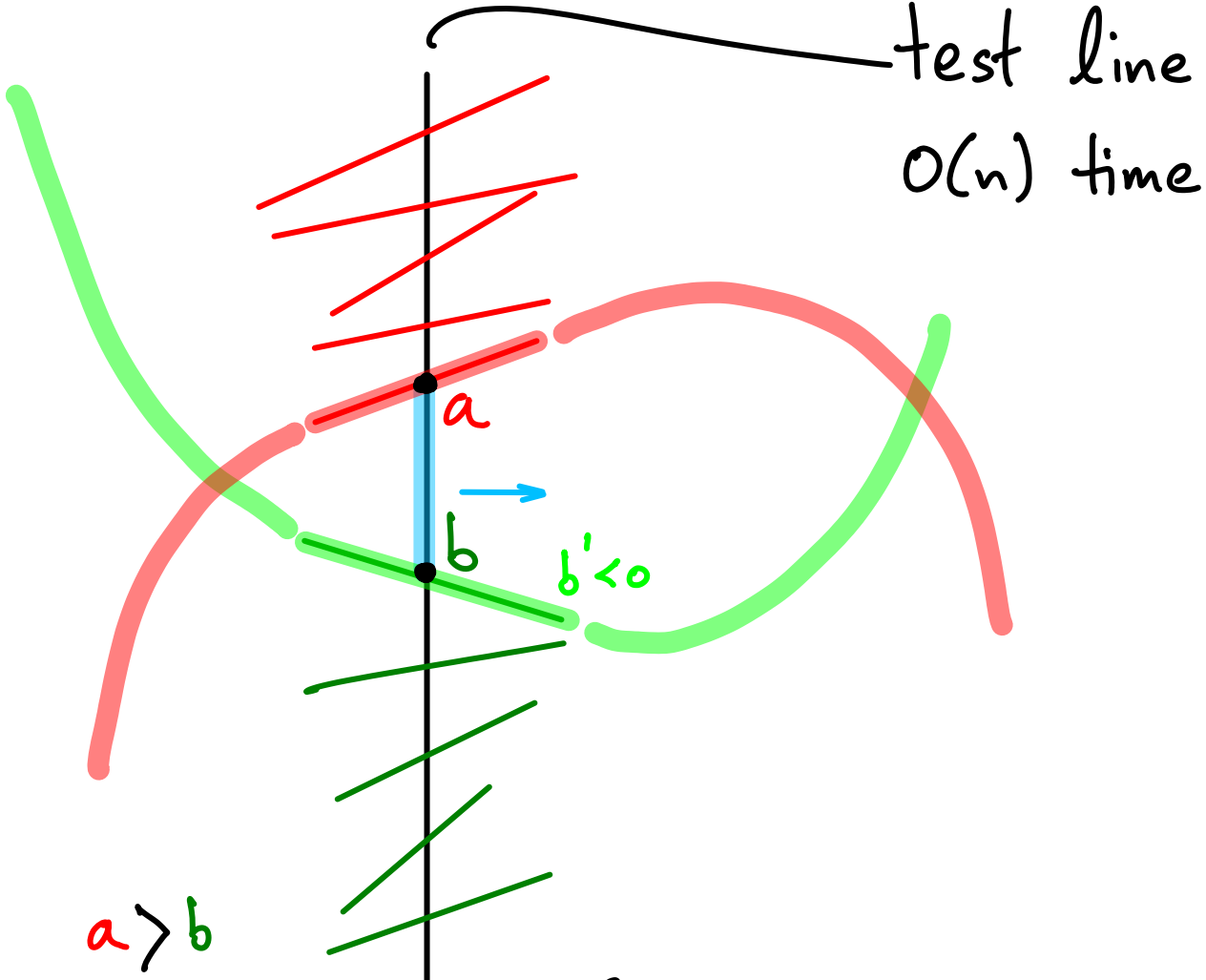
Suppose we test some x-coord:

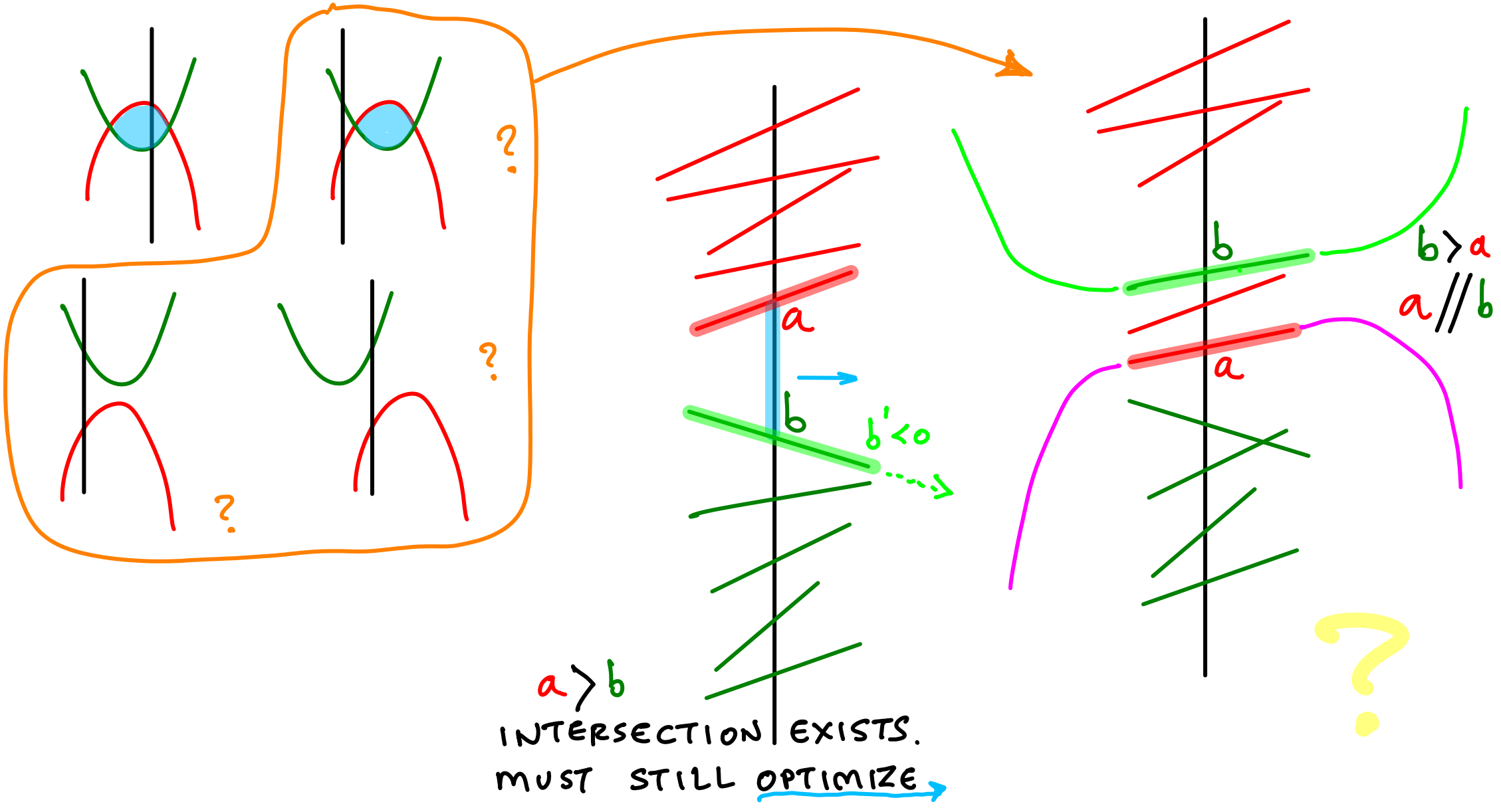


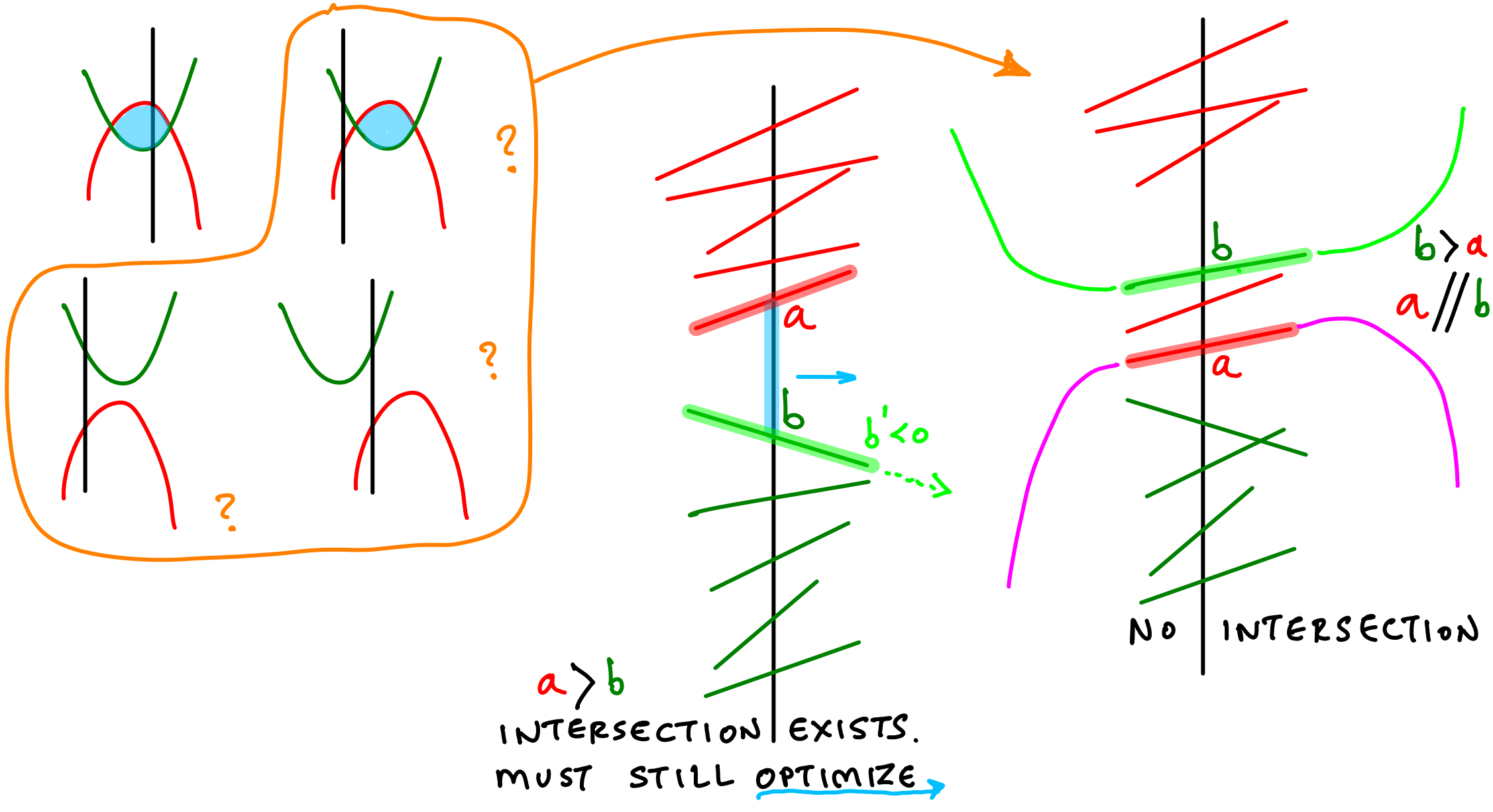


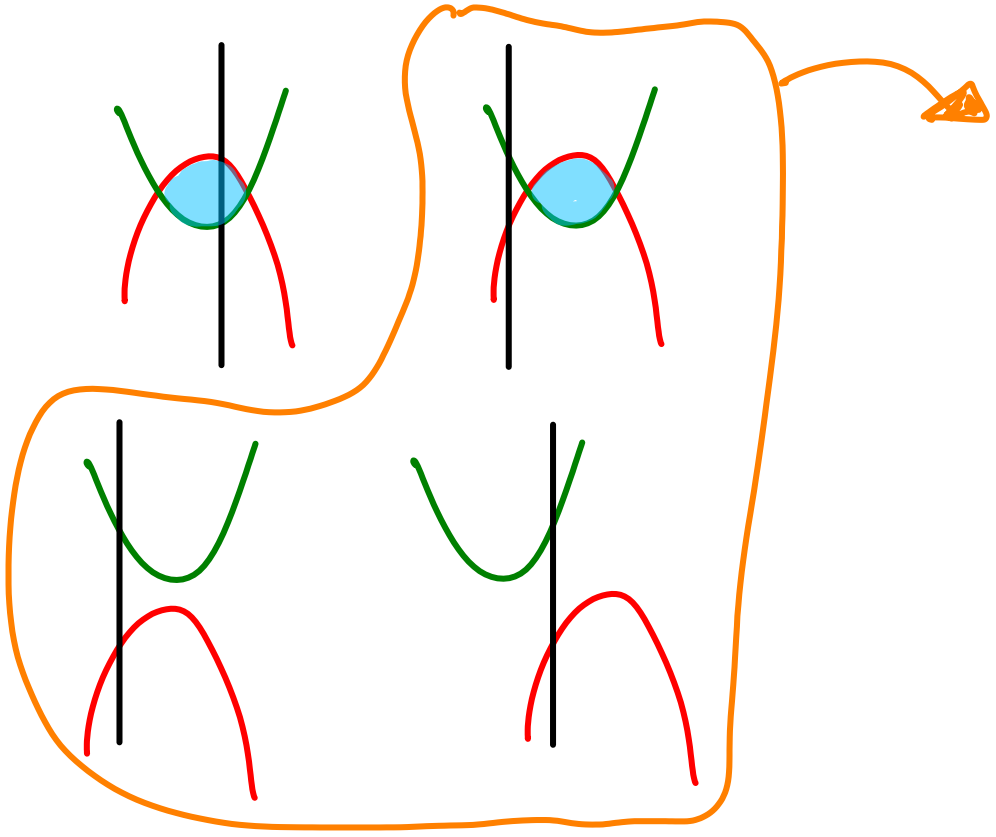

test line
O(n) time
a
b
a > b
INTERSECTION EXISTS.

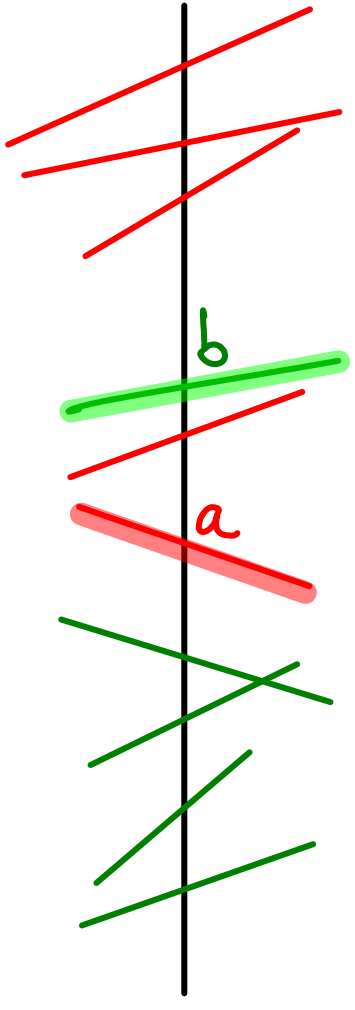



test line
O(n) time
a
b
b'<0
a>b
INTERSECTION EXISTS.
MUST STILL OPTIMIZE → Try larger x-coord


?
?
?
a
b
b'<0
b
a
b>a
a // b
?
a > b
INTERSECTION EXISTS.
MUST STILL OPTIMIZE

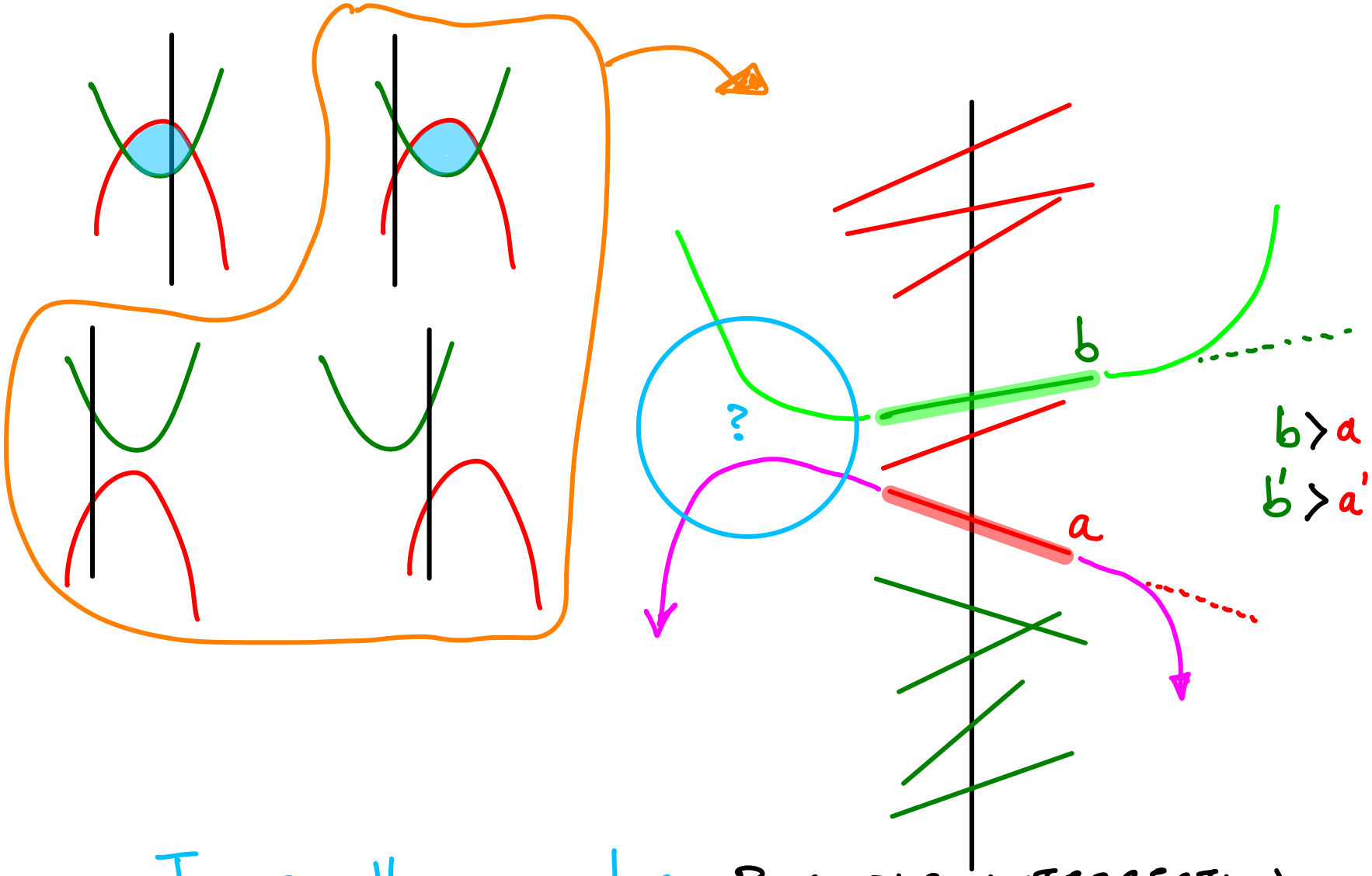

?
?
?
a
b
b'<0
a>b
INTERSECTION EXISTS.
MUST STILL OPTIMIZE
b
b>a
a//b
a
NO INTERSECTION





$b > a$

$b' \neq a'$?


?
b
a
b > a
b' > a'
b' < a' symmetric
Try smaller x-coord ← POSSIBLE INTERSECTION TO THE LEFT

GIVEN A VERTICAL TEST LINE

WE CAN EITHER DECIDE THAT NO INTERSECTION EXISTS

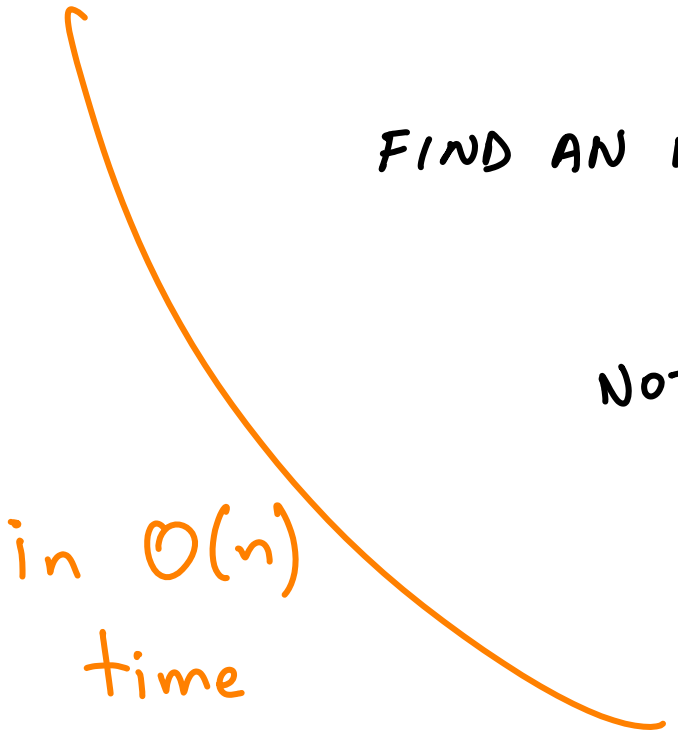
OR

FIND AN INTERSECTION AND WHICH SIDE MIN. IS ON

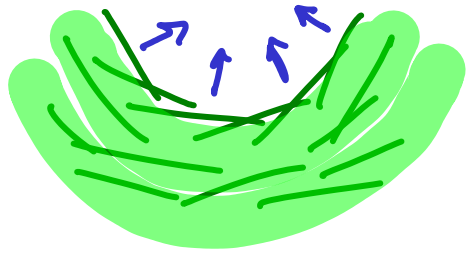
OR

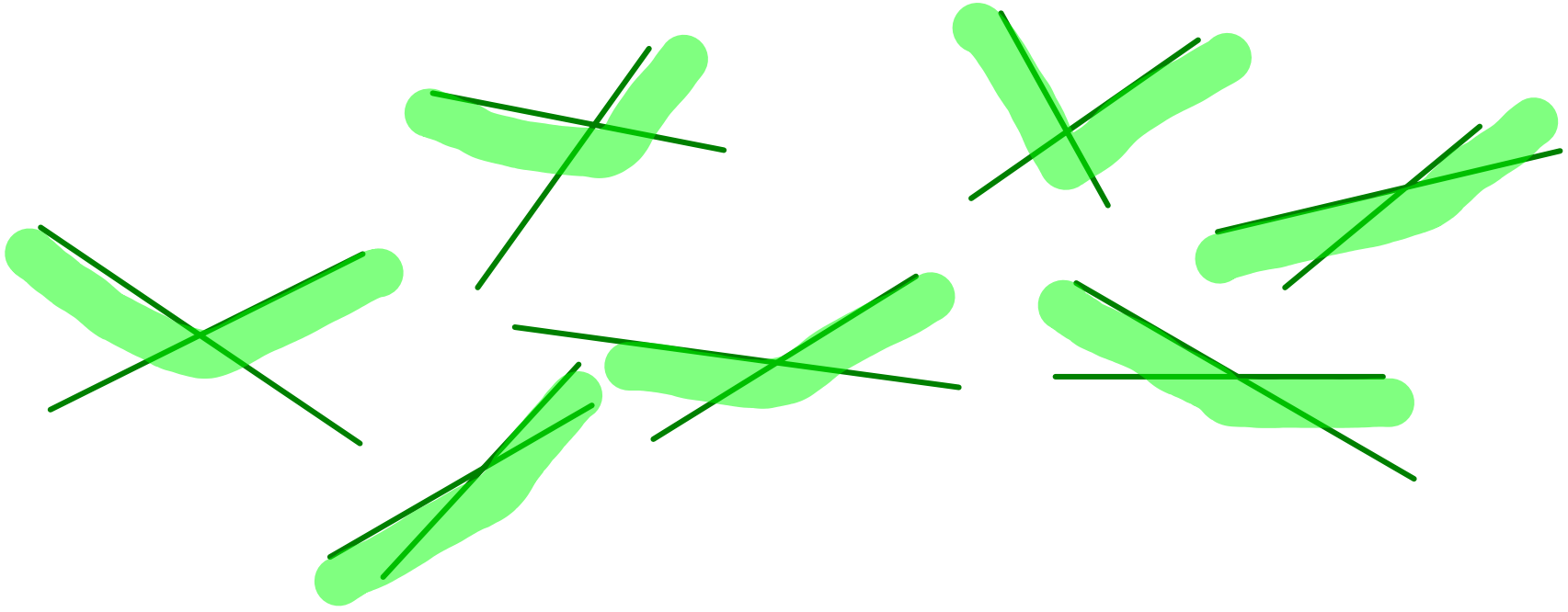
NOT KNOW ANSWER BUT AT LEAST

KNOW WHAT SIDE IT MIGHT BE ON



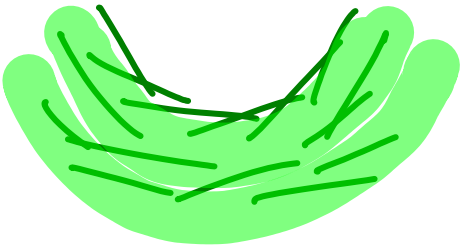
ARBITRARILY PAIR UP THE LOWER LINES

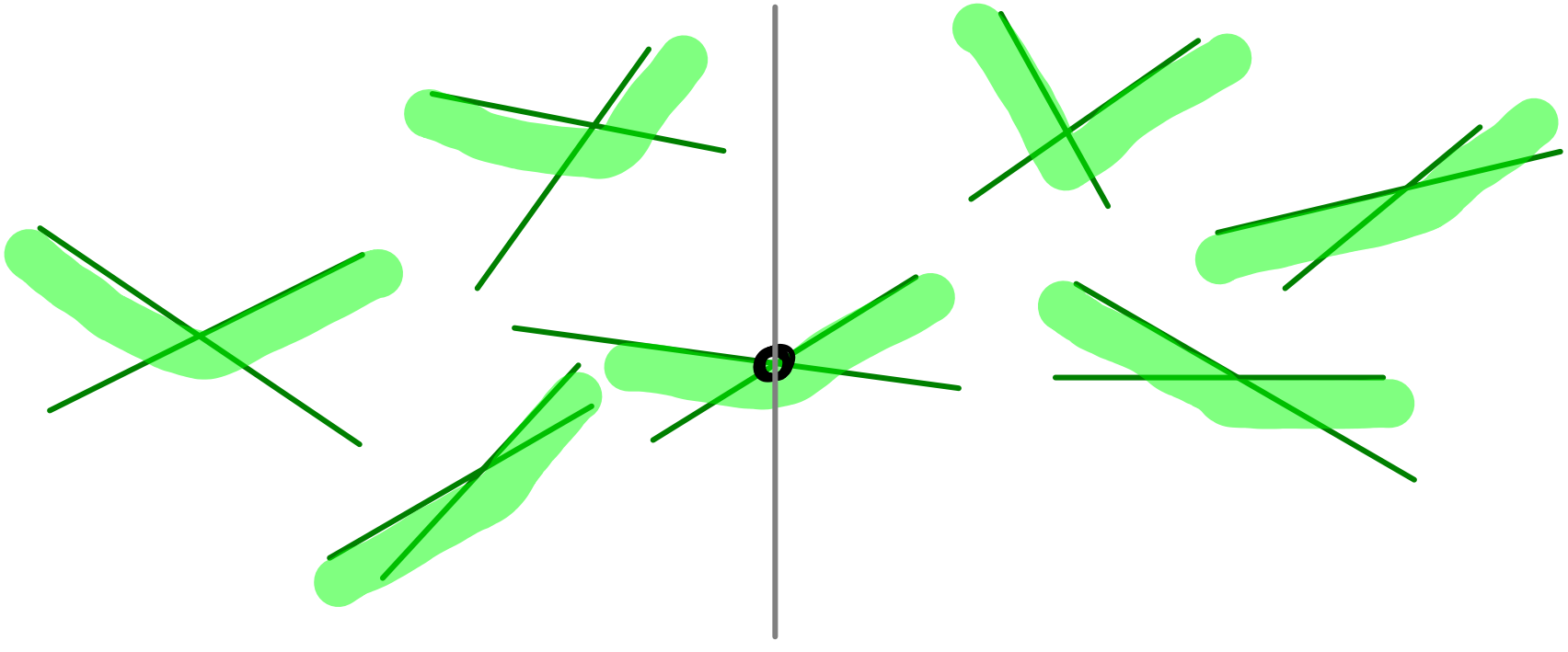




ARBITRARILY PAIR UP THE LOWER LINES

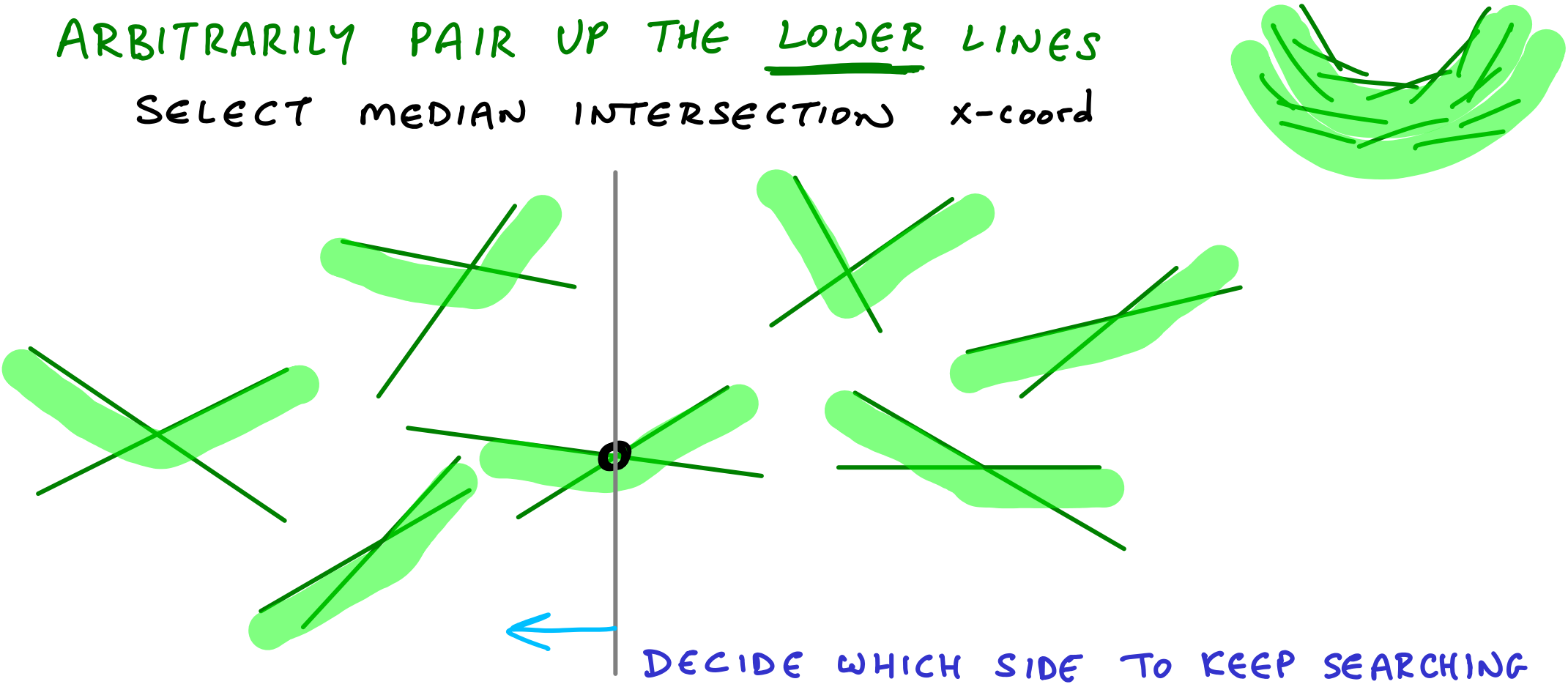
SELECT MEDIAN INTERSECTION x-coord





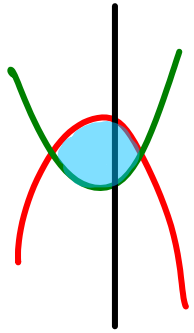
ARBITRARILY PAIR UP THE LOWER LINES

SELECT MEDIAN INTERSECTION x-coord

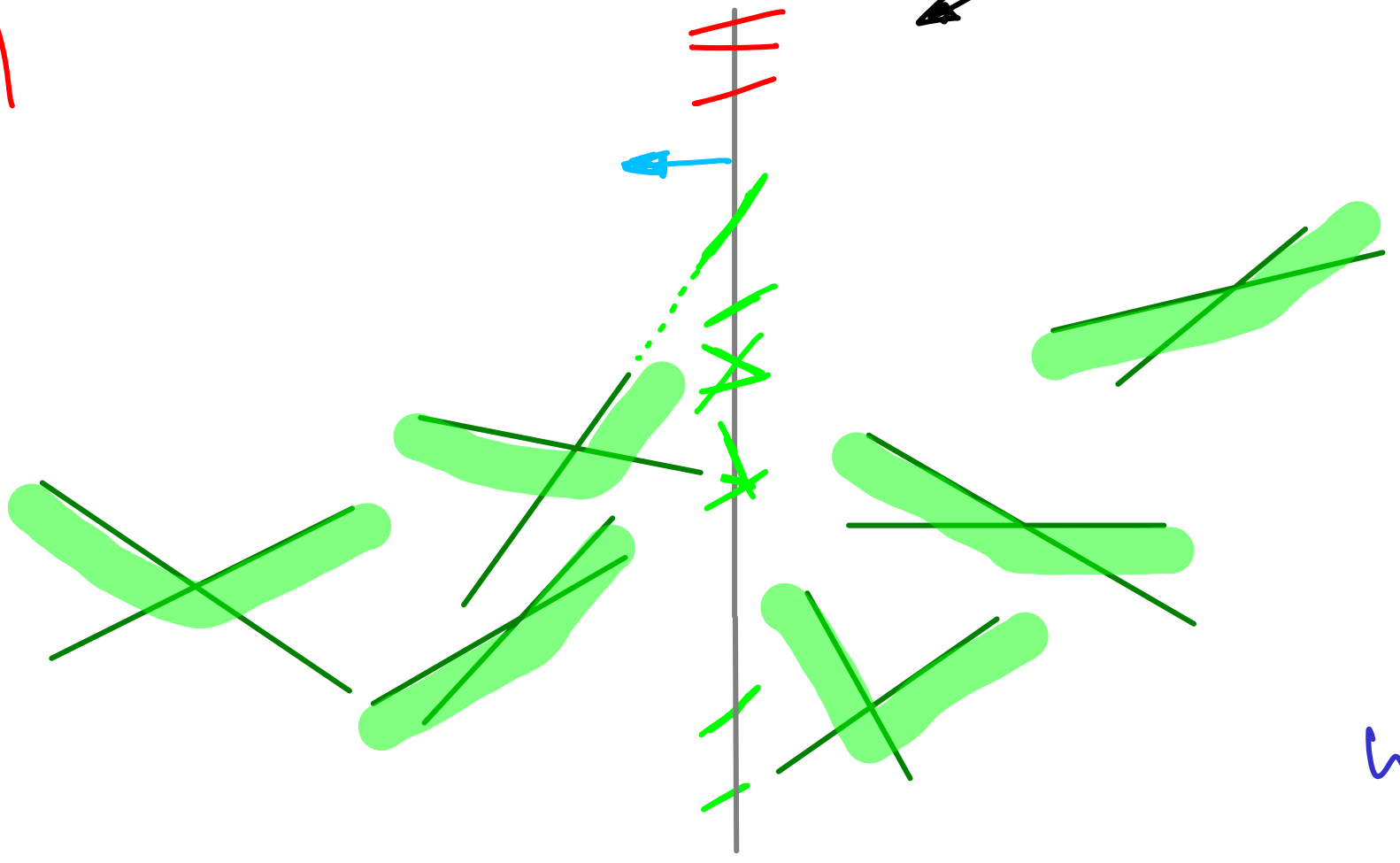


DECIDE WHICH SIDE TO KEEP SEARCHING

(if not done)

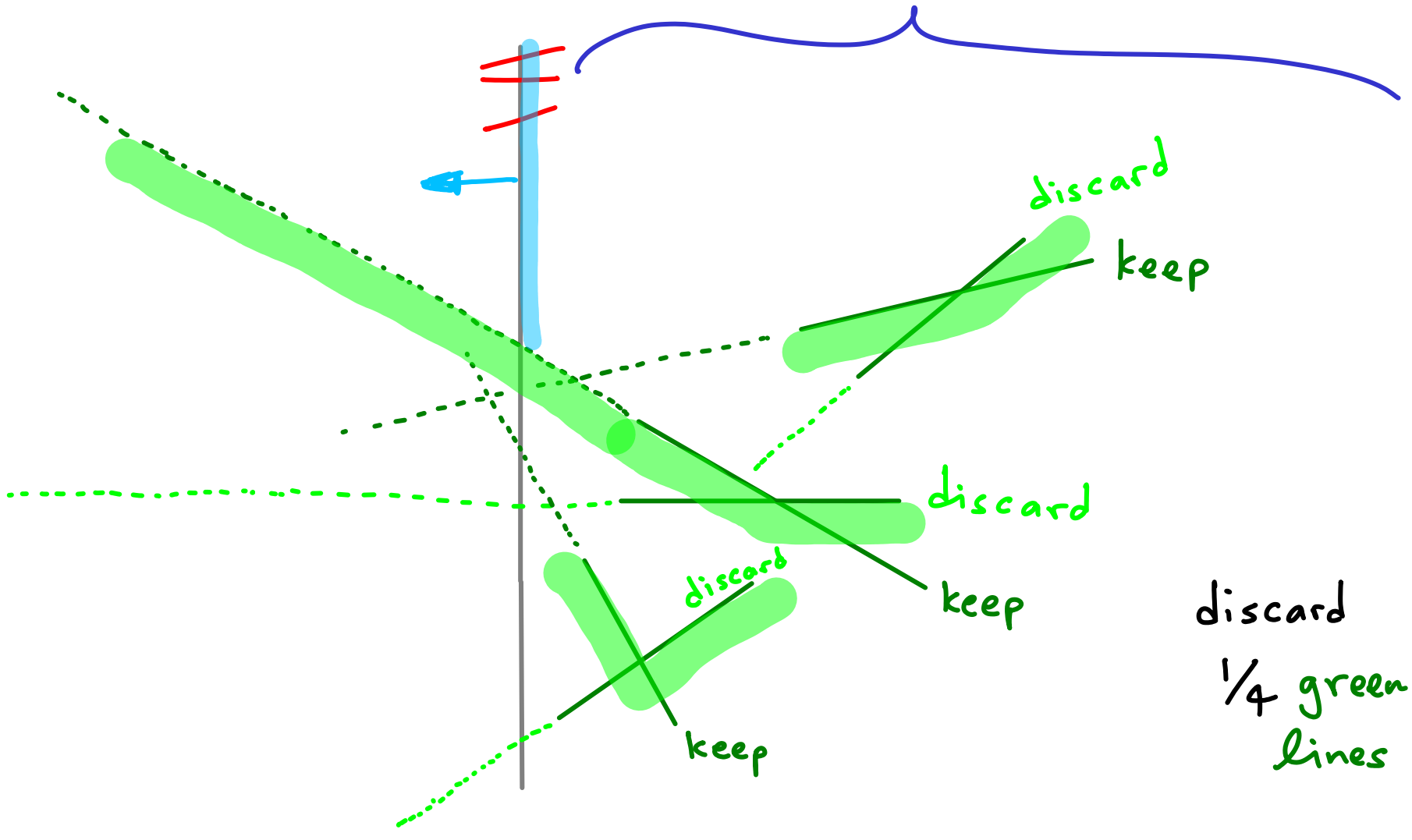


e.g. found intersection & need to optimize



What next?

For half of the pairs, discard one constraint (line)



discard
1/4 green lines

Not good to just keep discarding green : get $\Theta(\log n)$ rounds,
but keep doing $\Theta(n)$ work (still have lots of red)

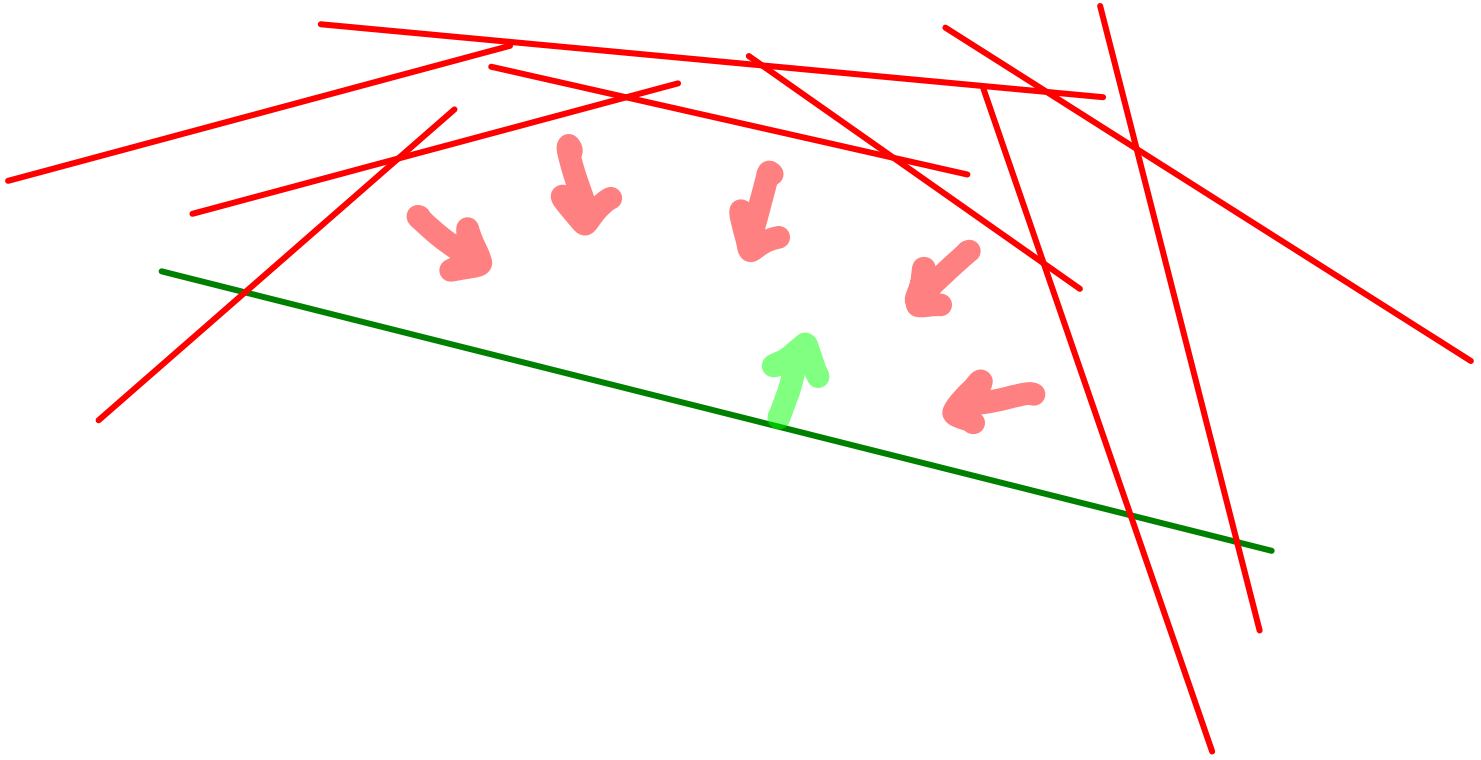
Not good to just keep discarding green : get $\Theta(\log n)$ rounds,
but keep doing $\Theta(n)$ work (still have lots of red)

Could take turns on each set

Not good to just keep discarding green : get $O(\log n)$ rounds,
but keep doing $\Theta(n)$ work (still have lots of red)

Could take turns on each set

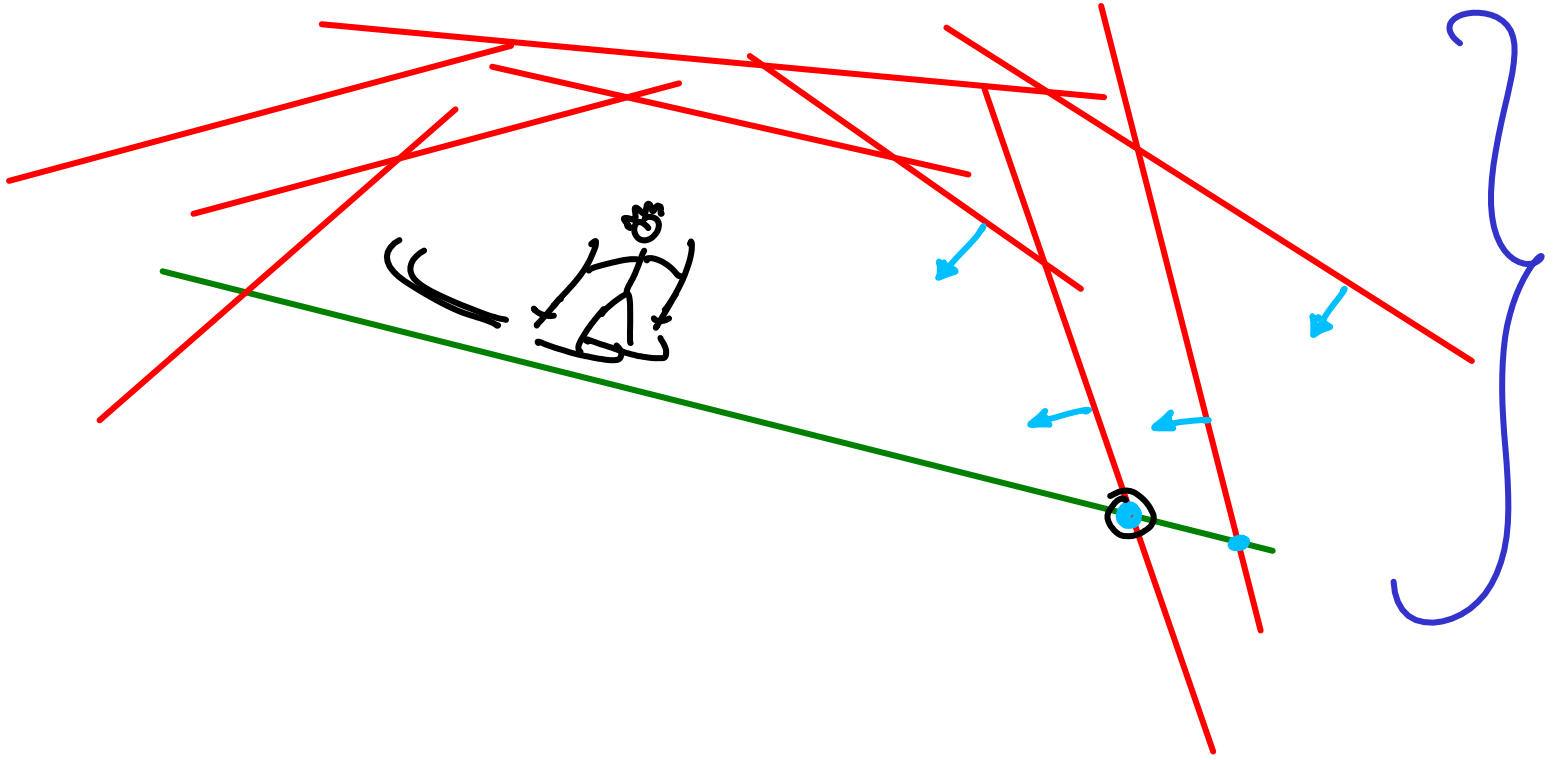
FINALLY, ONE OF THE TWO GROUPS WILL BECOME 1 LINE



Not good to just keep discarding green : get $O(\log n)$ rounds,
but keep doing $\Theta(n)$ work (still have lots of red)

Could take turns on each set

FINALLY, ONE OF THE TWO GROUPS WILL BECOME 1 LINE



JUST COMPUTE ALL INTERSECTIONS OF STEEPER UPPER LINES & PICK HIGHEST

~1D PROBLEM

$O(n)$

RECAP

0) ROTATE COORDINATE SYSTEM

1) PAIR LINES upper w/ upper ; lower w/ lower

2) COMPUTE MEDIAN OF $n/2$ INTERSECTION POINTS

3) PERFORM VERTICAL TEST & FIND SIDE CONTAINING MIN

4) FOR PAIRS ON WRONG SIDE, DISCARD 1 LINE

5) IF STILL >1 LINE IN EACH SET,
REPEAT FROM (1) on 3/4 of the lines

6) EASY SOLUTION WHEN ONLY 1 LINE IN A SET.

$O(n)$

Megiddo has a $O(n)$-time algorithm to solve LP

for any constant dimension