



# Convex Hull Trick

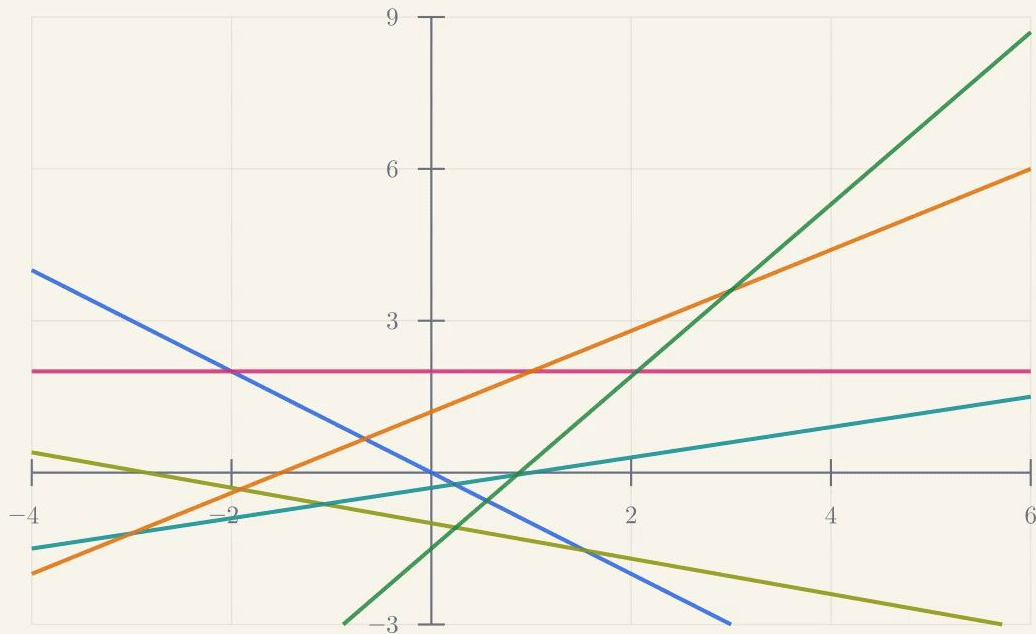
|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

By Bora

# Our motivating question

---

# Our motivating question

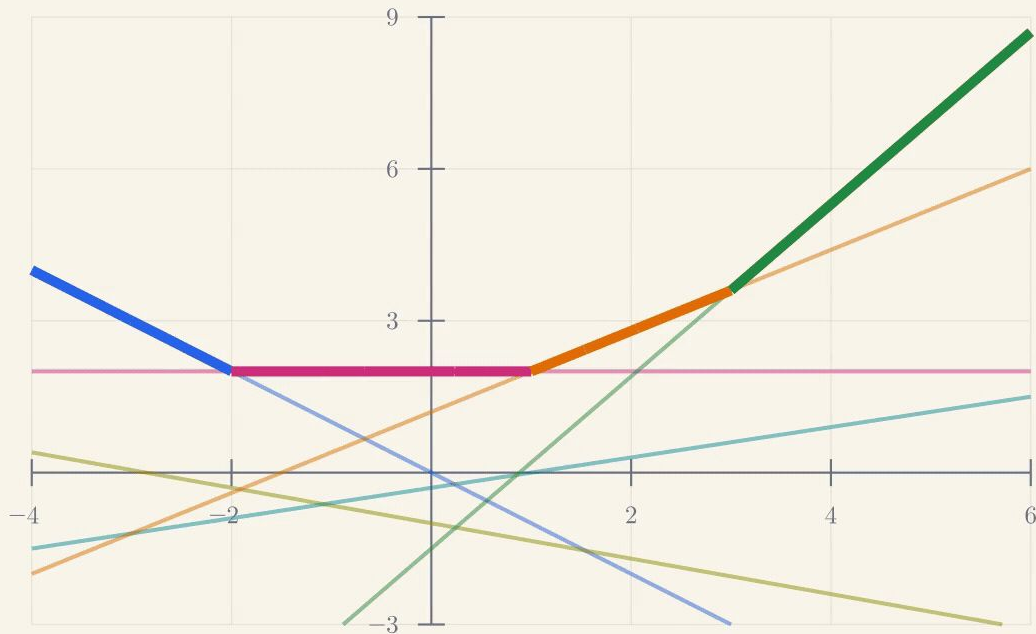


$$f_i(x) = m_i x + b_i$$

$$\text{query}(x) = \max_i f_i(x)$$

At a query point  $x$ , which line gives the largest value?

# Our motivating question

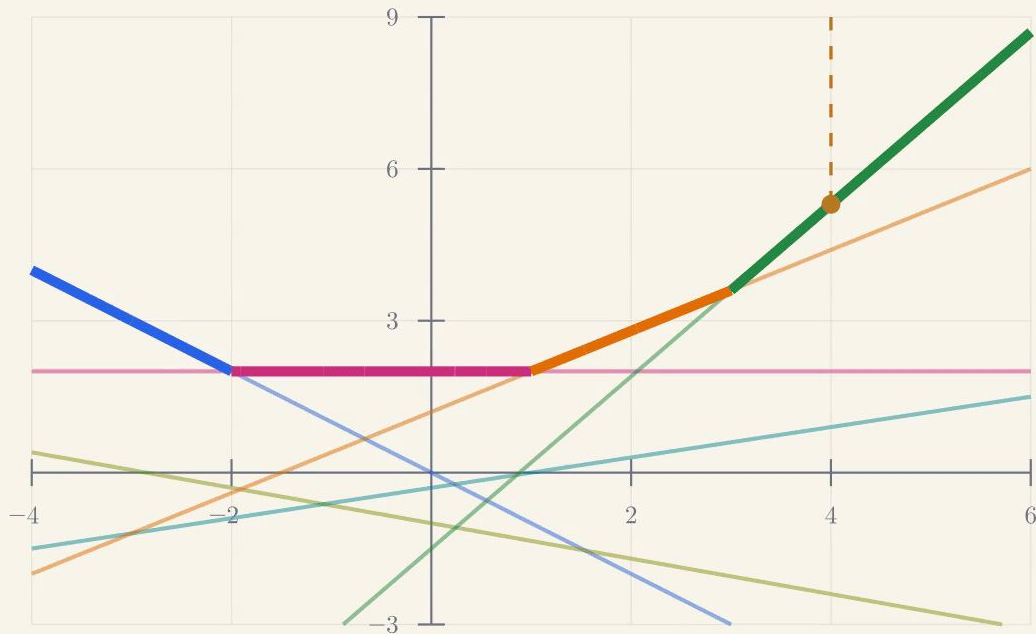


$$f_i(x) = m_i x + b_i$$

$$\text{query}(x) = \max_i f_i(x)$$

The upper hull gives the maximum at each x.

# Our motivating question



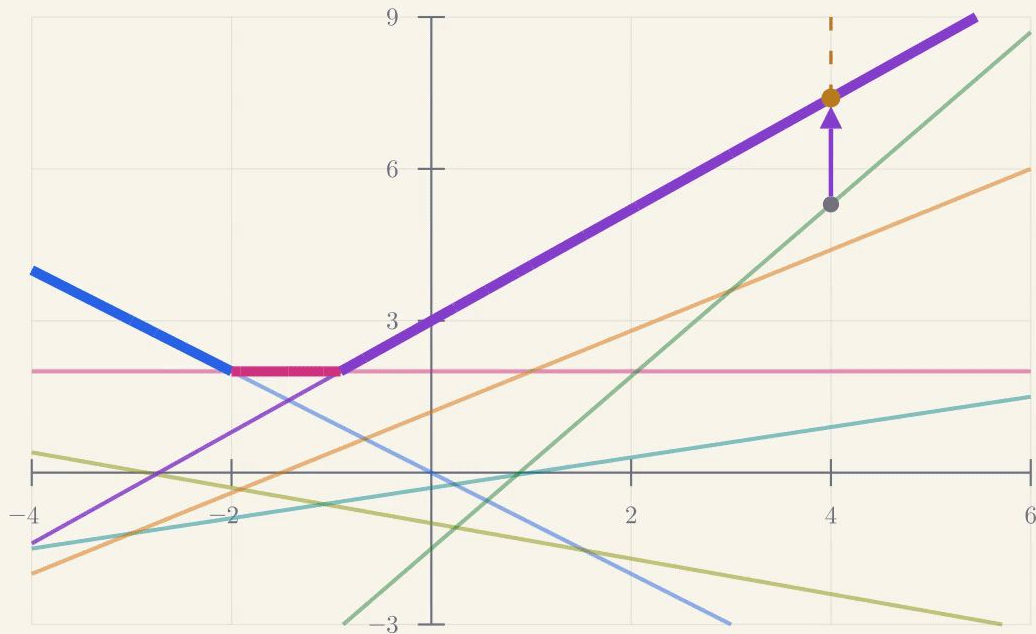
QUERY

$$x = 4.0$$

$$\max_i f_i(x) = 5.3$$

At  $x$ , trace down from the top to the upper hull.

# Our motivating question



QUERY

$$x = 4.0$$

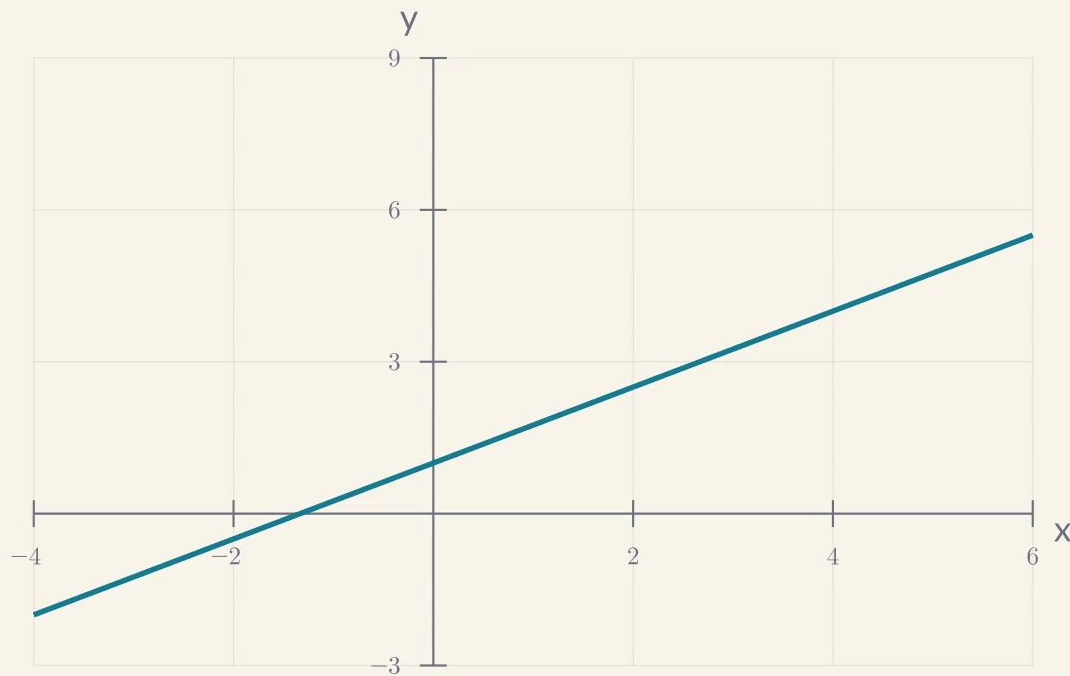
$$\max_i f_i(x) = 7.4$$

Insert a new line: part of the hull changes.

# Lines and intersections

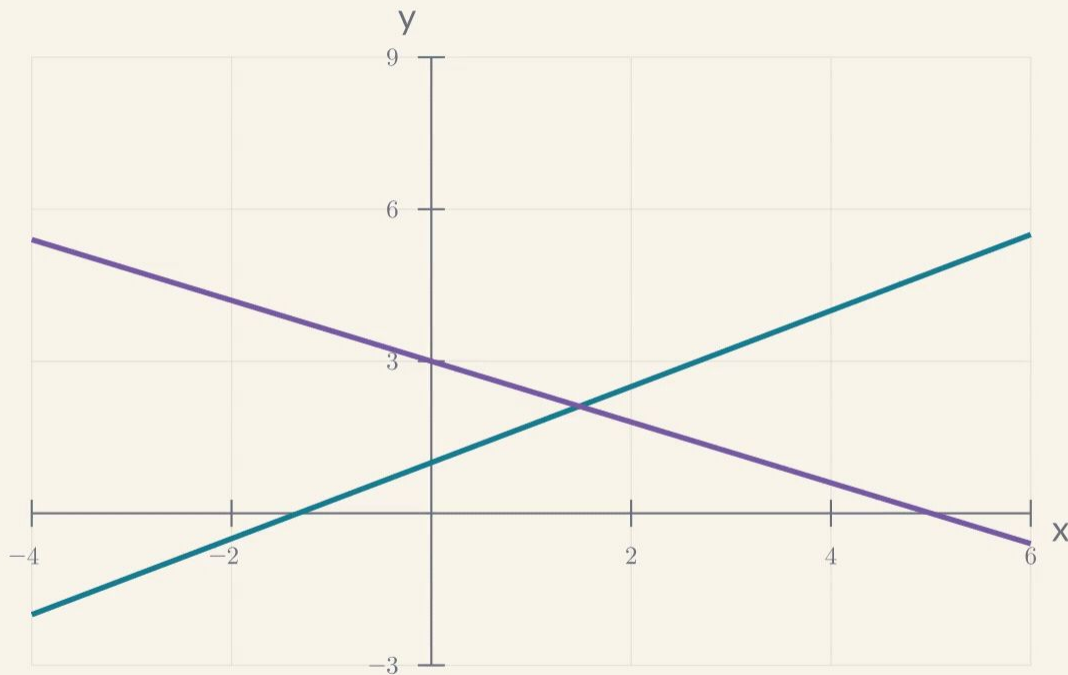
---

# Lines and intersections



$$f(x) = mx + b$$

# Lines and intersections



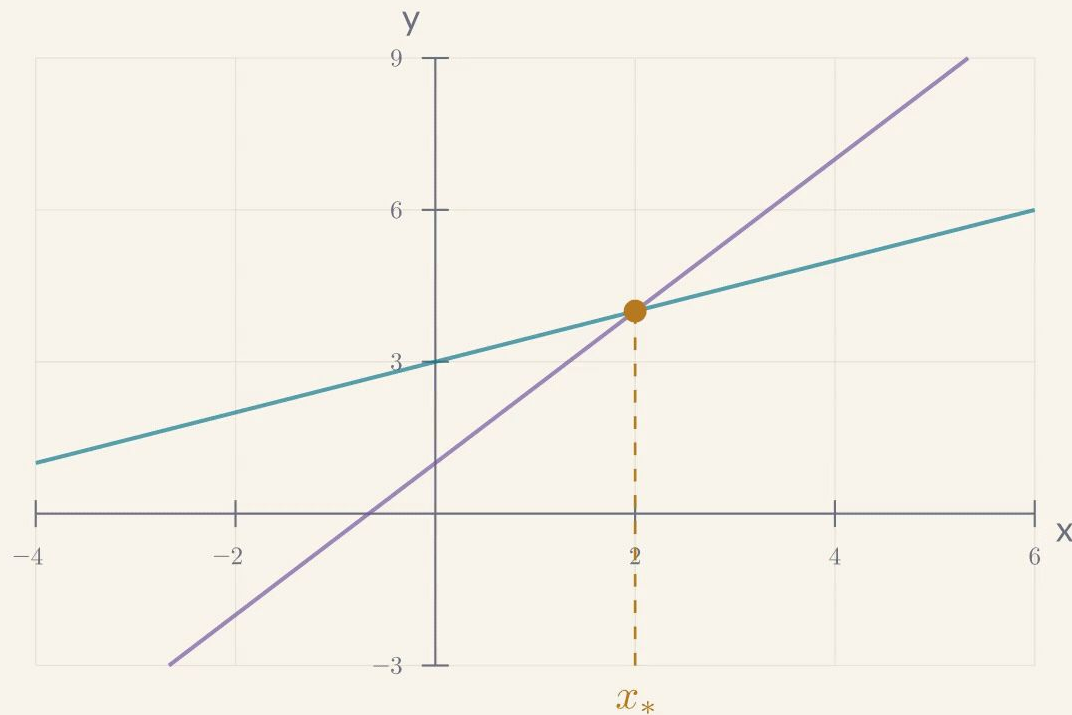
$$f(x) = mx + b$$

$$g(x) = m_2x + b_2$$

# 1 intersection case

---

# 1 intersection case



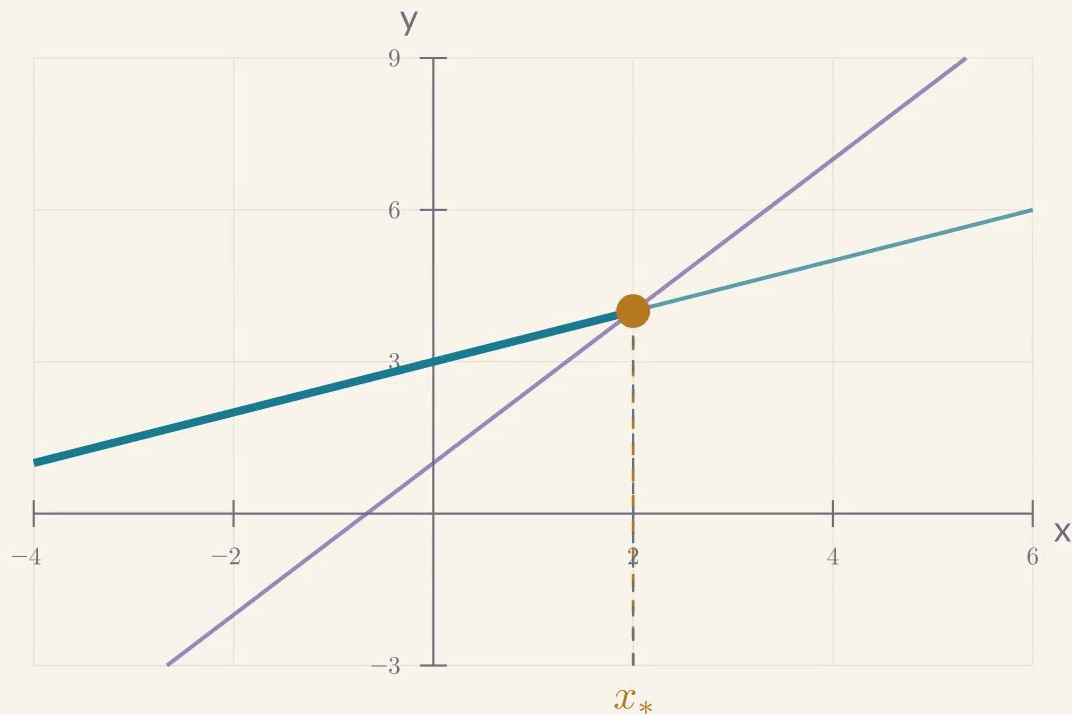
Assume the slopes are ordered

$$f(x) = m_1x + b_1$$

$$g(x) = m_2x + b_2$$

$$m_1 < m_2$$

# 1 intersection case



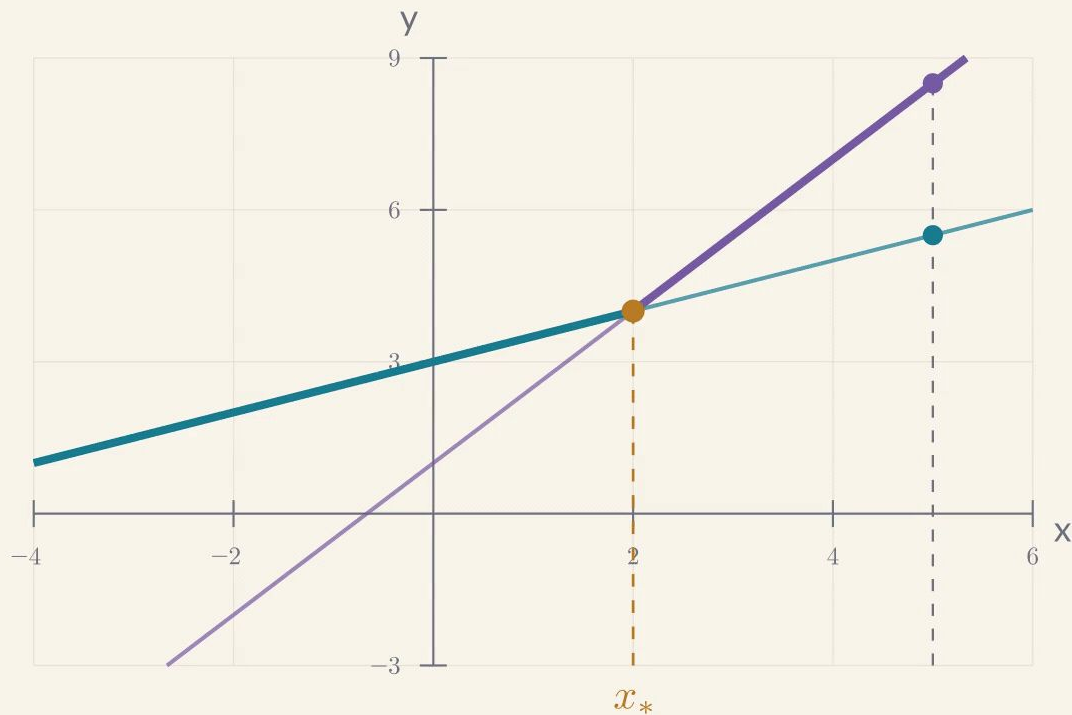
Before the intersection

$$x < x_*$$

$$f(x) > g(x)$$

The smaller slope is larger.

# 1 intersection case



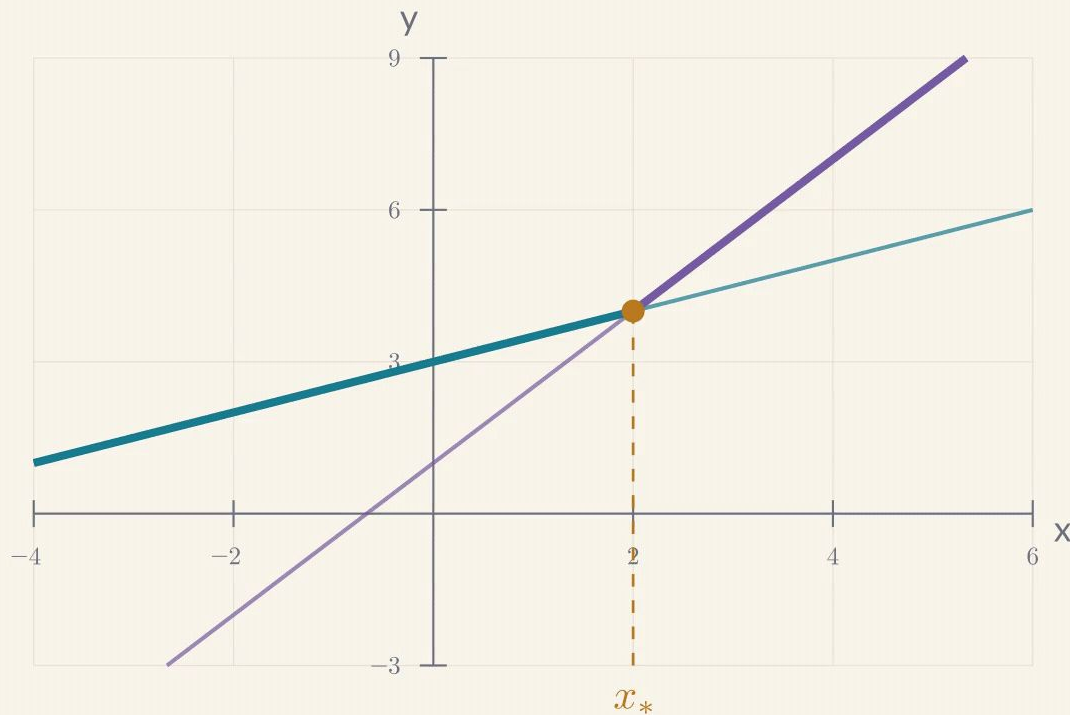
After the intersection

$$x > x_*$$

$$g(x) > f(x)$$

The greater slope stays larger.

# 1 intersection case



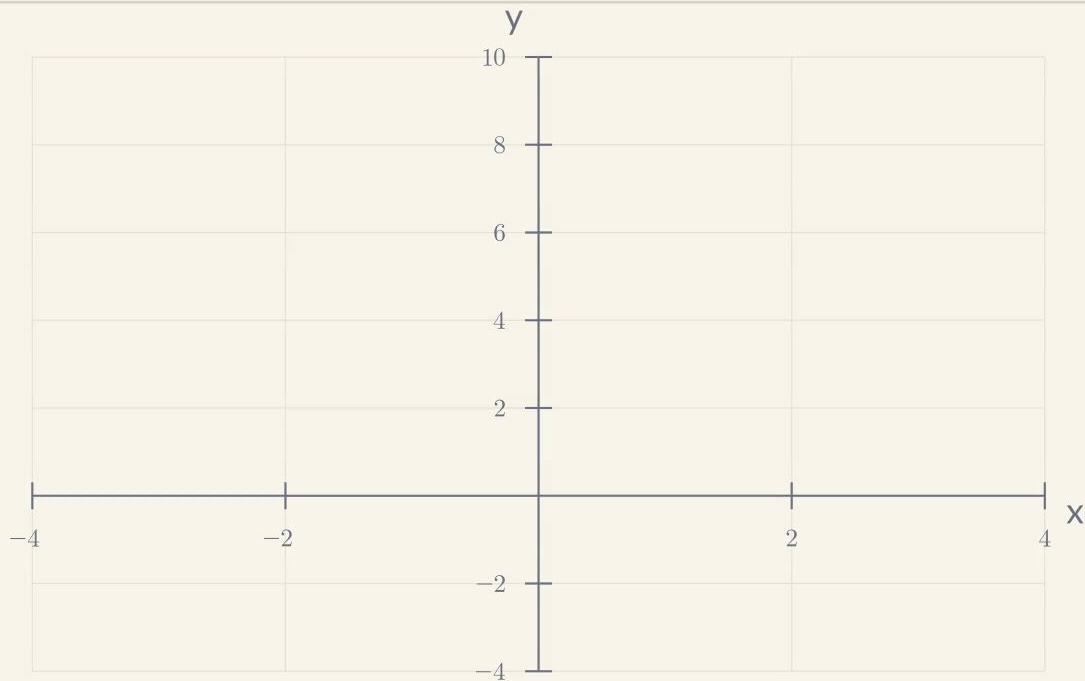
The single-crossing property

Larger slope permanently  
overtakes smaller slope  
after intersection

# Monotonic CHT

---

# Monotonic CHT



Insert by increasing slope

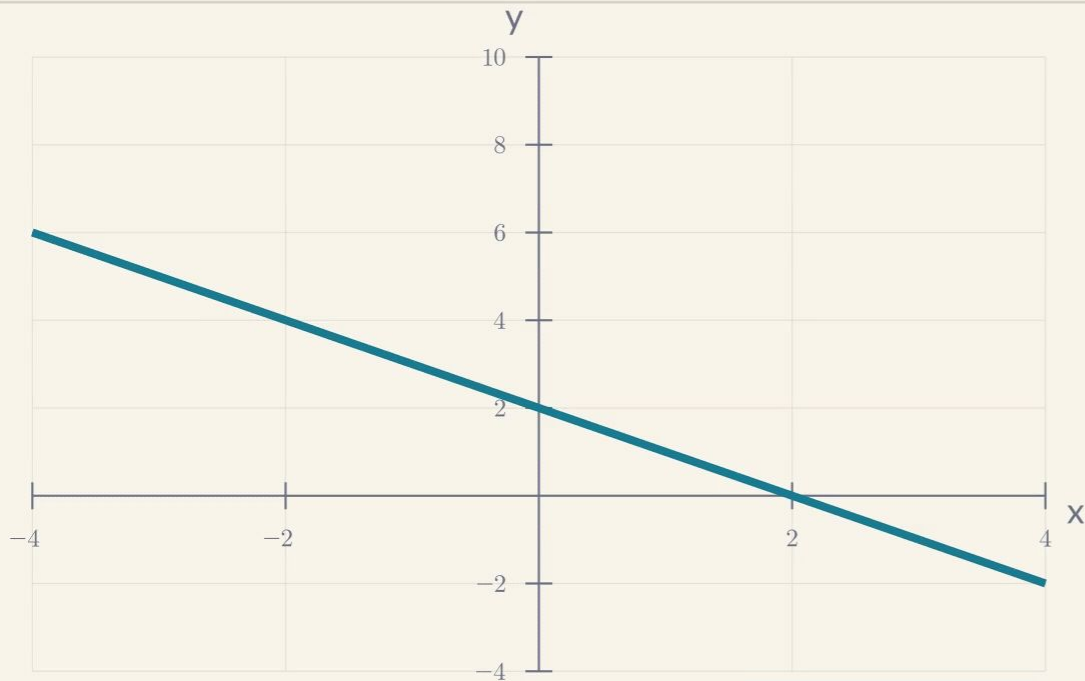
$$m_1 \leq m_2 \leq m_3 \leq \dots$$

Move queries to the right

$$x_1 \leq x_2 \leq x_3 \leq \dots$$

Insert slopes in order. Move query points only to the right.

# Monotonic CHT



$$f(x) = -x + 2$$

First insert a line with slope  $-1$ .

Insert by increasing slope

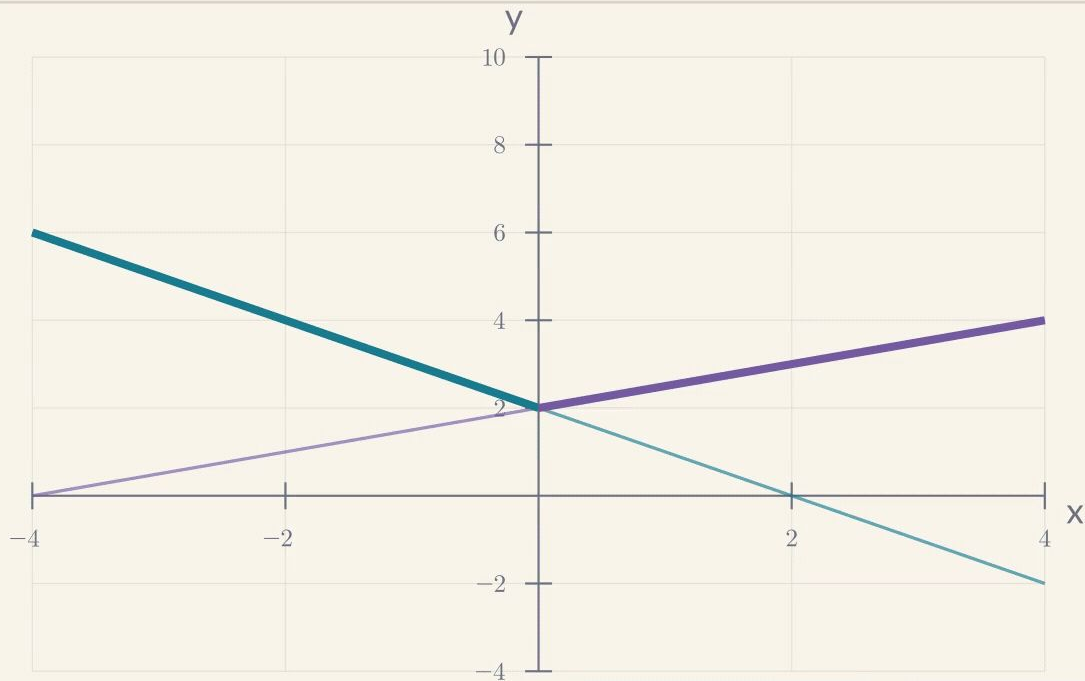
$$m_1 \leq m_2 \leq m_3 \leq \dots$$

Move queries to the right

$$x_1 \leq x_2 \leq x_3 \leq \dots$$

$$f(x) = -x + 2$$

# Monotonic CHT



$$f(x) = -x + 2$$

$$c(x) = \frac{1}{2}x + 2$$

Insert by increasing slope

$$m_1 \leq m_2 \leq m_3 \leq \dots$$

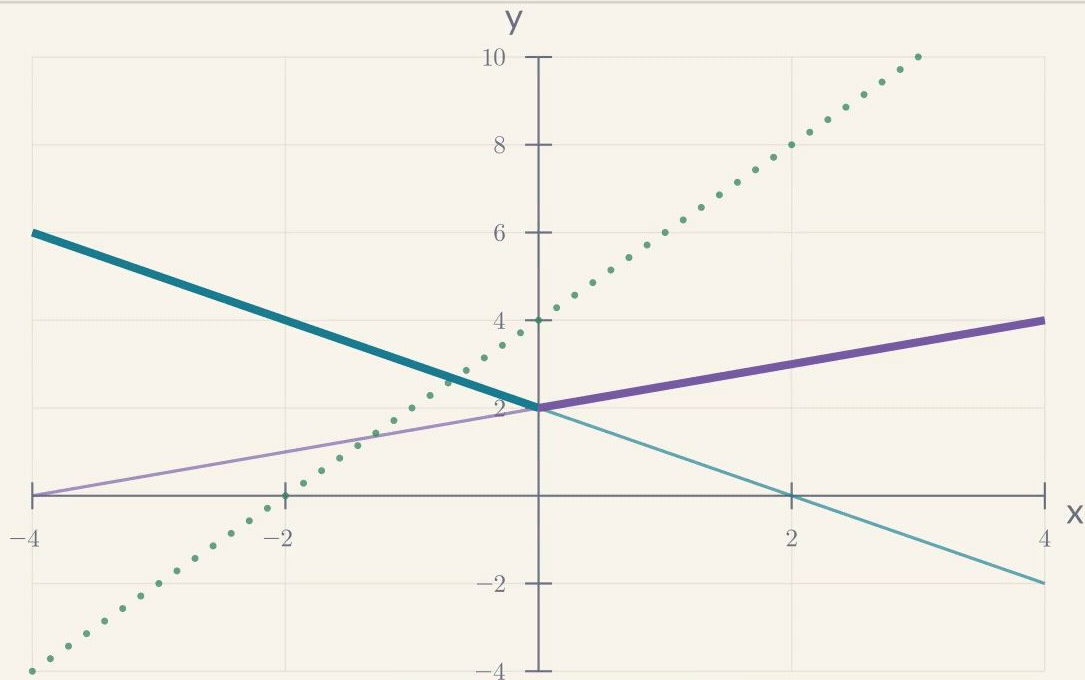
Move queries to the right

$$x_1 \leq x_2 \leq x_3 \leq \dots$$

$$c(x) = \frac{1}{2}x + 2$$

Blue and purple both contribute to the hull.

# Monotonic CHT



Insert by increasing slope

$$m_1 \leq m_2 \leq m_3 \leq \dots$$

Move queries to the right

$$x_1 \leq x_2 \leq x_3 \leq \dots$$

$$g(x) = 2x + 4$$

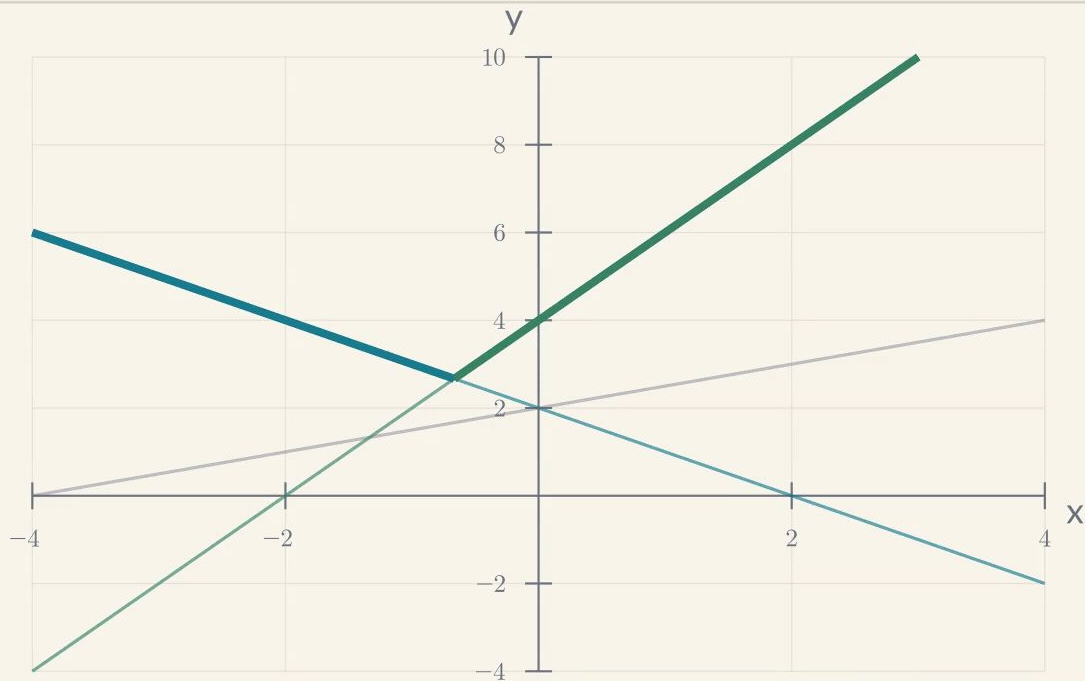
$$f(x) = -x + 2$$

$$c(x) = \frac{1}{2}x + 2$$

$$g(x) = 2x + 4$$

Insert the green line as a dotted candidate.

# Monotonic CHT



Insert by increasing slope

$$m_1 \leq m_2 \leq m_3 \leq \dots$$

Move queries to the right

$$x_1 \leq x_2 \leq x_3 \leq \dots$$

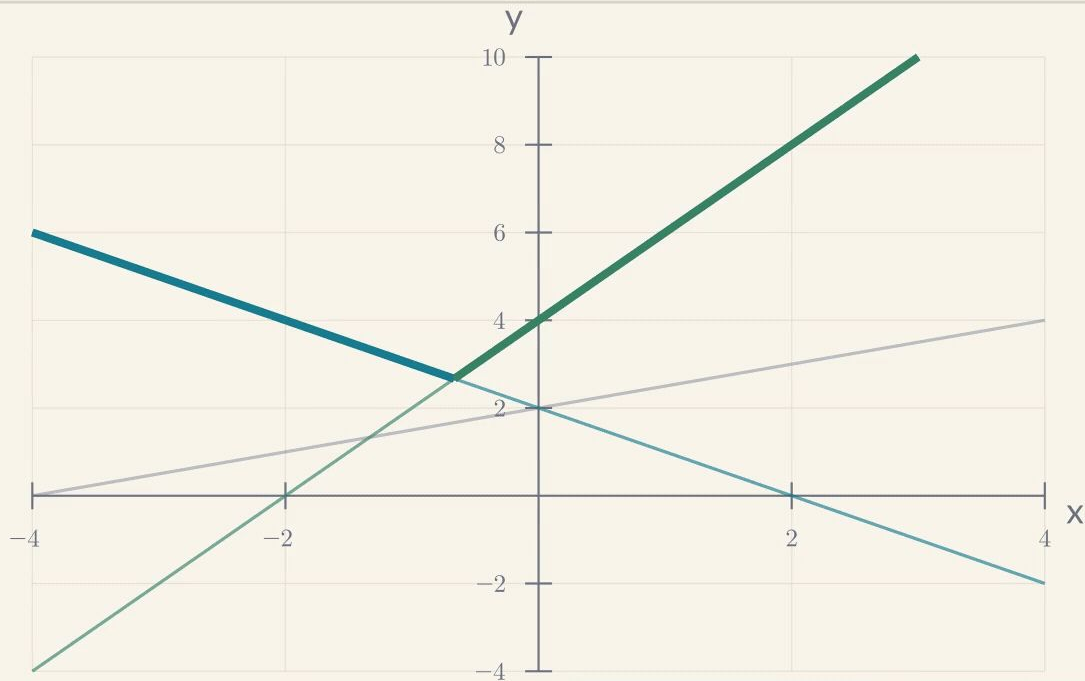
$$g(x) = 2x + 4$$

$$f(x) = -x + 2$$

$$g(x) = 2x + 4$$

Gray out purple and make green solid.

# Monotonic CHT



Idea: Keep track only of useful lines

$$f(x) = -x + 2$$

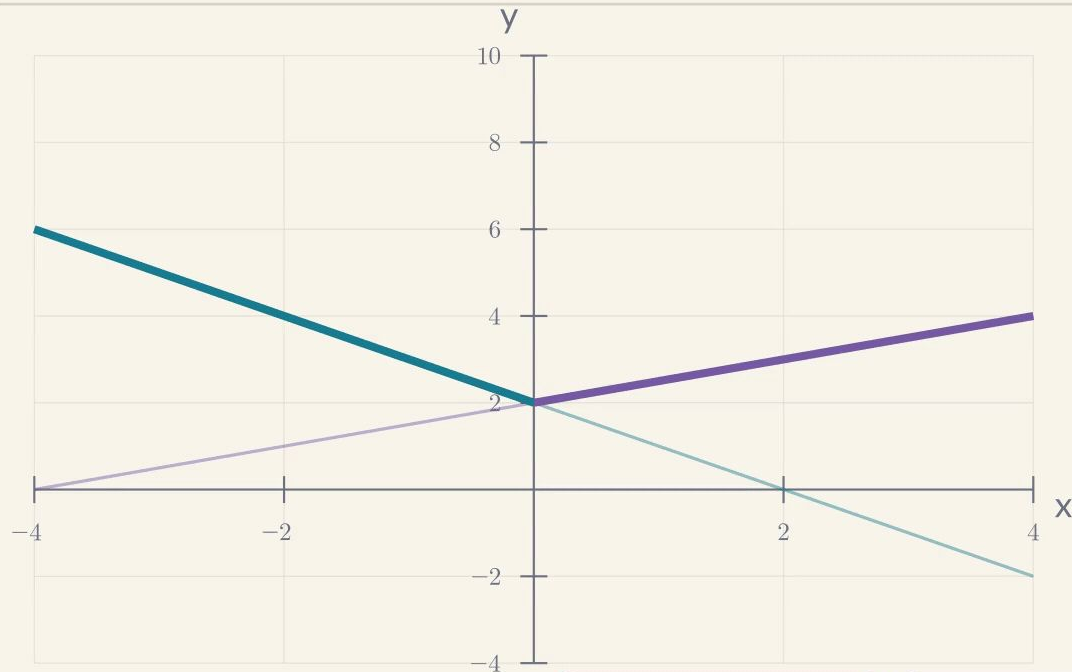
$$g(x) = 2x + 4$$

Store only the lines that contribute to the upper hull.

# Line insertions

---

# Line insertions



The last two + a new line

$$f(x) = -x + 2$$

$$c(x) = \frac{1}{2}x + 2$$

$$g(x) = 2x + 4$$

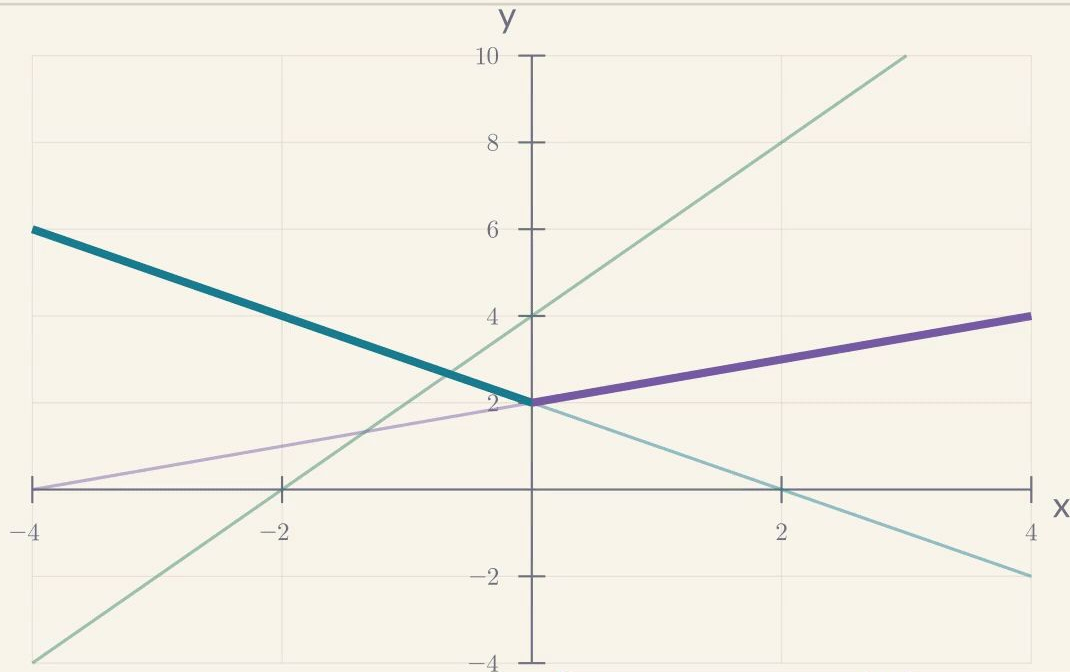
$$m_f < m_c < m_g$$

$$f(x) = -x + 2$$

$$c(x) = \frac{1}{2}x + 2$$

Look at the last two stored lines,  $f$  and  $c$ .

# Line insertions



The last two + a new line

$$f(x) = -x + 2$$

$$c(x) = \frac{1}{2}x + 2$$

$$g(x) = 2x + 4$$

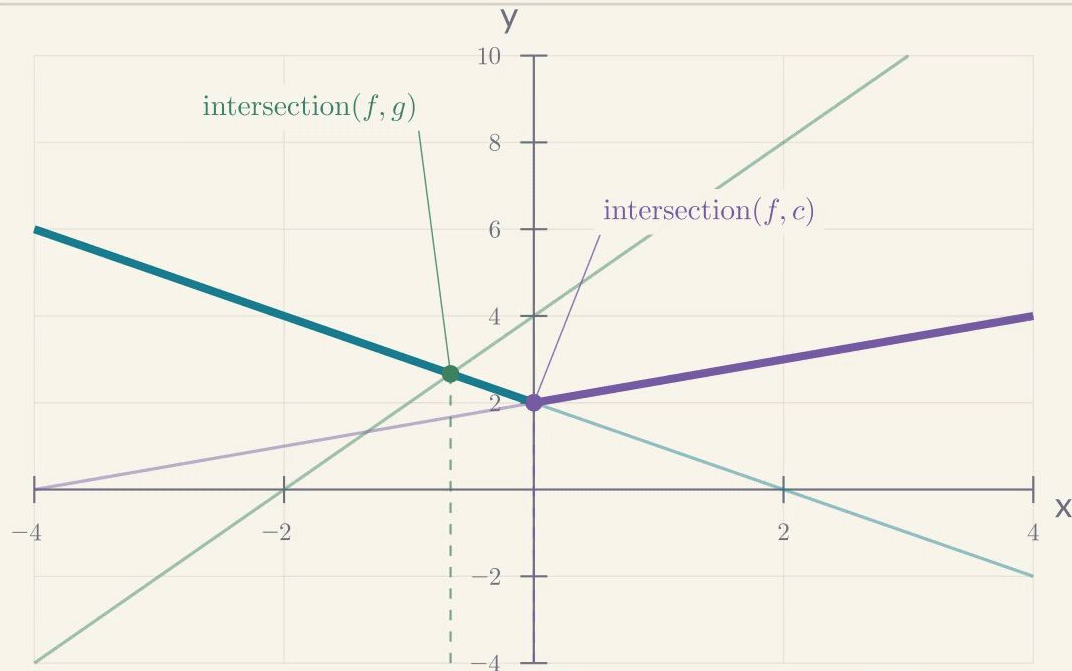
$$m_f < m_c < m_g$$

$$f(x) = -x + 2$$

$$c(x) = \frac{1}{2}x + 2$$

The new line eventually takes over. Does c survive?

# Line insertions



$$f(x) = -x + 2$$

$$c(x) = \frac{1}{2}x + 2$$

Compare the crossings

$$\text{intersection}(a, b) = \frac{b_a - b_b}{m_b - m_a}$$

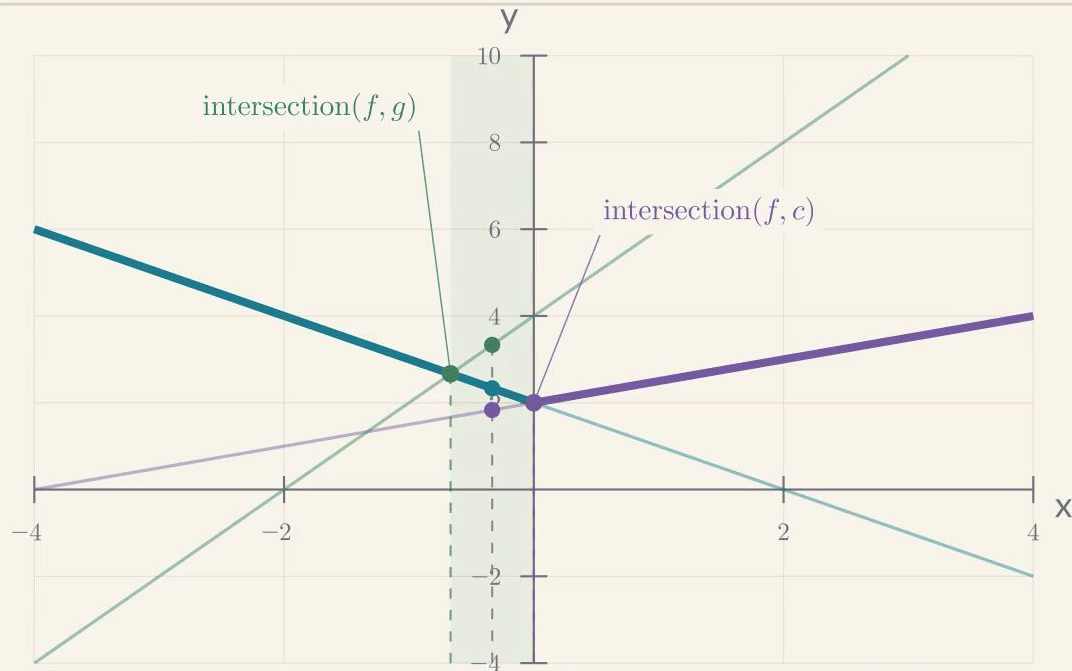
$$\text{intersection}(f, c) = 0$$

$$\text{intersection}(f, g) = -\frac{2}{3}$$

Before  $x = 0$ :  $f(x) \geq c(x)$ .

Use the intersection formula from the previous section.

# Line insertions



$$f(x) = -x + 2$$

$$c(x) = \frac{1}{2}x + 2$$

Transitivity at the same  $x$

$$-\frac{2}{3} \leq x \leq 0$$

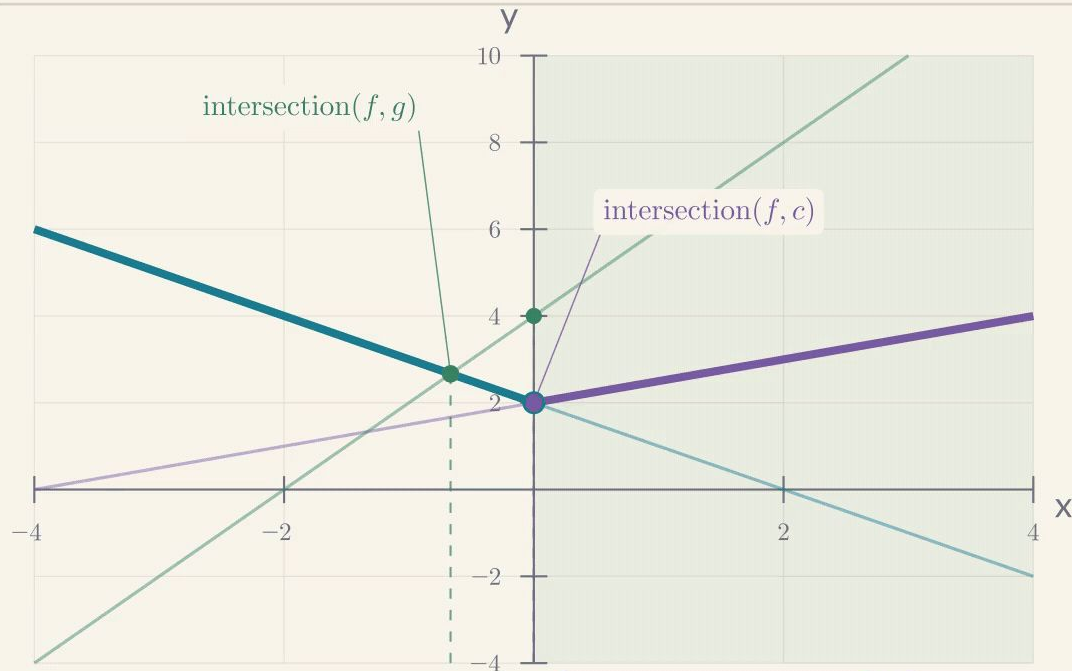
$$g(x) \geq f(x) \geq c(x)$$

$g$  is at least  $f$ ;  $f$  is at least  $c$ .

$$\implies g(x) \geq c(x)$$

Between these crossings:  $g \geq f \geq c$ , so  $g \geq c$ .

# Line insertions



$$f(x) = -x + 2$$

$$c(x) = \frac{1}{2}x + 2$$

What happens after that?

$$g(0) = 4 > f(0) = c(0) = 2$$

$$m_g = 2 > m_c = \frac{1}{2}$$

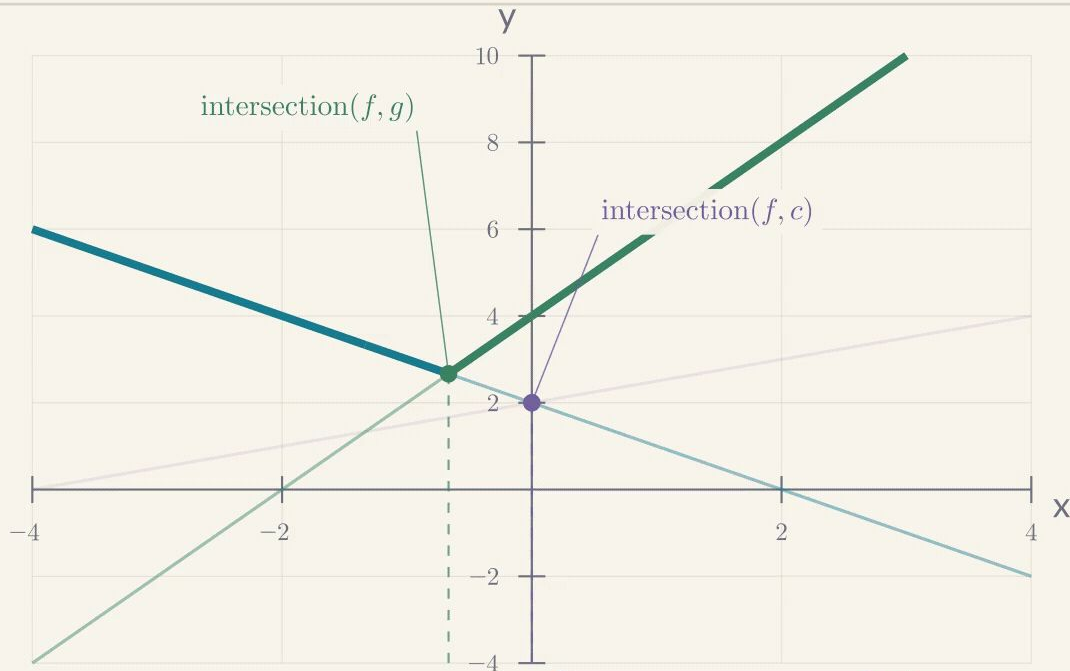
Before  $x = 0$ :  $f$  covers  $c$ .

After  $x = 0$ :  $g$  stays above  $c$ .

**$c$  never needs to be the winner.**

At  $x = 0$ ,  $f = c$ , but  $g$  is already higher. Its larger slope keeps it higher.

# Line insertions



$$f(x) = -x + 2$$

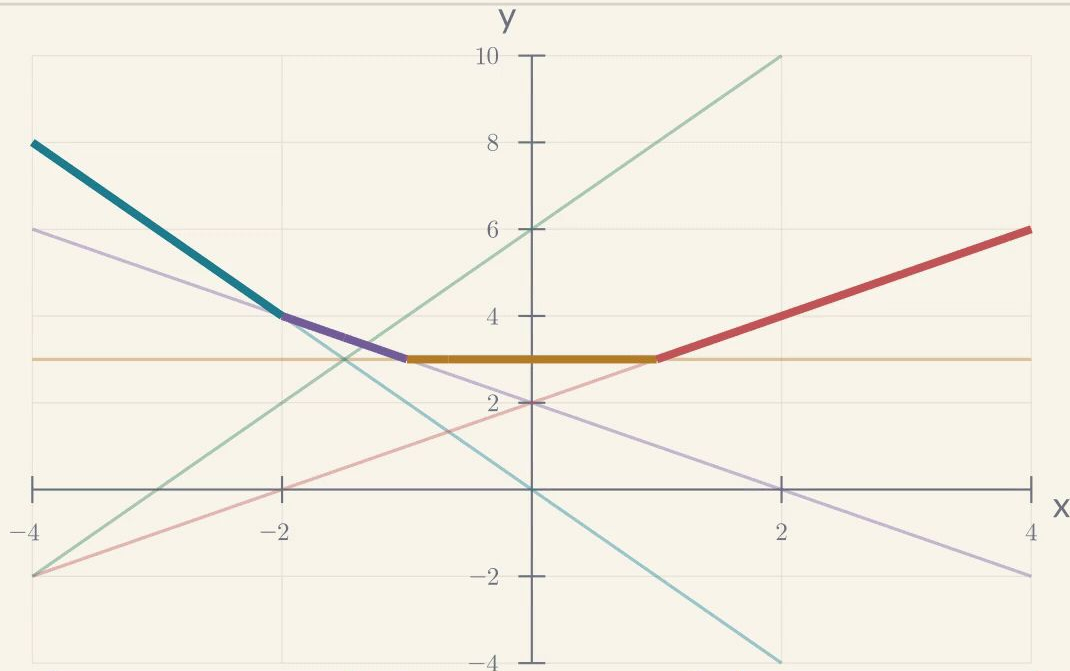
$$g(x) = 2x + 4$$

The middle line is useless if  
 $\text{intersection}(f, g) \leq \text{intersection}(f, c)$   
 g intersects f before c

$$-\frac{2}{3} \leq 0$$

Remove the last line, then test the new last two.

# Line insertions



One insertion, several removals

$$A = -2x, \quad B = -x + 2$$

$$C = 3, \quad D = x + 2$$

$$G = 2x + 6$$

Test C, D, G first.

$$A(x) = -2x$$

$$B(x) = -x + 2$$

$$C(x) = 3$$

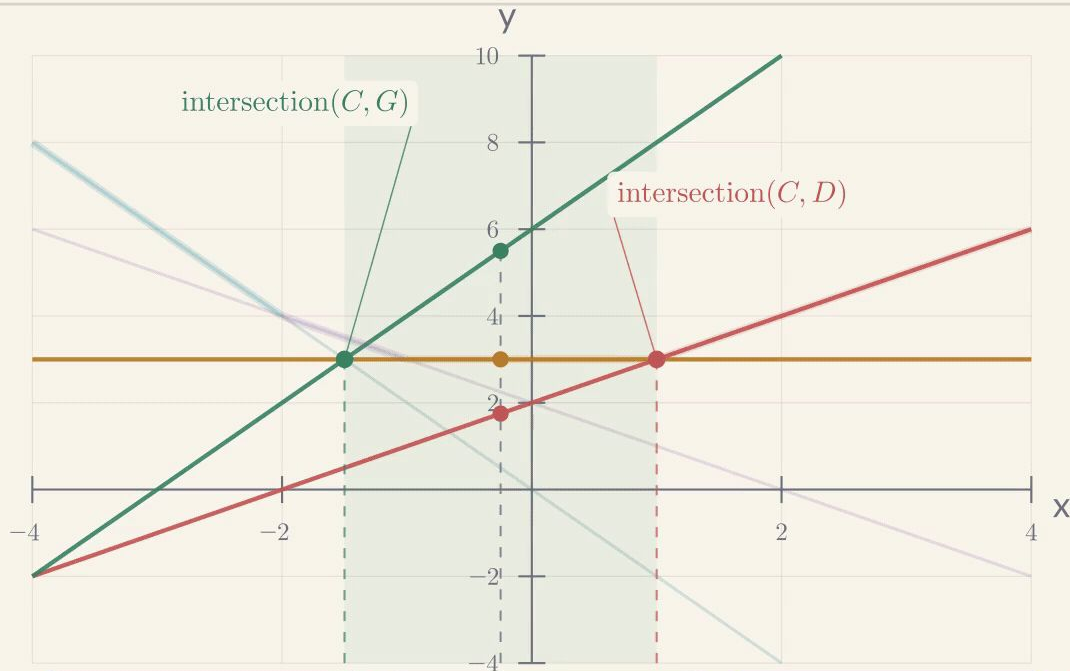
$$D(x) = x + 2$$

$$G(x) = 2x + 6$$

new

A single insertion can remove several old lines.

# Line insertions



Transitivity at the same  $x$

$$-\frac{3}{2} \leq x \leq 1$$

$$G(x) \geq C(x) \geq D(x)$$

$G$  is at least  $C$ ;  $C$  is at least  $D$ .

$$\Rightarrow G(x) \geq D(x)$$

$$A(x) = -2x$$

$$B(x) = -x + 2$$

$$C(x) = 3$$

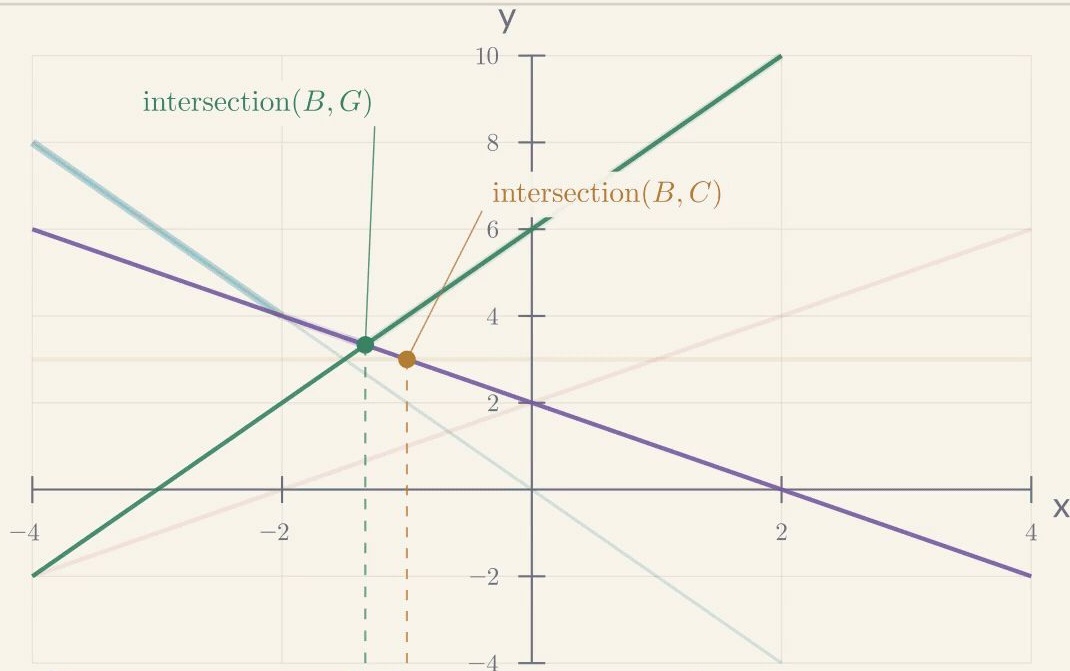
$$D(x) = x + 2$$

new

$$G(x) = 2x + 6$$

Between these crossings:  $G \geq C \geq D$ , so  $G \geq D$ .

# Line insertions



$$A(x) = -2x$$

$$B(x) = -x + 2$$

new

$$G(x) = 2x + 6$$

What happens after that?

$$G(-1) = 4 > B(-1) = C(-1) = 3$$

$$m_G = 2 > m_C = 0$$

Before  $x = -1$ : B covers C.

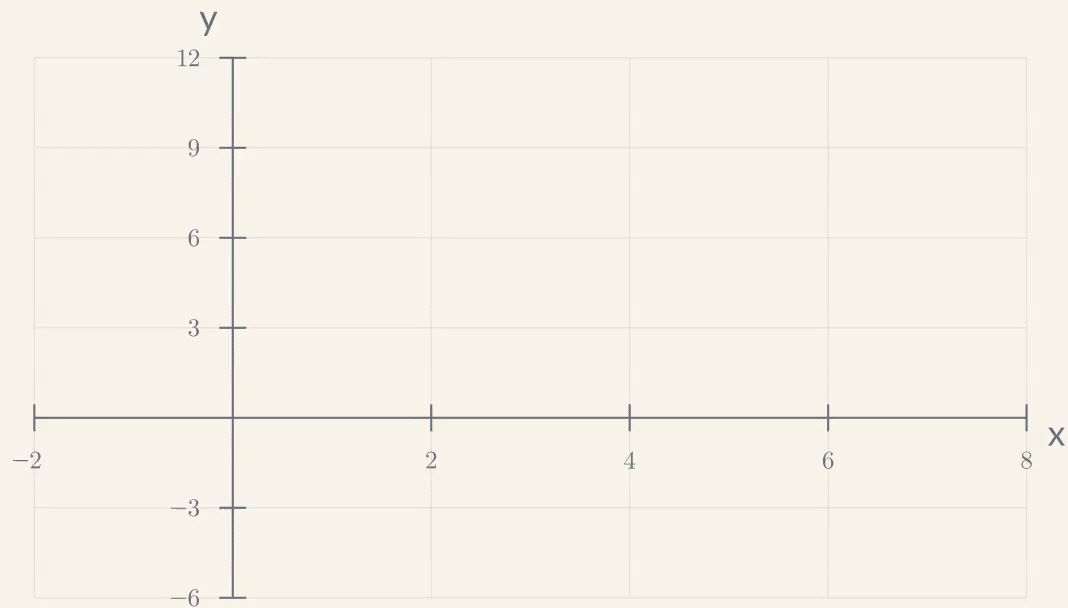
After  $x = -1$ : G stays above C.

C never needs to be the winner.

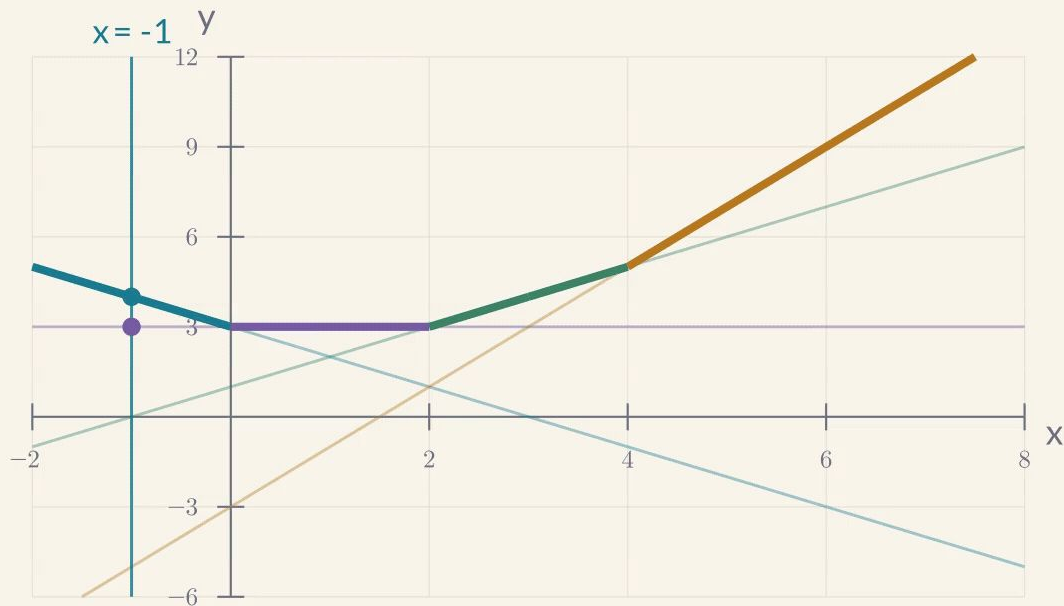
B covers C on the left. G covers C on the right. Pop C.

# Querying

---



# Querying



Query  $x = -1$

$$f_0(-1) = 4 > f_1(-1) = 3$$

$$\max_i f_i(-1) = 4$$

Blue wins this query.

Keep all four lines.

$$f_0(x) = -x + 3$$

front

$$f_1(x) = 3$$

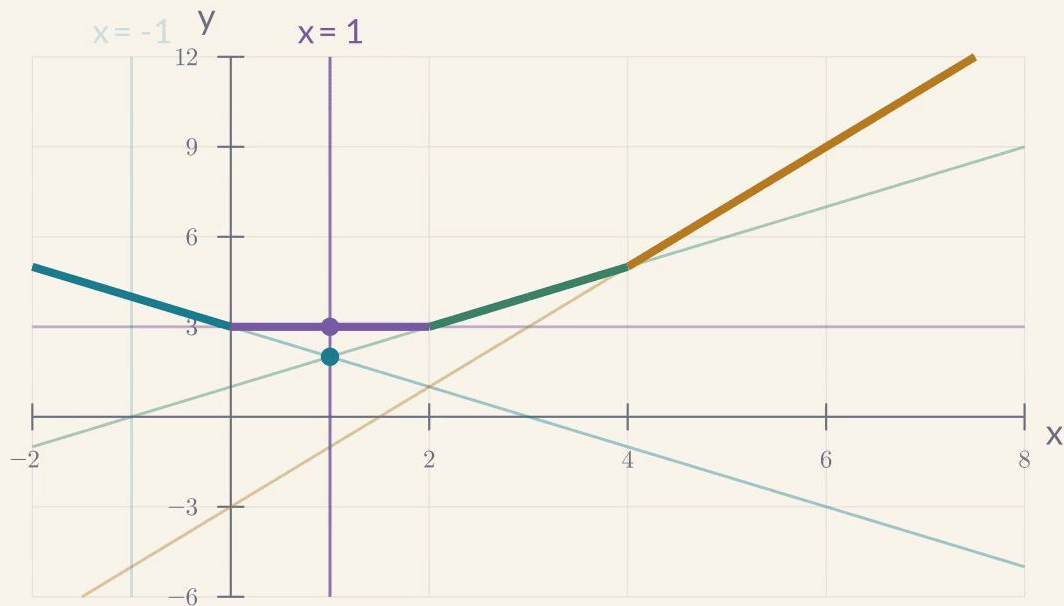
$$f_2(x) = x + 1$$

$$f_3(x) = 2x - 3$$

back

At  $-1$ , blue gives 4. No line is retired.

# Querying



Query  $x = 1$

$$f_0(1) = 2 < f_1(1) = 3$$

Purple has overtaken blue.

$$\max_i f_i(1) = 3$$

$$f_0(x) = -x + 3$$

front

$$f_1(x) = 3$$

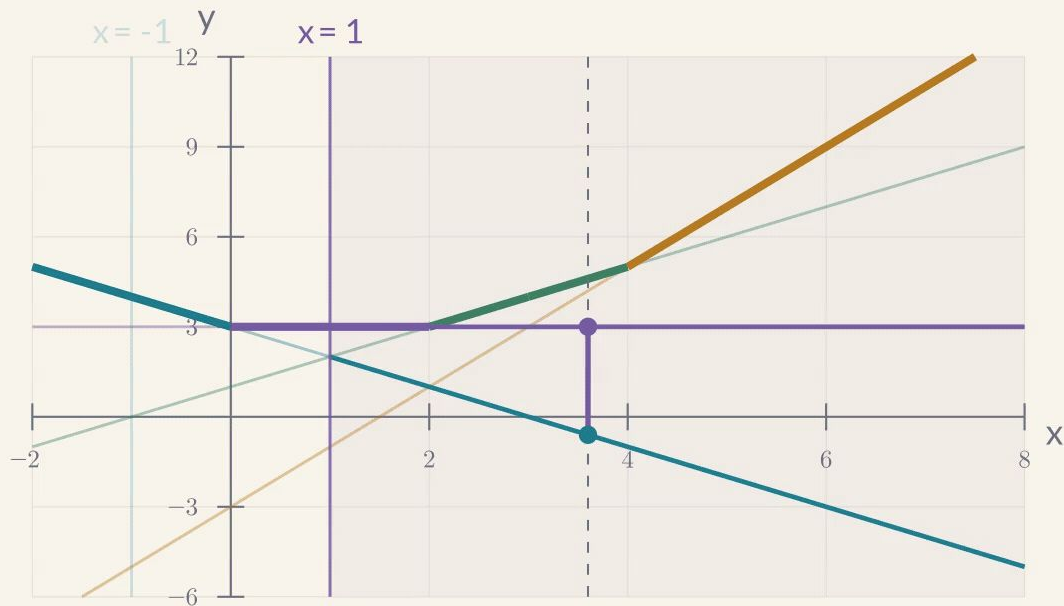
$$f_2(x) = x + 1$$

$$f_3(x) = 2x - 3$$

back

At 1,  $f_0(1) < f_1(1)$ : purple has overtaken blue.

# Querying



$$f_0(x) = -x + 3$$

front

$$f_1(x) = 3$$

$$f_2(x) = x + 1$$

$$f_3(x) = 2x - 3$$

back

For every  $x \geq 1$

$$f_1(x) - f_0(x) = x \geq 1$$

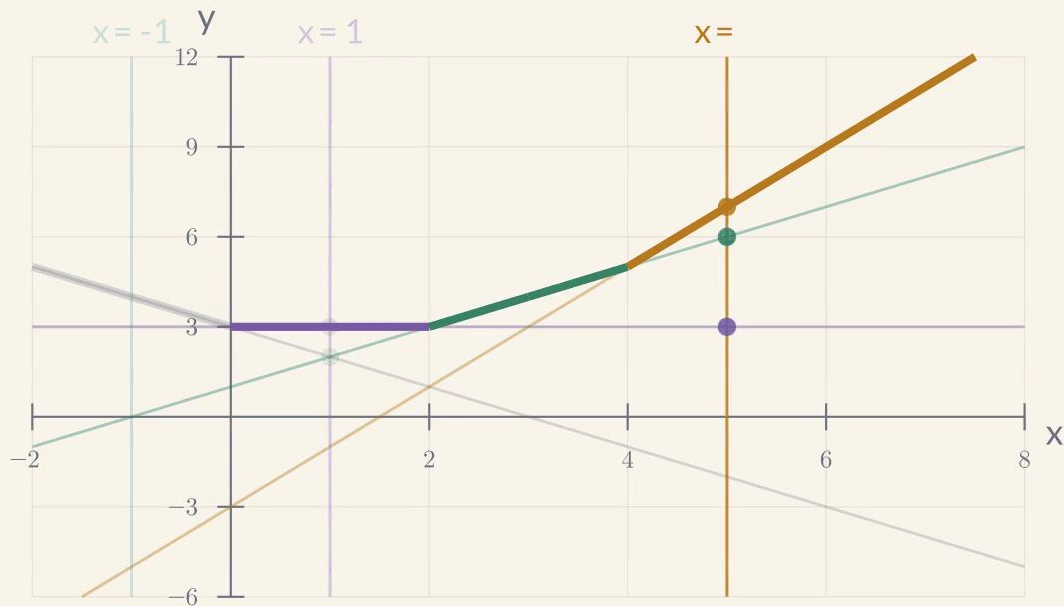
Purple stays above blue.

Their gap keeps growing.

Blue cannot win a future query.

To the right of 1, purple stays above blue forever.

# Querying



Query  $x = 5$

$$f_1(5) = 3 < f_3(5) = 7$$

$$f_2(5) = 6 < f_3(5) = 7$$

Yellow has overtaken both purple and green.

$$f_1(x) = 3$$

front

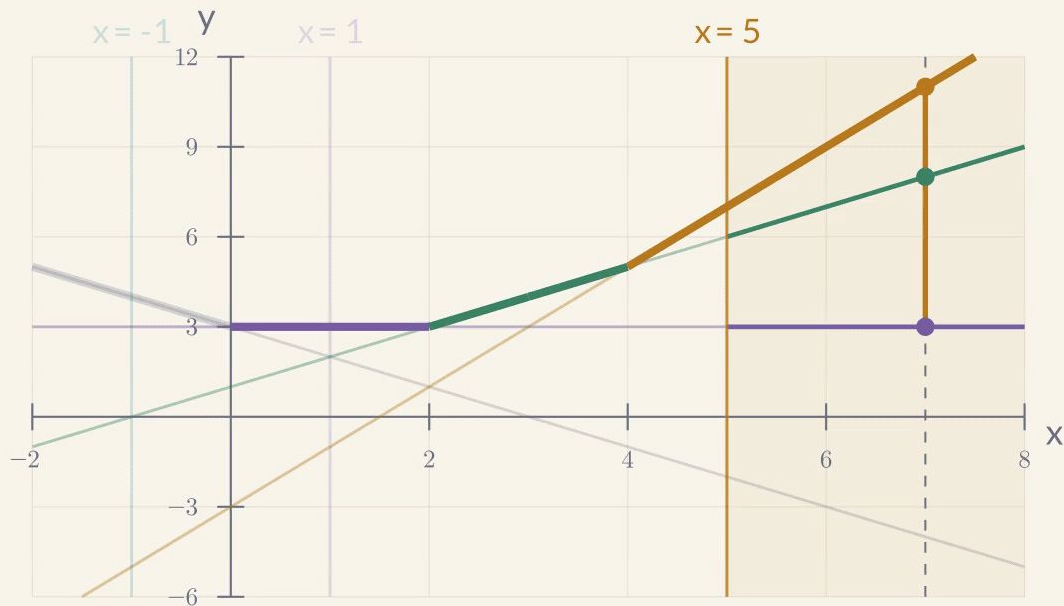
$$f_2(x) = x + 1$$

$$f_3(x) = 2x - 3$$

back

At 5, yellow gives 7: above purple's 3 and green's 6.

# Querying



$$f_1(x) = 3$$

front

$$f_2(x) = x + 1$$

$$f_3(x) = 2x - 3$$

back

For every  $x \geq 5$

$$f_3(x) - f_1(x) = 2x - 6 \geq 4$$

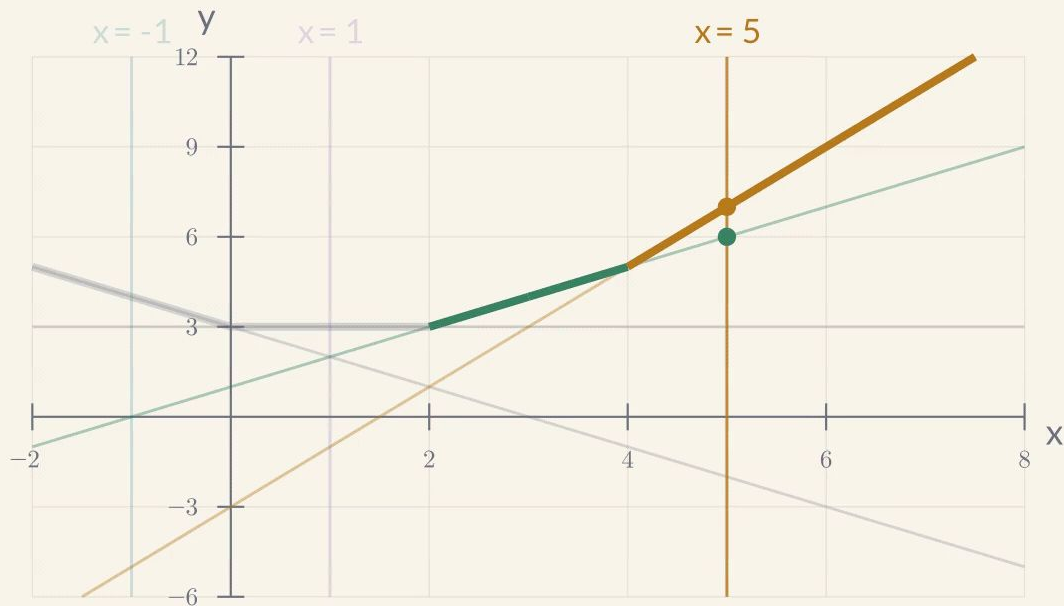
$$f_3(x) - f_2(x) = x - 4 \geq 1$$

Yellow stays above both.

Both gaps grow to the right.

After 5, yellow stays above purple and green at every point.

# Querying



Pop the front, one line at a time

$$f_1(5) = 3 < f_2(5) = 6$$

Pop purple first.

Then check the new front.

$$f_2(x) = x + 1$$

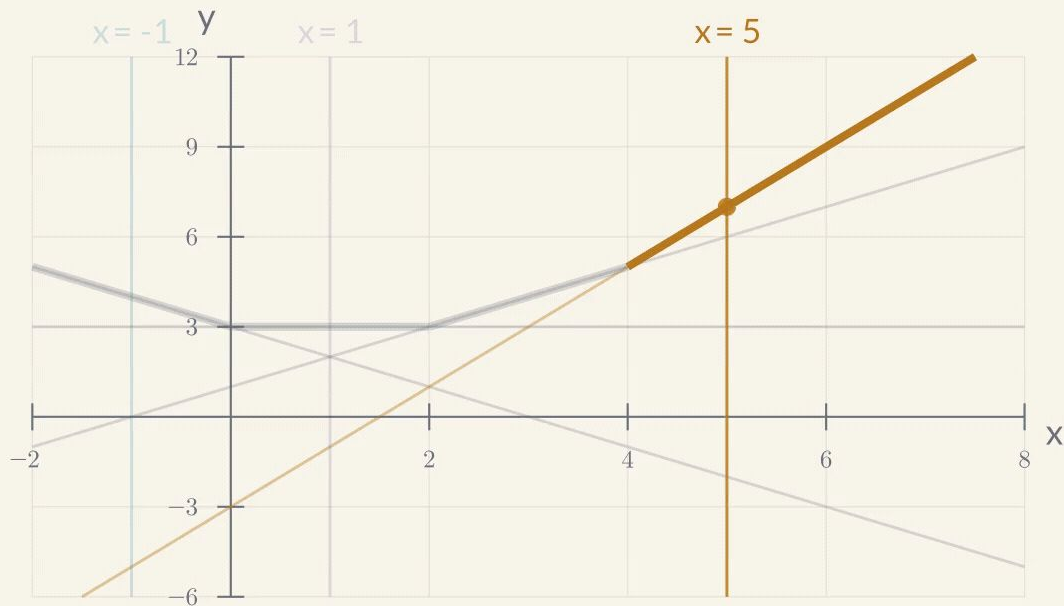
front

$$f_3(x) = 2x - 3$$

back

The deque checks neighbors: green beats purple, so pop purple.

# Querying



$$f_3(x) = 2x - 3$$

front

Pop green; return 7

$$f_2(5) = 6 < f_3(5) = 7$$

Yellow is the new front.

$$\max_i f_i(5) = f_3(5) = 7$$

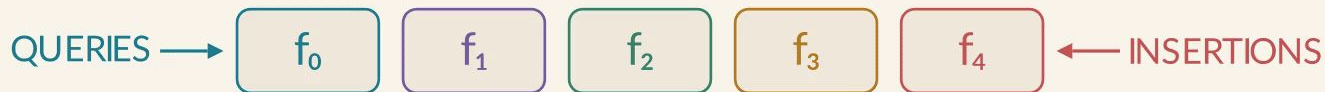
Yellow beats green. Pop green and return 7.

# Turning this into (pseudo)code

---

# Turning this into (pseudo)code

---



`front()`   `back()`   `pop_front()`   `pop_back()`   `push_back()`

$O(1)$  per deque operation

```
std::deque<Line>
```

# Turning this into (pseudo)code

---

QUERIES →



← INSERTIONS

`front()`   `back()`   `pop_front()`   `pop_back()`   `push_back()`

one call:  $O(n)$  in the worst case



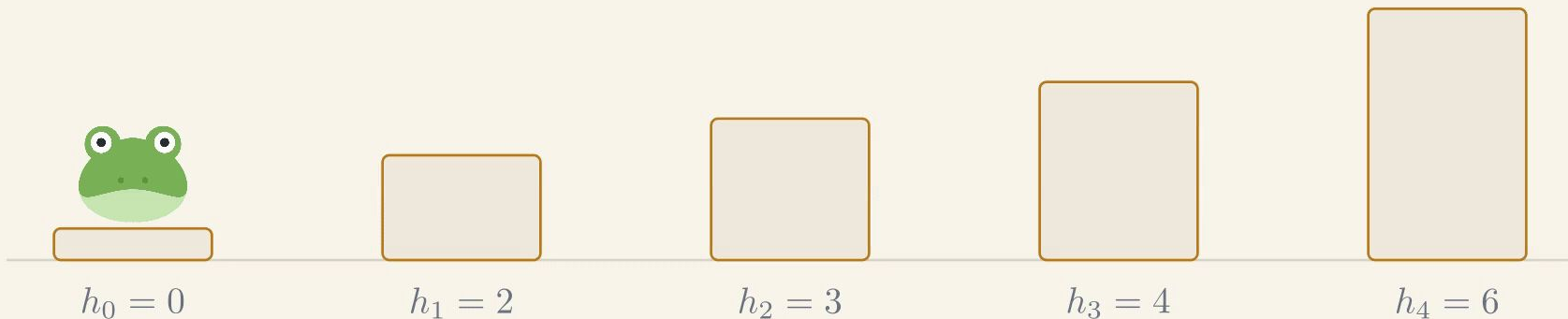


# Frog

---

Increasing heights. Start at height 0.

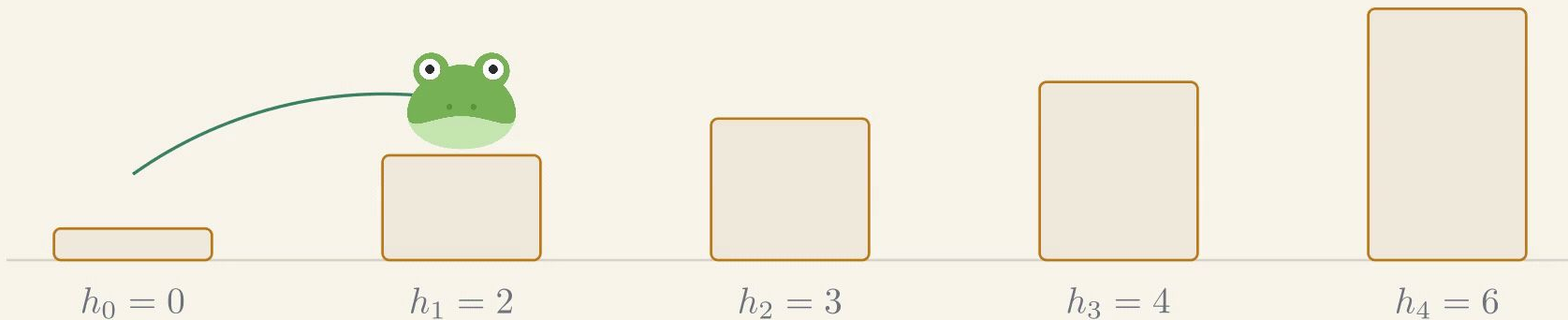
Reach the last pillar using the least total energy.



# Frog

jump from  $i$  to  $j$  :  $(h_j - h_i)^2 + C$

A larger height difference costs more;  $C$  is paid for every jump.

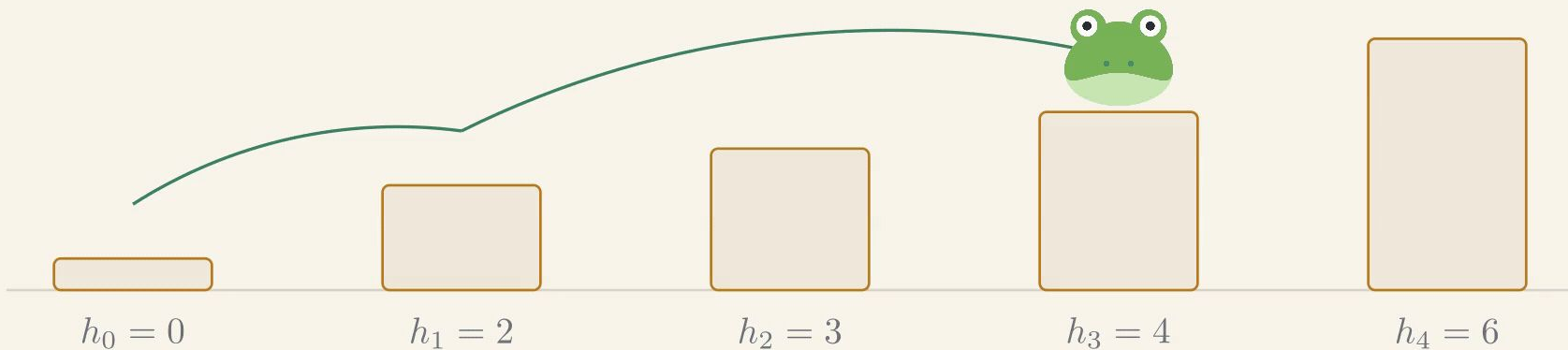


$$C = 4 : \quad (2 - 0)^2 + 4 = 8$$

# Frog

$$\text{heights } 0 \rightarrow 4 : \quad (4 - 0)^2 + 4 = 20$$

$$\text{heights } 0 \rightarrow 2 \rightarrow 4 : \quad (2 - 0)^2 + 4 + (4 - 2)^2 + 4 = 8 + 8 = 16$$



# Frog

---

$\text{energy}[j] = \text{least total energy to reach pillar } j$

$\text{energy}[0] = 0$

Compute the pillars from left to right.

$$\underbrace{\text{energy}[i]}_{\text{best route to } i} + \underbrace{(h_j - h_i)^2 + C}_{\text{one last jump to } j}$$

# Frog

---

$$\text{energy}[j] = \min_{0 \leq i < j} ((h_j - h_i)^2 + C + \text{energy}[i])$$

# Frog

```
vector<long long> energy(n, LLONG_MAX);
energy[0] = 0;
for (int j = 1; j < n; ++j) {
    for (int i = 0; i < j; ++i) {
        long long delta = height[j] - height[i];
        energy[j] = min(energy[j], energy[i] + delta * delta + C);
    }
}
```

$$1 + 2 + \cdots + (n - 1) = \frac{n(n - 1)}{2} = O(n^2)$$

# Frog

---

# Frog

---

How can the convex hull trick help?

$$\text{energy}[j] = \min_{i < j} ((h_j - h_i)^2 + C + \text{energy}[i])$$

This doesn't look like a line. (YET)

# Frog

---

Expand the square

$$\text{energy}[j] = \min_{i < j} (\textcolor{teal}{h_i^2} + \textcolor{brown}{h_j^2} - \textcolor{purple}{2h_i h_j} + \textcolor{brown}{C} + \textcolor{teal}{\text{energy}[i]})$$

$$(h_j - h_i)^2 = h_i^2 + h_j^2 - 2h_i h_j$$

# Frog

---

$$\text{energy}[j] = \min_{i < j} (\textcolor{teal}{h_i^2} + \boxed{\textcolor{brown}{h_j^2}} - 2\textcolor{purple}{h_i h_j} + \boxed{\textcolor{brown}{C}} + \textcolor{teal}{\text{energy}[i]})$$

$h_j^2$  and  $C$  do not depend on which  $i$  we choose.

# Frog

---

$$\text{energy}[j] = \min_{i < j} (h_i^2 - 2h_i h_j + \text{energy}[i]) + h_j^2 + C$$

$h_j^2$  and  $C$  do not depend on which  $i$  we choose.

# Frog

---

Now fix one earlier pillar  $i$

$$\text{energy}[j] = \min_{i < j} (h_i^2 - 2h_i h_j + \text{energy}[i]) + h_j^2 + C$$

$h_i^2$  and  $\text{energy}[i]$

Both are already known constants.

# Frog

---

The mixed term is slope  $\times$  input

$$-2h_i h_j = \underbrace{(-2h_i)}_{\text{slope}} \underbrace{h_j}_{\text{input}}$$

$$m_i = -2h_i$$

$$x = h_j$$

# Frog

---

The constant terms become the y-intercept

$$f_i(x) = (-2h_i)x + \underbrace{(\text{energy}[i] + h_i^2)}_{\text{y-intercept}}$$

$$b_i = \text{energy}[i] + h_i^2$$

# Frog

---

Evaluate that line at the destination height

$$f_i(h_j) = (-2h_i)h_j + (\text{energy}[i] + h_i^2)$$

$$m_i = -2h_i \quad x = h_j \quad b_i = \text{energy}[i] + h_i^2$$

Each earlier pillar contributes one line.

# Frog

---

Constructing the minimum hull

# Frog

Constructing the minimum hull

$$f_i(x) = -2h_i x + (\text{energy}[i] + h_i^2)$$

Insert on the LEFT

$$m_i = -2h_i$$

Heights increase; new slopes decrease.

Repair the left end of the hull.

Add the new line on the left.

Query from the RIGHT

$$x = h_j$$

Query heights increase.

Compare the rightmost neighbours.

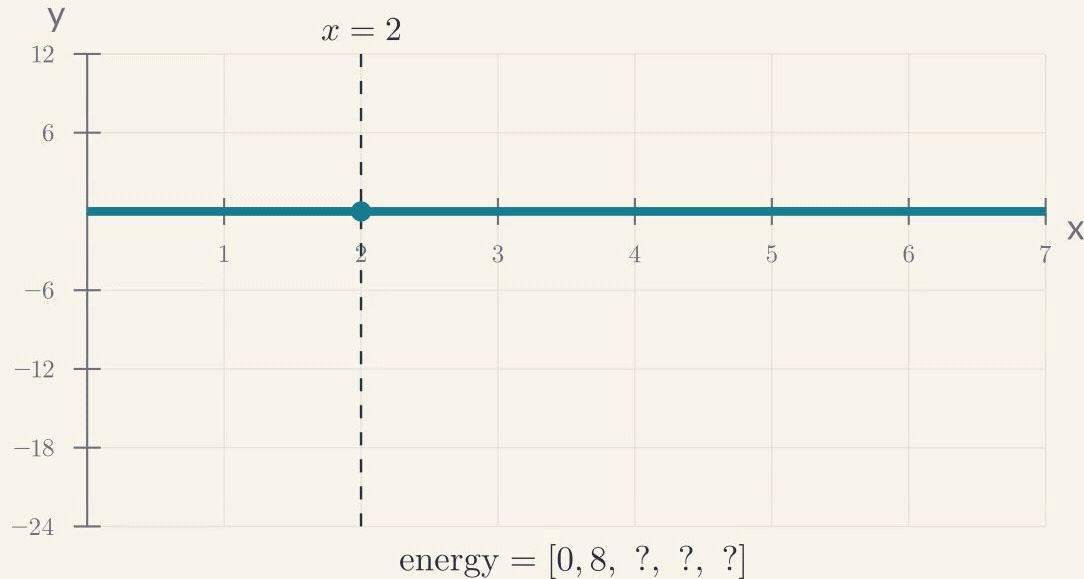
Pop right if the neighbour is lower or tied.

$$\text{energy}[j] = \min_{i < j} f_i(h_j) + h_j^2 + C$$

Keep the lowest useful lines. Store their slopes increasing from left to right.

# Frog

Constructing the minimum hull



1. Query at height 2

$$\min f_i(2) = f_0(2) = 0$$

$$\text{energy}[1] = 0 + 2^2 + 4 = 8$$

Now the next line is known.

$$f_1(x) = -4x + (8 + 2^2) = -4x + 12$$

$$f_0(x) = 0$$

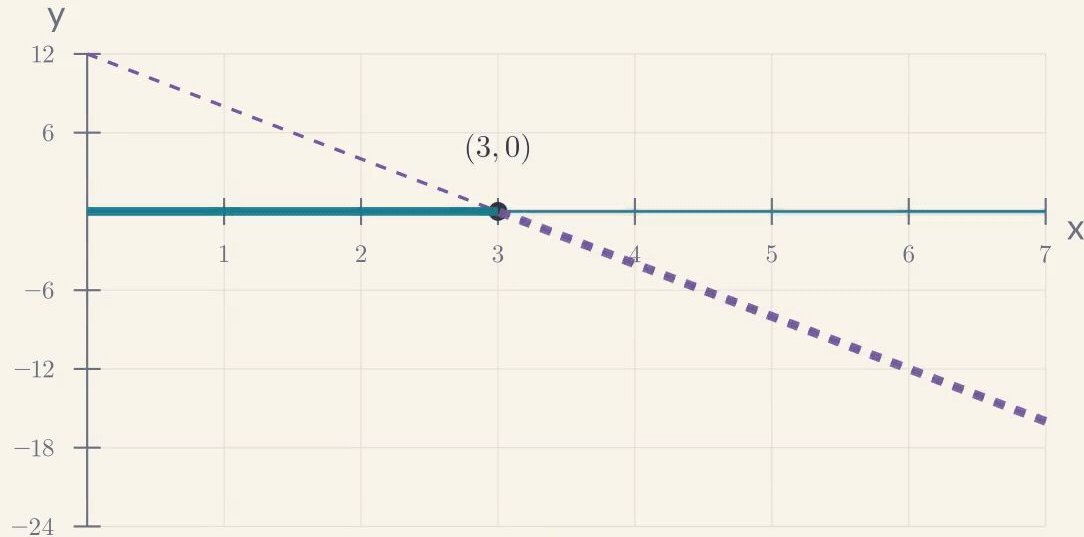
LEFT • insert / repair

slopes increase →

RIGHT • query / pop

## Frog

Constructing the minimum hull

energy =  $[0, 8, ?, ?, ?]$ 

new line

$$f_1(x) = -4x + 12$$

LEFT • insert / repair

$$f_0(x) = 0$$

slopes increase  $\rightarrow$ 

2. Insert purple on the LEFT

$$f_1(x) = -4x + 12$$

Blue is lower before  $x = 3$ .Purple is lower after  $x = 3$ .

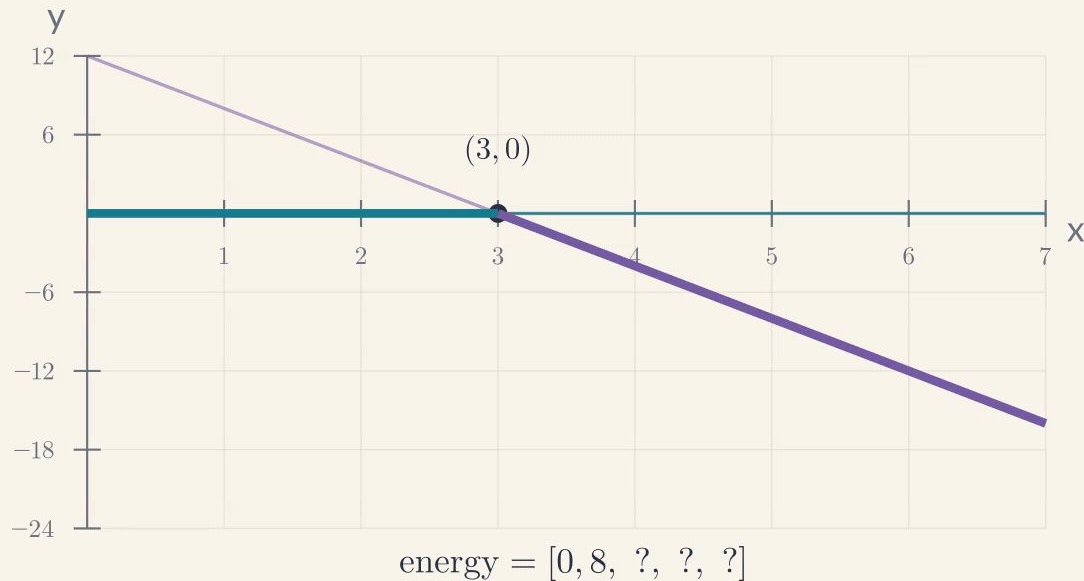
Keep both. Add purple on the left.

$$-4 < 0$$

RIGHT • query / pop

## Frog

Constructing the minimum hull



2. Insert purple on the LEFT

$$f_1(x) = -4x + 12$$

Blue is lower before  $x = 3$ .Purple is lower after  $x = 3$ .

Keep both. Add purple on the left.

$$-4 < 0$$

$$f_1(x) = -4x + 12$$

$$f_0(x) = 0$$

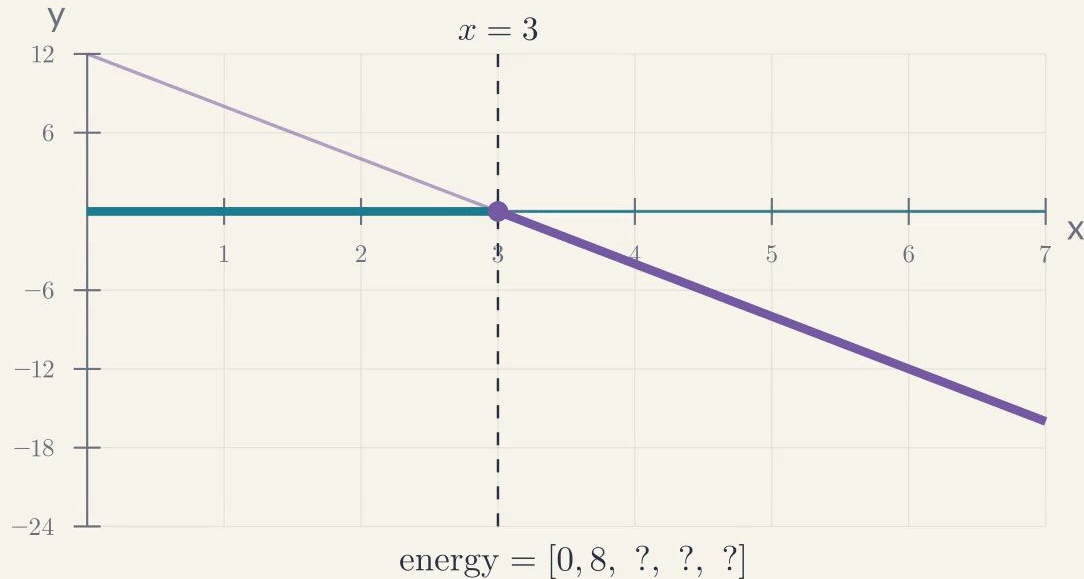
LEFT • insert / repair

slopes increase →

RIGHT • query / pop

## Frog

Constructing the minimum hull



$$f_1(x) = -4x + 12$$

$$f_0(x) = 0$$

LEFT • insert / repair

slopes increase →

RIGHT • query / pop

3. Query at height 3

$$f_1(3) = f_0(3) = 0$$

Purple has the smaller slope.

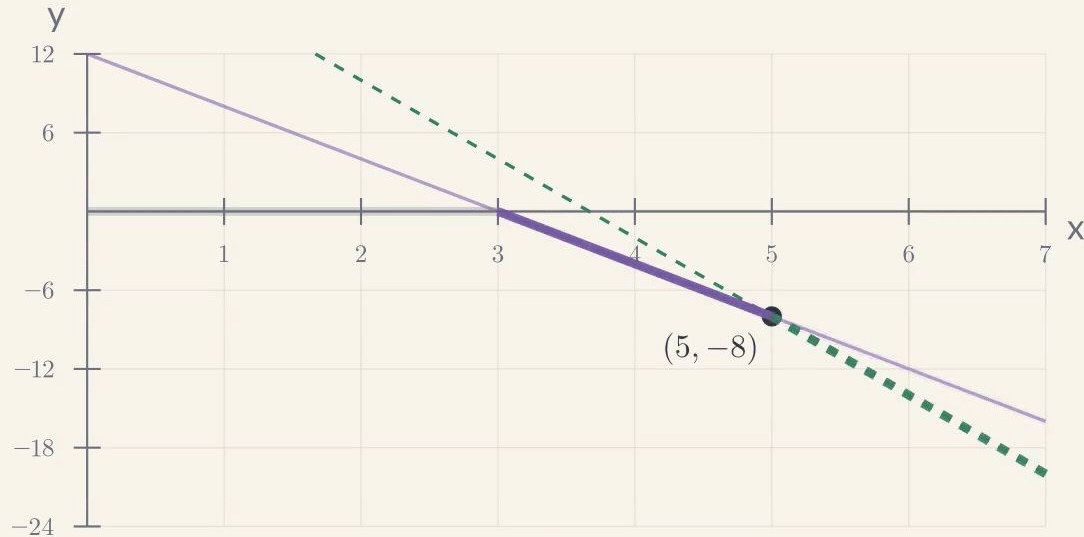
For  $x > 3$ , purple stays lower.

Pop blue from the RIGHT.

$$\text{energy}[2] = 0 + 3^2 + 4 = 13$$

## Frog

Constructing the minimum hull



energy = [0, 8, 13, ?, ?]

new line

$$f_2(x) = -6x + 22$$

LEFT • insert / repair

$$f_1(x) = -4x + 12$$

slopes increase →

4. Insert green on the LEFT

$$f_2(x) = -6x + (13 + 3^2)$$

$$f_2(x) = -6x + 22$$

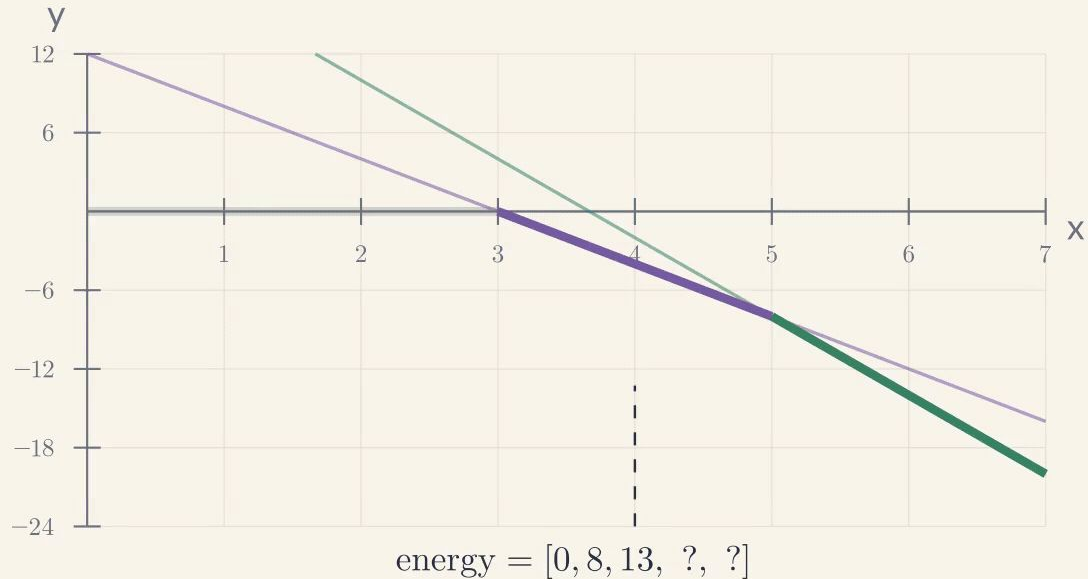
Purple is lower until  $x = 5$ .Green is lower after  $x = 5$ .

$$-6 < -4 \quad (\text{keep both})$$

RIGHT • query / pop

## Frog

Constructing the minimum hull



5. Query at height 4

$$f_1(4) = -4 < -2 = f_2(4)$$

The rightmost line is lower.

$$\min f_i(4) = -4$$

$$\text{energy}[3] = -4 + 4^2 + 4 = 16$$

Green can win later: keep it.

$$f_2(x) = -6x + 22$$

$$f_1(x) = -4x + 12$$

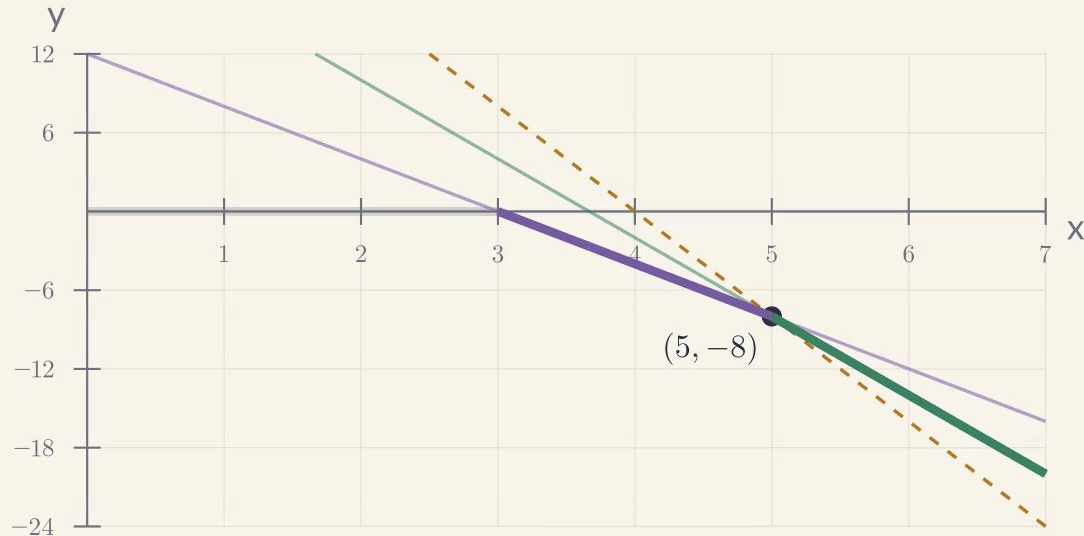
LEFT • insert / repair

slopes increase →

RIGHT • query / pop

## Frog

Constructing the minimum hull



6. Try inserting yellow

$$f_3(x) = -8x + (16 + 4^2)$$

$$f_3(x) = -8x + 32$$

All three meet at  $(5, -8)$ .

Does green still need to be kept?

new line

energy =  $[0, 8, 13, 16, ?]$ 

$$f_3(x) = -8x + 32$$

$$f_2(x) = -6x + 22$$

$$f_1(x) = -4x + 12$$

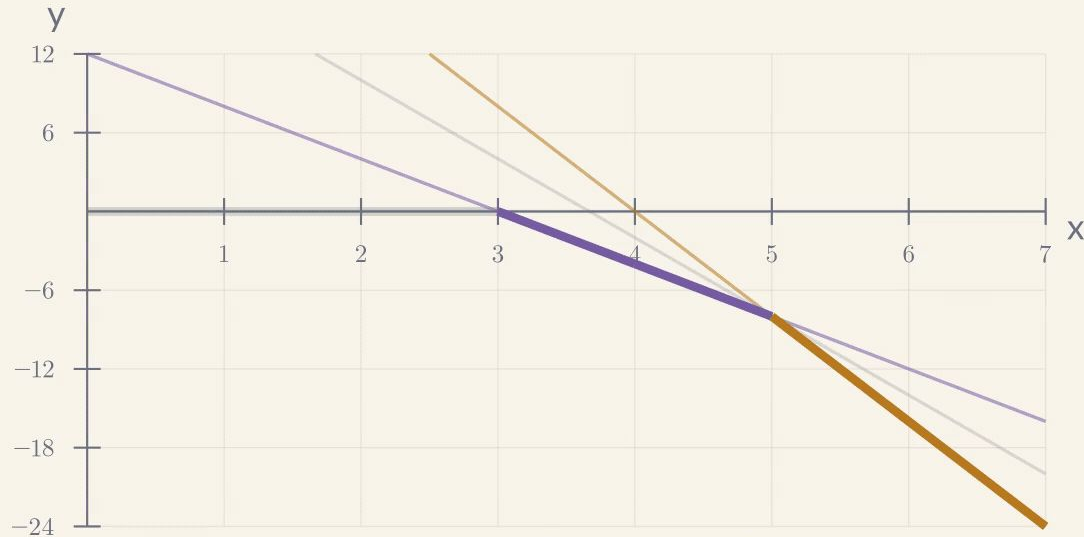
LEFT • insert / repair

slopes increase  $\rightarrow$ 

RIGHT • query / pop

## Frog

Constructing the minimum hull



energy = [0, 8, 13, 16, ?]

$$f_3(x) = -8x + 32$$

$$f_1(x) = -4x + 12$$

Repair the LEFT, then insert  
 Remove green from the LEFT.  
 Insert yellow on the LEFT.  
 Purple and yellow remain useful.  
 $-8 < -4$  (slope order)

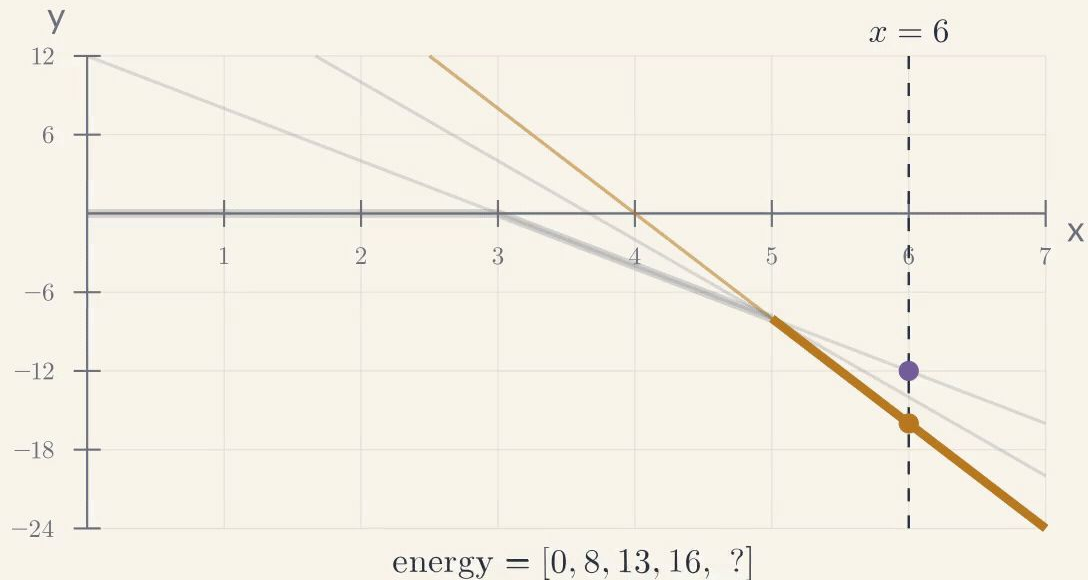
LEFT • insert / repair

slopes increase →

RIGHT • query / pop

## Frog

Constructing the minimum hull



$$f_3(x) = -8x + 32$$

LEFT • insert / repair

slopes increase →

RIGHT • query / pop

7. Query at height 6

$$f_3(6) = -16 < -12 = f_1(6)$$

The left neighbour is lower.

Pop purple from the RIGHT.

$$\min f_i(6) = f_3(6) = -16$$

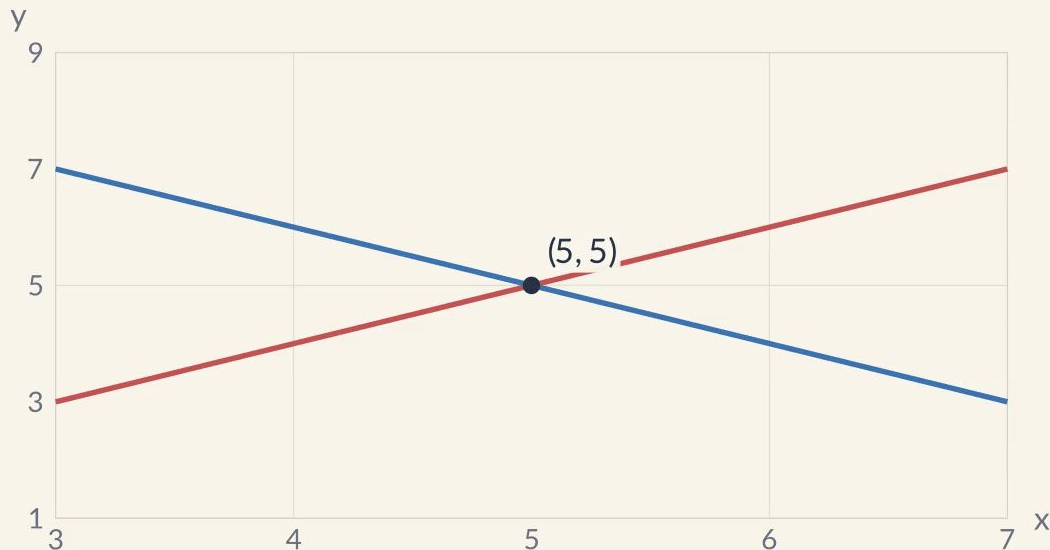
$$\text{energy}[4] = -16 + 6^2 + 4 = 24$$

# Generalized single crossing

---

# Generalized single crossing

Start at the intersection: recall single crossing



Blue has the smaller slope.  
It is higher before the crossing.  
Red has the larger slope.  
It is higher after the crossing.

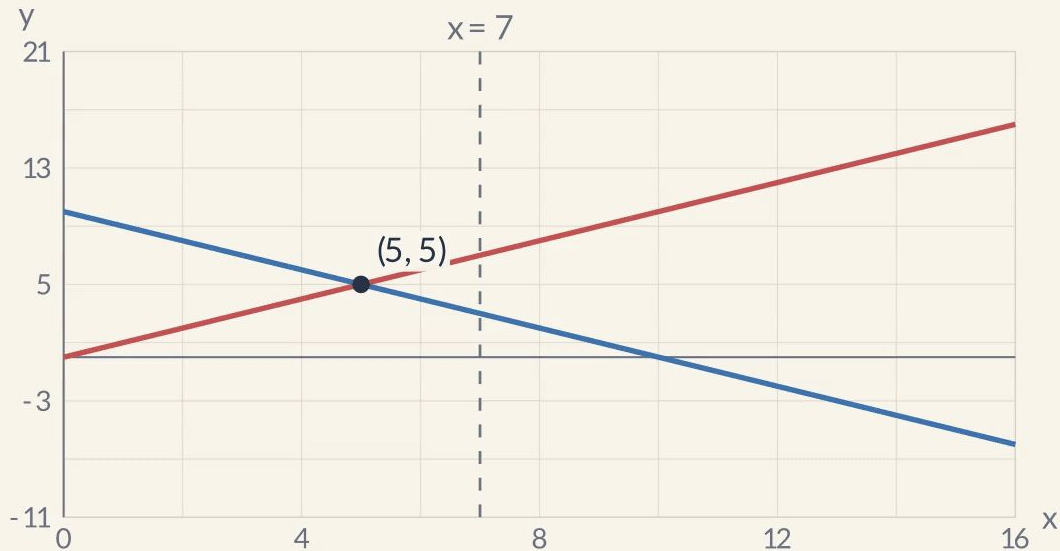
$$f(x) = x$$

$$g(x) = -x + 10$$

Before the crossing: smaller slope. After it: larger slope.

# Generalized single crossing

Zoom out, then draw a vertical line



Choose any vertical line

$$x = 7$$

Its position is arbitrary.

Where did the two lines cross?

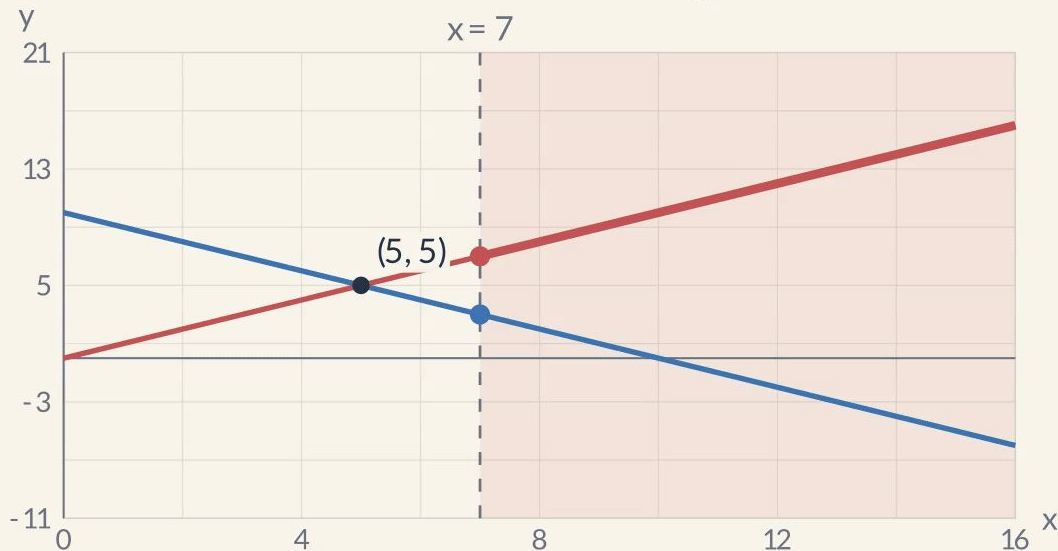
$$f(x) = x$$

$$g(x) = -x + 10$$

Draw a vertical line anywhere: here, at  $x = 7$ .

# Generalized single crossing

The crossing is to the LEFT of the vertical line



Red is on top at the divider

$$f(7) = 7 > 3 = g(7)$$

The crossing already happened.

Red stays above blue on the RIGHT.

Blue cannot improve the maximum on this side: it is useless here.

Blue can still matter on the left.

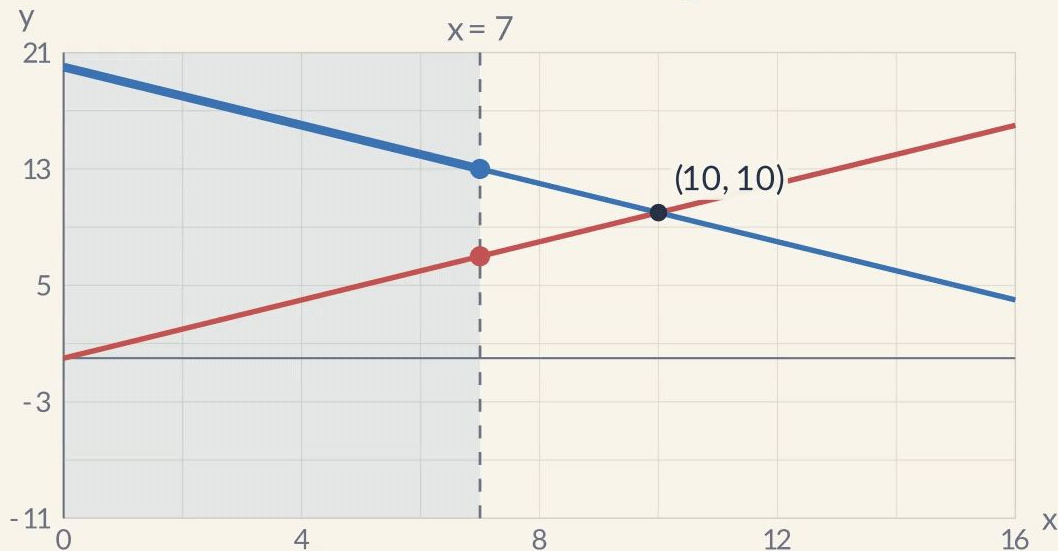
$$f(x) = x$$

$$g(x) = -x + 10$$

Red wins at the divider and makes blue useless on the right.

# Generalized single crossing

The crossing is to the RIGHT of the vertical line



Blue is on top at the divider

$$g(7) = 13 > 7 = f(7)$$

The crossing has not happened.

Blue stays above red on the LEFT.

Red cannot improve the maximum on this side: it is useless here.

Red can still matter on the right.

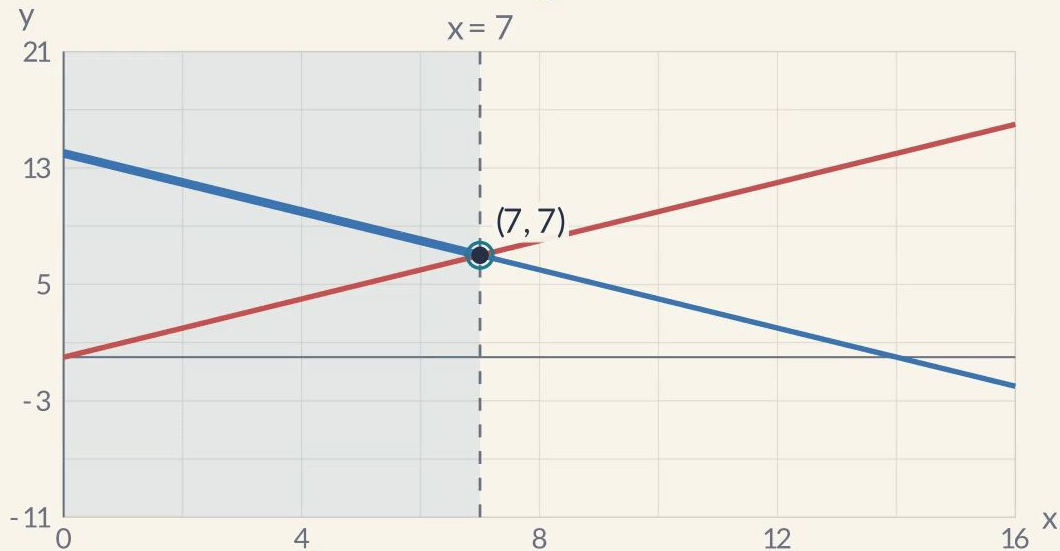
$$f(x) = x$$

$$g(x) = -x + 20$$

Blue wins at the divider and makes red useless on the left.

# Generalized single crossing

A crossing ON the vertical line belongs to the right case



On the vertical line, the lines tie  
 $f(7) = g(7) = 7$

Either line gives the value at 7.

Blue is higher on the LEFT:

red is useless on that side.

Red can still matter on the RIGHT.

$$f(x) = x$$

$$g(x) = -x + 14$$

A tie at the divider: blue rules red out on the left, but not on the right.

# Generalized single crossing

---

Idea: What ranges might a line still be useful on?

Potentially useful on a range

Our single-crossing argument has not ruled the line out as a possible maximum somewhere in that range.

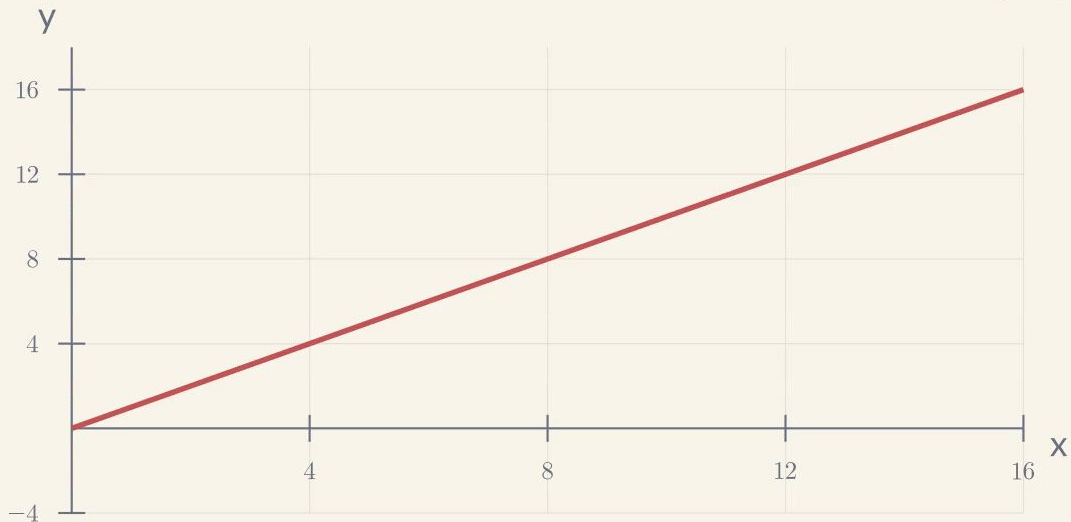
This argument is kind of confusing.

We aren't claiming a line is the max value on a range.

We're claiming it's not impossible for a line to be the max on a range.

# What ranges matter?

Start with the range  $[0, 16)$



Insert red

$$R(x) = x$$

No line is saved for this range.

Keep red for  $[0, 16)$ .

Integer queries:  $0, 1, \dots, 15$

$$R(x) = x$$

$$B(x) = -2x + 15$$

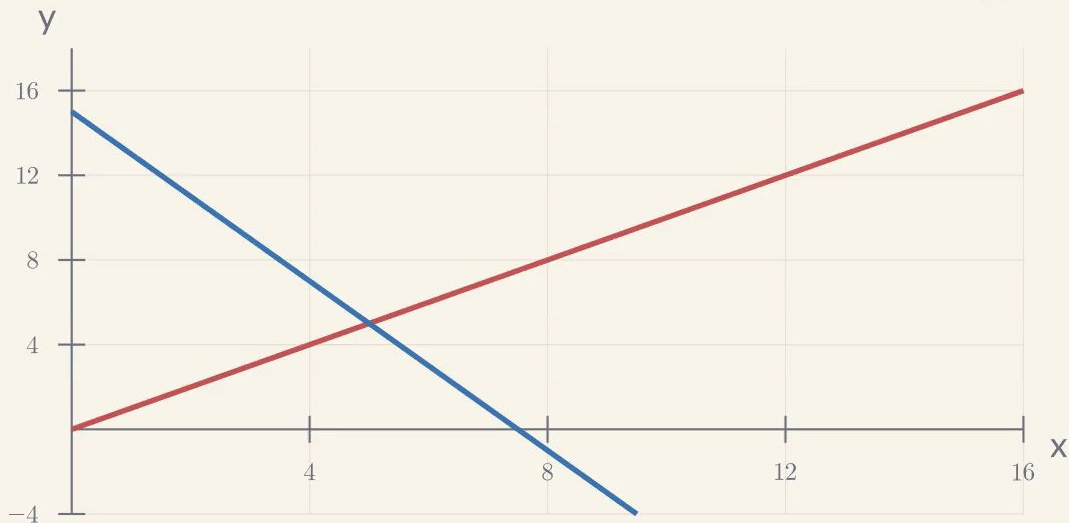
$$Y(x) = 10$$

$$W(x) = -x + 14$$

Start with red. It is our candidate for the whole range.

# What ranges matter?

Insert blue alongside red



Draw the new line

$$B(x) = -2x + 15$$

Keep red visible.

Compare these two lines.

$$R(x) = x$$

$$B(x) = -2x + 15$$

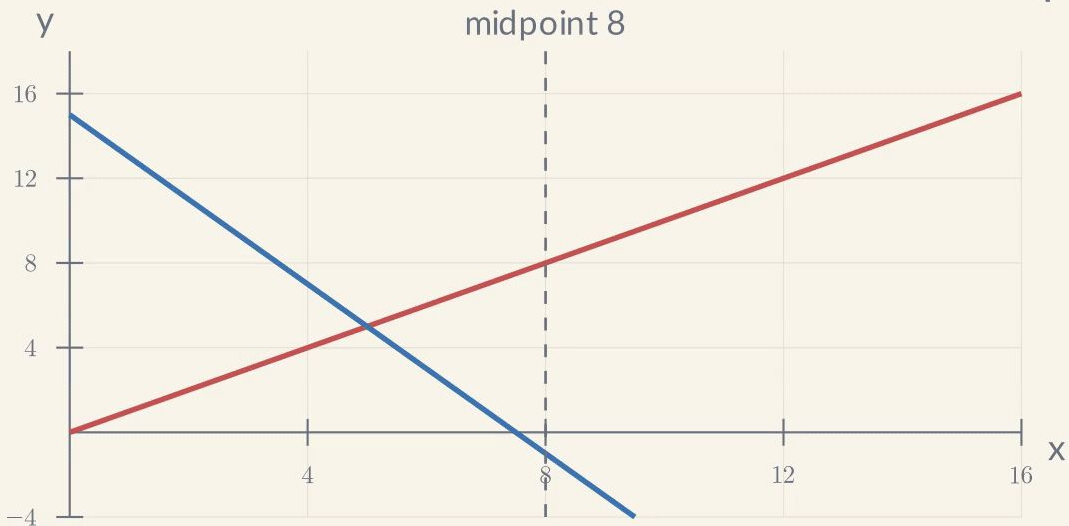
$$Y(x) = 10$$

$$W(x) = -x + 14$$

First draw blue alongside red.

# What ranges matter?

Now draw the midpoint



What does this divider tell us?

$$\frac{0 + 16}{2} = 8$$

Given this vertical line,  
what information do we know?

$$R(x) = x$$

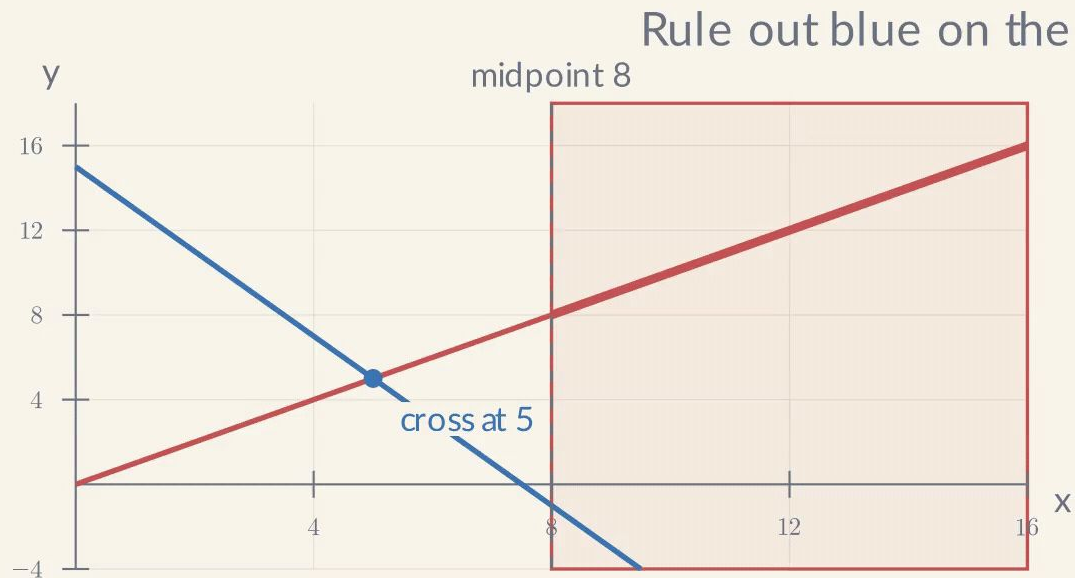
$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

Given this vertical line, what information do we know?

# What ranges matter?



Red always wins on  $[8, 16)$

$$R(8) = 8 > B(8) = -1$$

They already crossed at  $x = 5$ .

Red has the larger slope.

Blue cannot win on  $[8, 16)$ .

$$R(x) = x$$

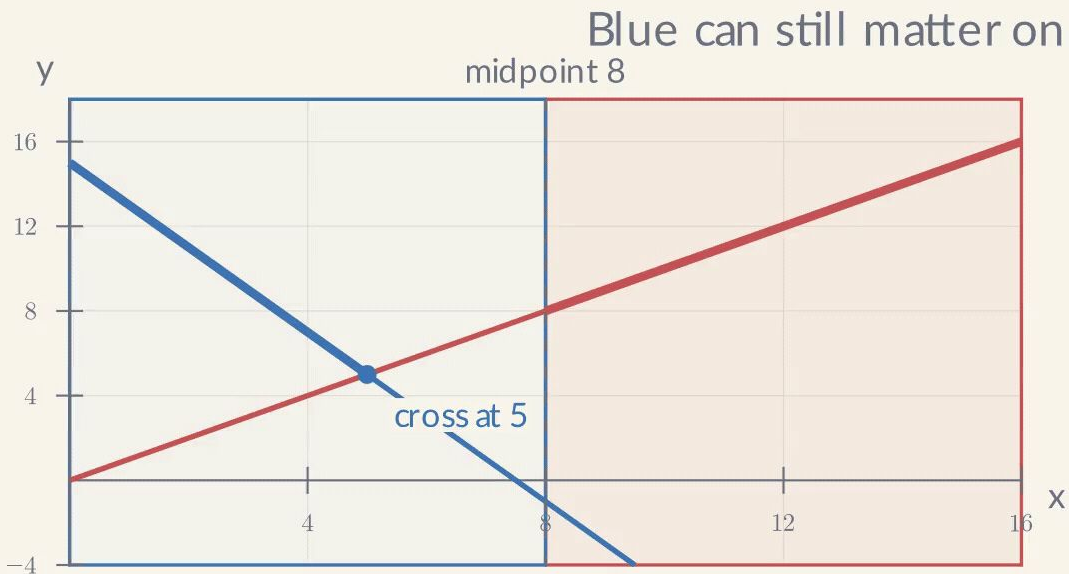
$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

On  $[8, 16)$ , red is always above blue.

# What ranges matter?



Keep blue for  $[0, 8)$

Blue wins here on  $[0, 5)$ .

$$B(2) = 11 > R(2) = 2$$

Red wins for  $5 < x < 8$ .

Blue is still a possible maximum  
somewhere in  $[0, 8)$ .

$$R(x) = x$$

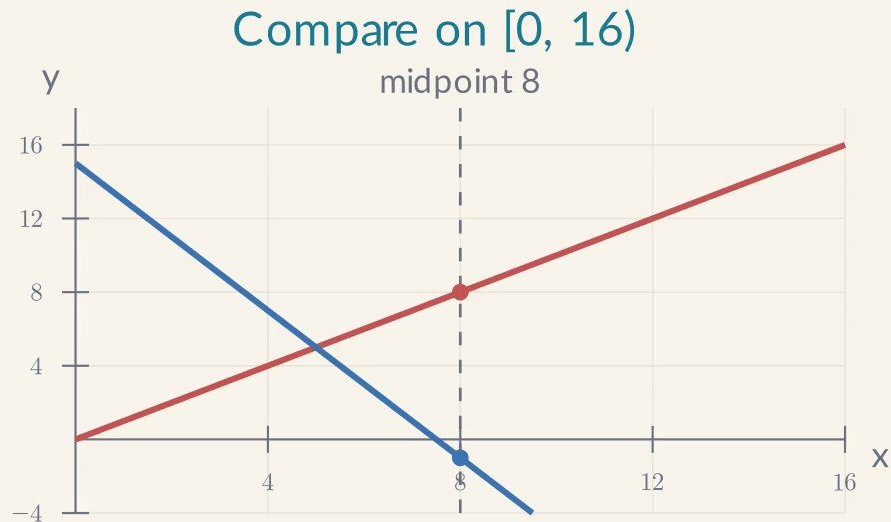
$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

Blue can still be the maximum somewhere in  $[0, 8)$ .

# What ranges matter?



$$R(8) = 8 > B(8) = -1$$

Keep red at this node.

$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

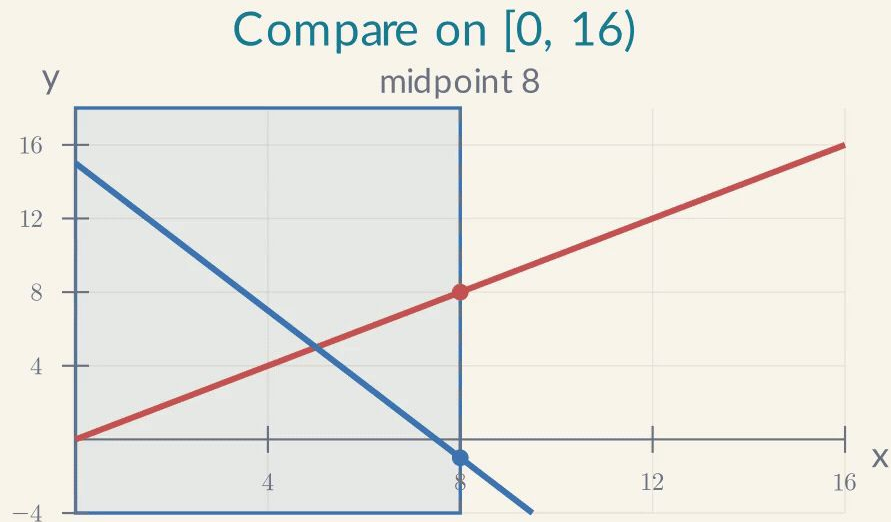
Possible lines

$[0, 16)$  mid 8

$$R(8) = 8$$

Organize possible lines by range. Red stays at the root.

# What ranges matter?



## Possible lines

$[0, 16)$  mid 8  
 $R(8) = 8$

$[0, 8)$  mid 4  
 $B(4) = 7$

$R(x) = x$

$B(x) = -2x + 15$

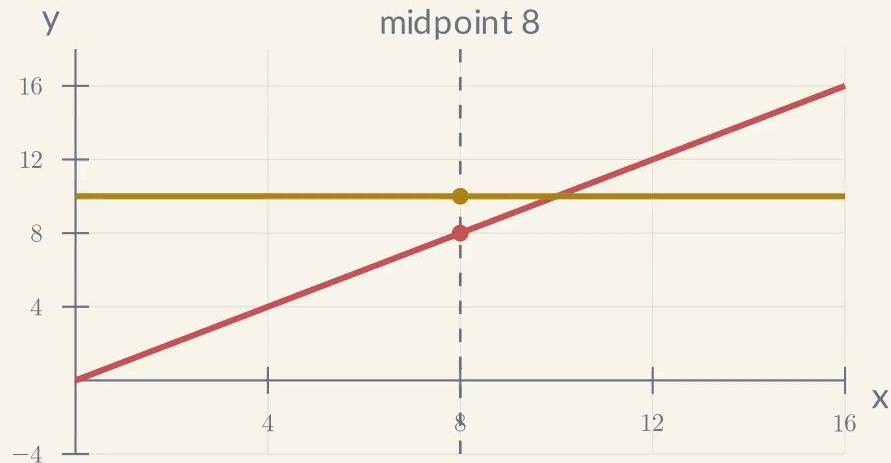
$Y(x) = 10$

$W(x) = -x + 14$

Blue may still win on the left: create the child  $[0, 8)$ .

# What ranges matter?

Insert yellow on  $[0, 16)$



$$Y(8) = 10 > R(8) = 8$$

Yellow takes the root's saved slot.

$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

Possible lines

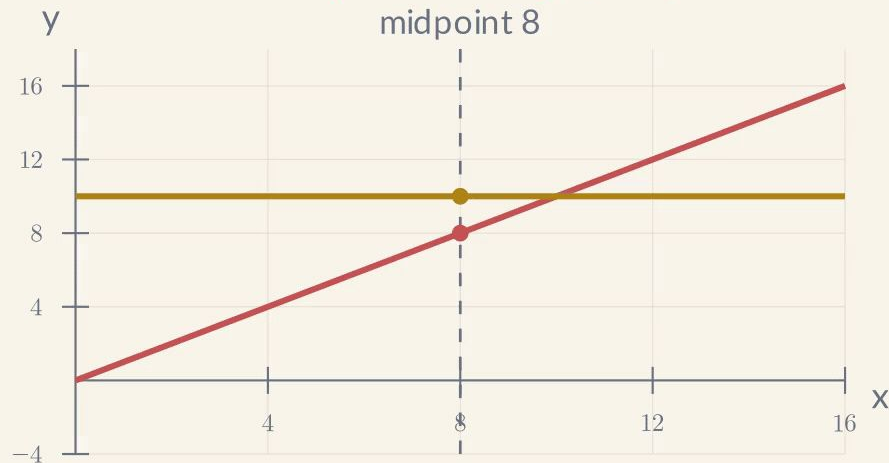
$$[0, 16) \text{ mid } 8 \\ R(8) = 8$$

$$[0, 8) \text{ mid } 4 \\ B(4) = 7$$

Insert yellow. It wins at 8, so yellow replaces red at the root.

# What ranges matter?

Insert yellow on  $[0, 16)$



$$Y(8) = 10 > R(8) = 8$$

Yellow takes the root's saved slot.

$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

Possible lines

$$[0, 16) \text{ mid } 8$$

$$Y(8) = 10$$

R

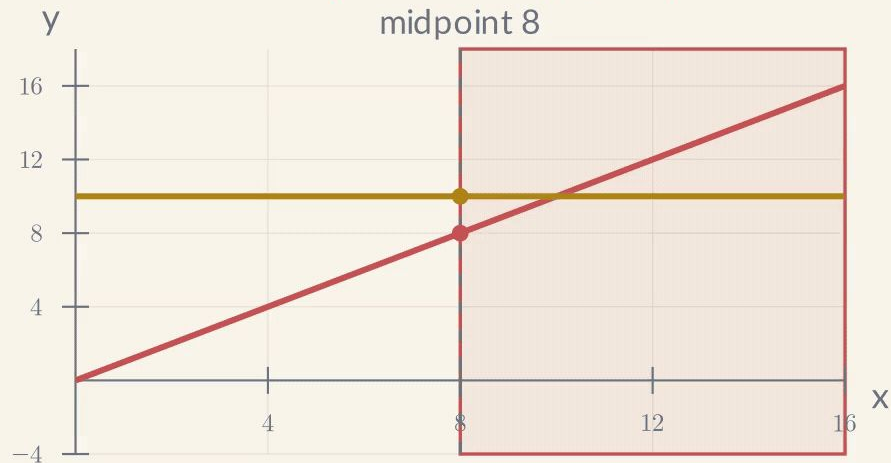
$$[0, 8) \text{ mid } 4$$

$$B(4) = 7$$

Insert yellow. It wins at 8, so yellow replaces red at the root.

# What ranges matter?

Insert yellow on  $[0, 16)$



Possible lines

$$[0, 16) \text{ mid } 8 \\ Y(8) = 10$$

$$[0, 8) \text{ mid } 4 \\ B(4) = 7$$

$$[8, 16) \text{ mid } 12 \\ R(12) = 12$$

$[0, 16) \rightarrow [8, 16)$

Send red into the right child.

$$R(x) = x$$

$$B(x) = -2x + 15$$

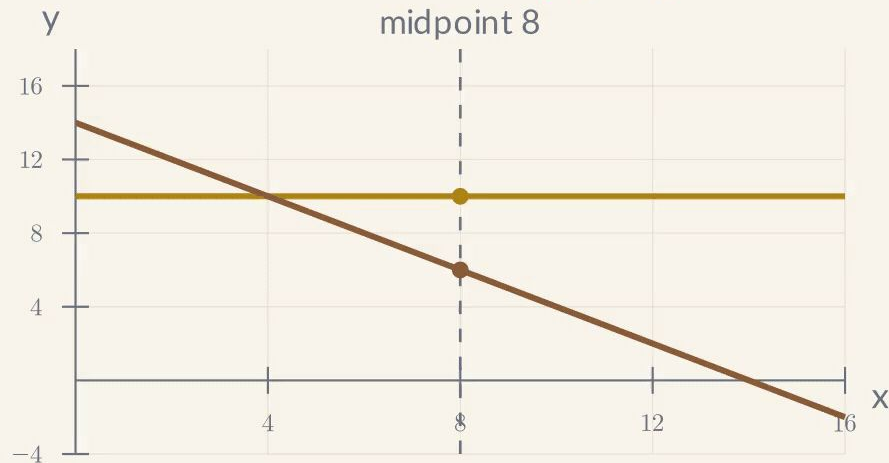
$$Y(x) = 10$$

$$W(x) = -x + 14$$

Red may still win on the right: create the child  $[8, 16)$ .

# What ranges matter?

Insert brown on  $[0, 16)$



$$Y(8) = 10 > W(8) = 6$$

Keep yellow; send brown to the left.

$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

Insert brown. Yellow stays at the root; brown goes left.

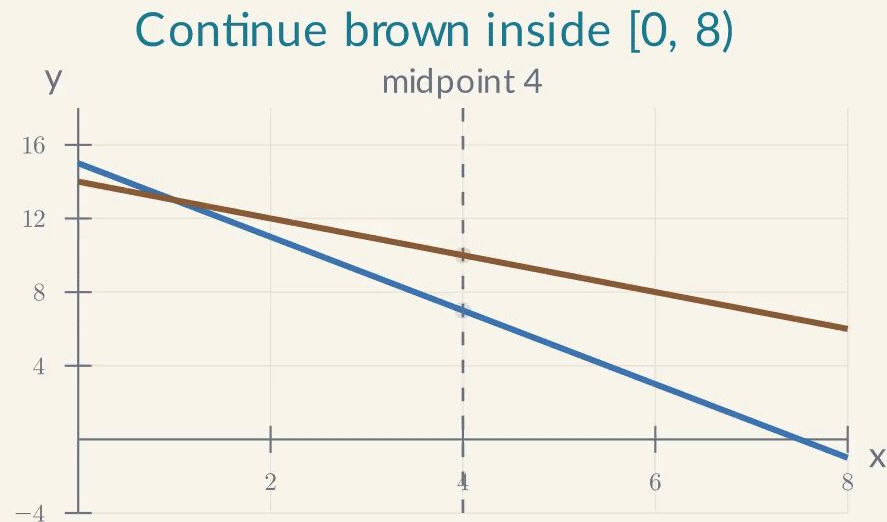
Possible lines

$$[0, 16) \text{ mid } 8 \\ Y(8) = 10$$

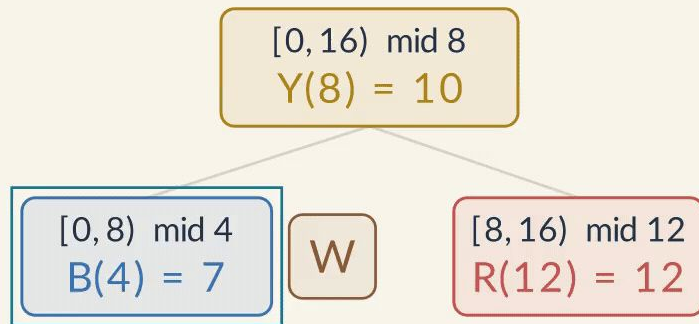
$$[0, 8) \text{ mid } 4 \\ B(4) = 7$$

$$[8, 16) \text{ mid } 12 \\ R(12) = 12$$

# What ranges matter?



## Possible lines



$$R(x) = x$$

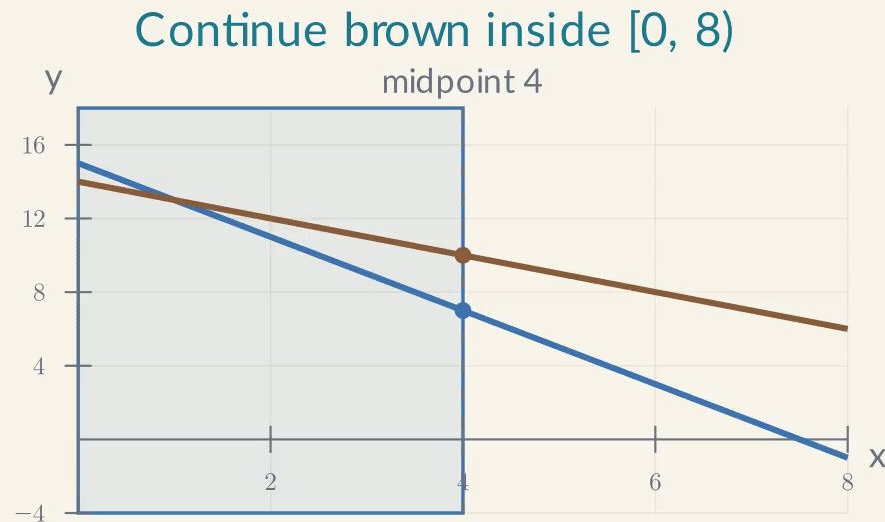
$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

Inside  $[0, 8)$ , compare brown with blue at the new midpoint, 4.

# What ranges matter?



$[0, 8) \rightarrow [0, 4)$

Send blue into the left child.

$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

## Possible lines

$$[0, 16) \text{ mid } 8$$

$$Y(8) = 10$$

$$[0, 8) \text{ mid } 4$$

$$W(4) = 10$$

$$[8, 16) \text{ mid } 12$$

$$R(12) = 12$$

$$[0, 4) \text{ mid } 2$$

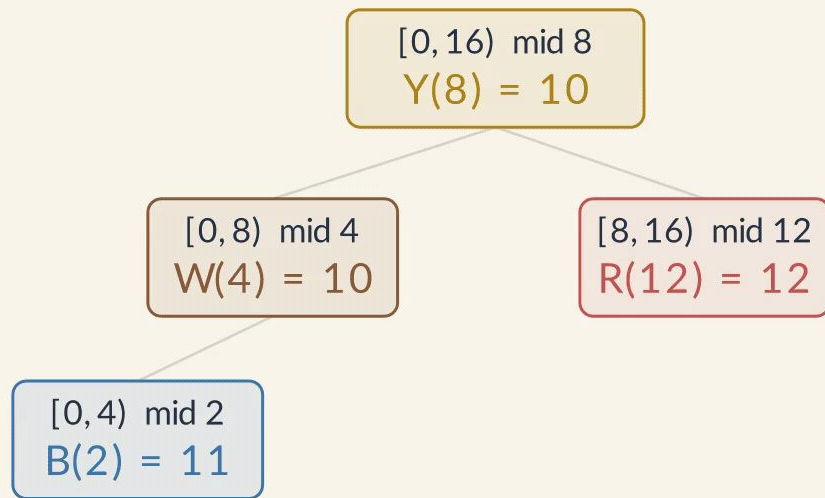
$$B(2) = 11$$

Blue may still win on  $[0, 4)$ , so store it in a new left child.

# Li Chao tree

Finally, we can define what a Li Chao tree is.

This structure is a Li Chao tree  
 Each node represents an interval  $[l, r)$ .  
 It stores one whole line function  $f(x)$ .  
 $m = \lfloor (l + r) / 2 \rfloor$   
 $[l, r) \rightarrow [l, m), [m, r)$   
 Create the children only when needed.



$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

A Li Chao tree associates one line with each stored query interval.

# Li Chao tree

## The midpoint invariant

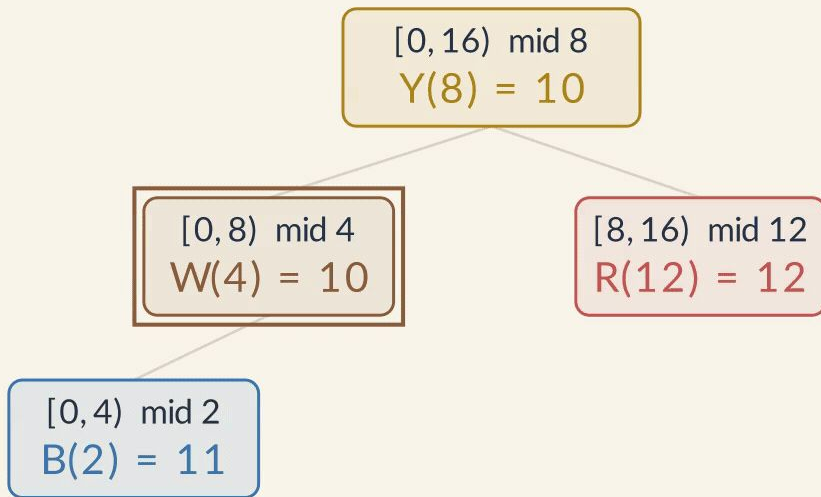
Keep the winner at this midpoint

The saved line has the largest value at  $m$   
among lines compared at this node.

For  $[0, 8)$ , the midpoint is 4:

$$W(4) = 10 > B(4) = 7$$

Keep  $W$  here;  $B$  may still matter below.



$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

Midpoint invariant: the saved line is best among lines compared at this node.

# Li Chao tree

## What the stored interval means

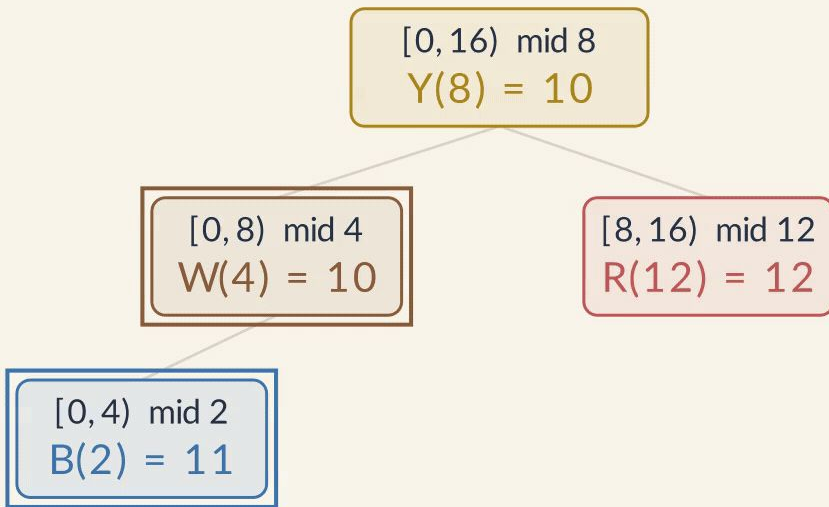
A possible line for this interval

$[0, 4)$  stores  $B(x) = -2x + 15$

Other nodes may give a larger value.

$$B(2) = 11 < W(2) = 12$$

The midpoint invariant is local to a node.



$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

An interval records where to keep a candidate line.

# Li Chao tree

## Pseudocode

```

insert(f, node [l,r]):
  if empty: store f; return
  m = (l+r) // 2
  if f(m) > g(m):
    swap(f, g)
  if r-l == 1: return
  if f(l) > g(l):
    insert(f, left [l,m])
  elif f(r-1) > g(r-1):
    insert(f, right [m,r])
  else: discard f

```

f: arriving line    g: stored line

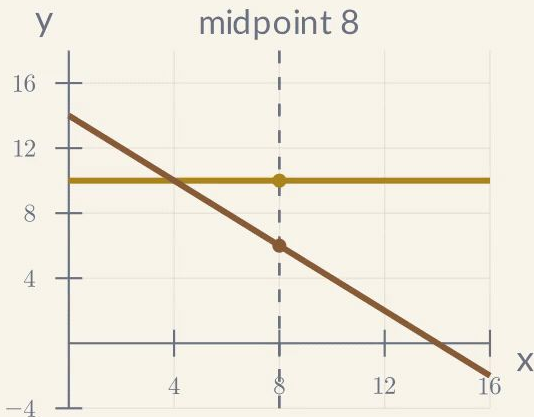
$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

$$W(x) = -x + 14$$

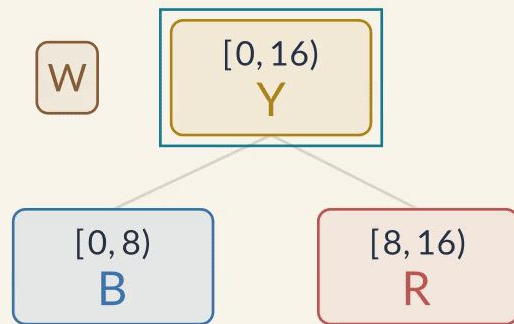
## Compare on [0, 16)



$$W(8) = 6 < Y(8) = 10$$

No swap: keep yellow.

## Li Chao tree



At midpoint 8, yellow wins. Keep yellow at the root.

# Li Chao tree

## Pseudocode

```

insert(f, node [l,r]):
    if empty: store f; return
    m = (l+r) // 2
    if f(m) > g(m):
        swap(f, g)
    if r-l == 1: return
    if f(l) > g(l):
        insert(f, left [l,m])
    elif f(r-1) > g(r-1):
        insert(f, right [m,r])
    else: discard f
  
```

f: arriving line    g: stored line

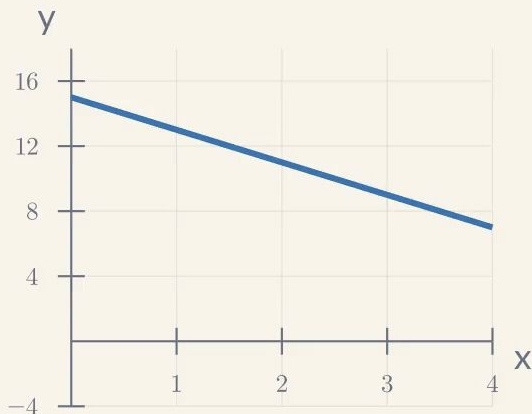
$$R(x) = x$$

$$B(x) = -2x + 15$$

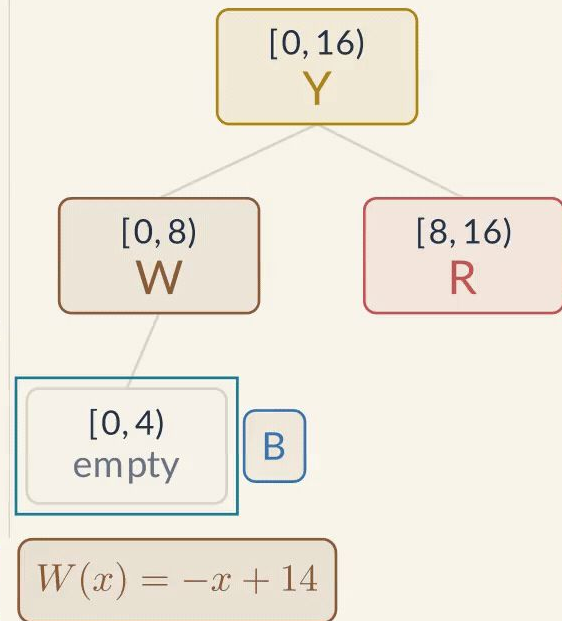
$$Y(x) = 10$$

$$W(x) = -x + 14$$

## Compare on [0, 4)



## Li Chao tree



The child is empty. Store blue and return.

# Li Chao tree

## Pseudocode

```

insert(f, node [l,r]):
  if empty: store f; return
  m = (l+r) // 2
  if f(m) > g(m):
    swap(f, g)
  if r-l == 1: return
  if f(l) > g(l):
    insert(f, left [l,m])
  elif f(r-1) > g(r-1):
    insert(f, right [m,r])
  else: discard f
  
```

f: arriving line    g: stored line

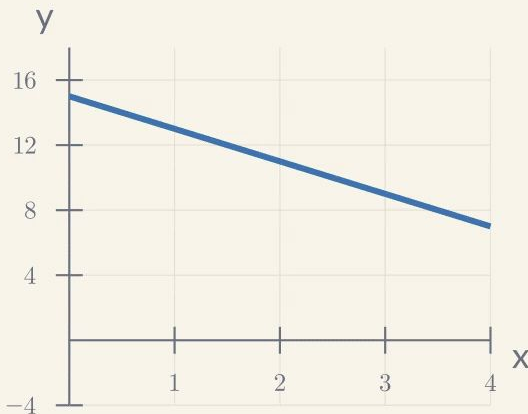
$$R(x) = x$$

$$B(x) = -2x + 15$$

$$Y(x) = 10$$

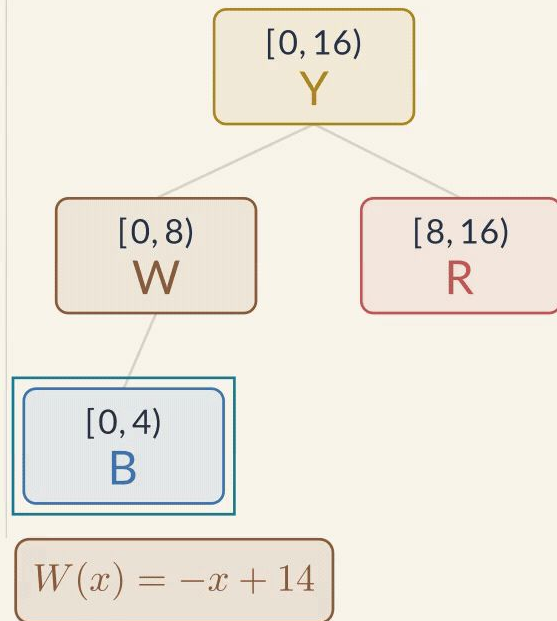
$$W(x) = -x + 14$$

## Compare on [0, 4)



Other branches  
Not taken in this insertion.

## Li Chao tree



If neither endpoint wins, discard the remaining line.

# Li Chao Queries

---

# Li Chao Queries

---

How do we answer our convex hull queries?

Suppose we want the maximum value at point  $x$ .

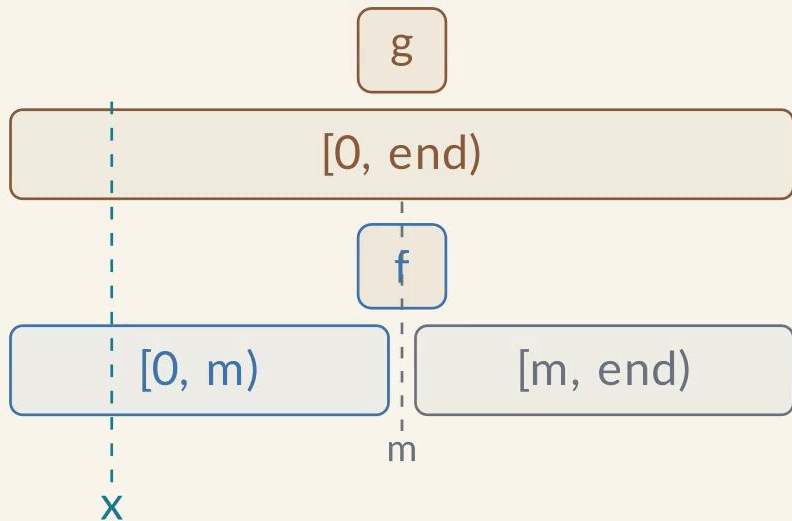
$$\text{query}(x) = \max_i f_i(x)$$

This is one of the more unintuitive parts.

Suppose we want to find the maximum value at  $x$ .

# Li Chao Queries

Think about a line that could be maximum at  $x$



Assume  $f$  is best at  $x$

$$x \in [0, m)$$

$$f(x) > g(x)$$

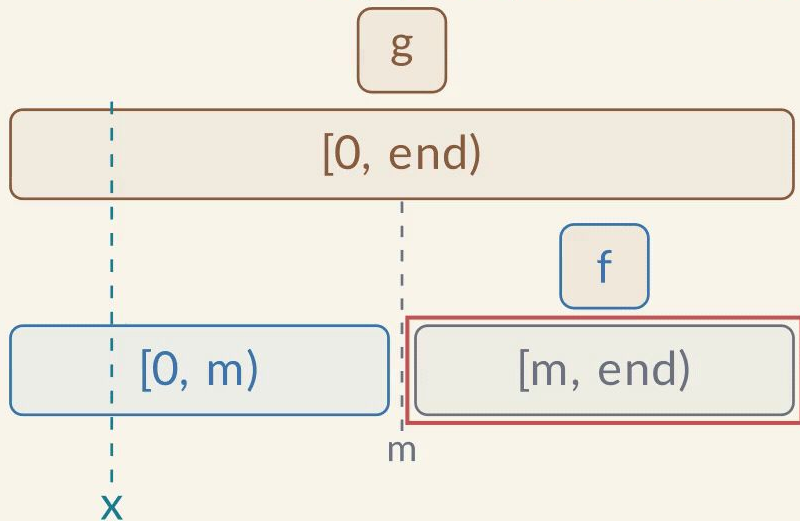
$g$  wins at the midpoint.

Which child must keep  $f$ ?

Assume  $f$  is best at  $x$ . Keep  $x$  fixed in the left half of  $g$ 's range.

# Li Chao Queries

Suppose, for contradiction:  $f$  is in  $g$ 's right child



What would sending  $f$  right require?

The left half must be ruled out for  $f$ .

$$g(t) \geq f(t) \quad \text{for } 0 \leq t < m$$

$$x < m \implies g(x) \geq f(x)$$

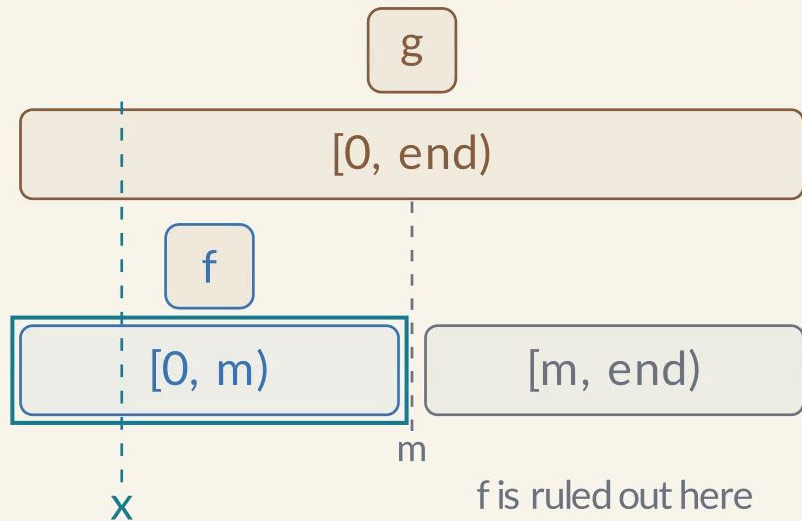
But we assumed  $f(x) > g(x)$ .

**Contradiction**

Routing  $f$  right would rule out the left half, contradicting  $f$  being best at  $x$  there.

# Li Chao Queries

Think about a line that could be maximum at  $x$



Why the maximum at  $x$  survives

We rule out a half only when another candidate is at least as good there.

The maximum value at  $x$  is preserved on a range that still contains  $x$ .

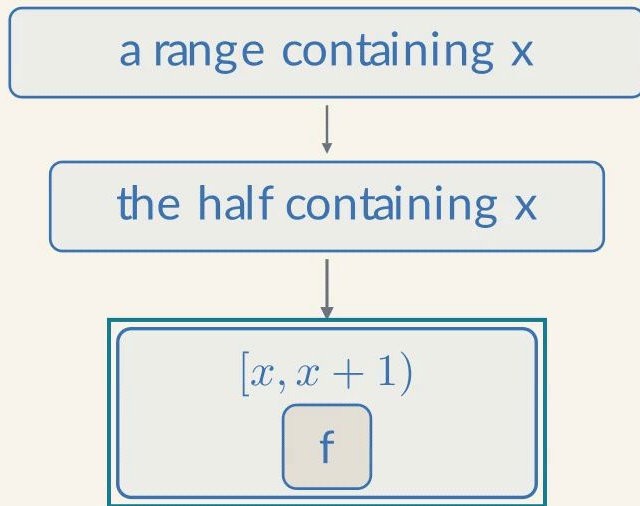
The candidate range still contains  $x$ .

Repeat the same reasoning whenever the line moves again.

A maximum at  $x$  cannot disappear from every range containing  $x$ .

# Li Chao Queries

What if the line reaches  $[x, x+1)$ ?



A worse line cannot replace  $f$   
 $f(x) > h(x) \implies \text{keep } f$   
Compare at  $x$ , then stop.  
No smaller range remains.  
Either tied line is enough.

The best value stays in the correct range:  $[x, x+1)$ .

Keep the best function at  $x$ , then stop. Its range still contains  $x$ .

# Li Chao Queries

---

Therefore

line is maximum at  $x$



line is stored in some  
range containing  $x$

# Li Chao Queries

---

So how do we answer the query?

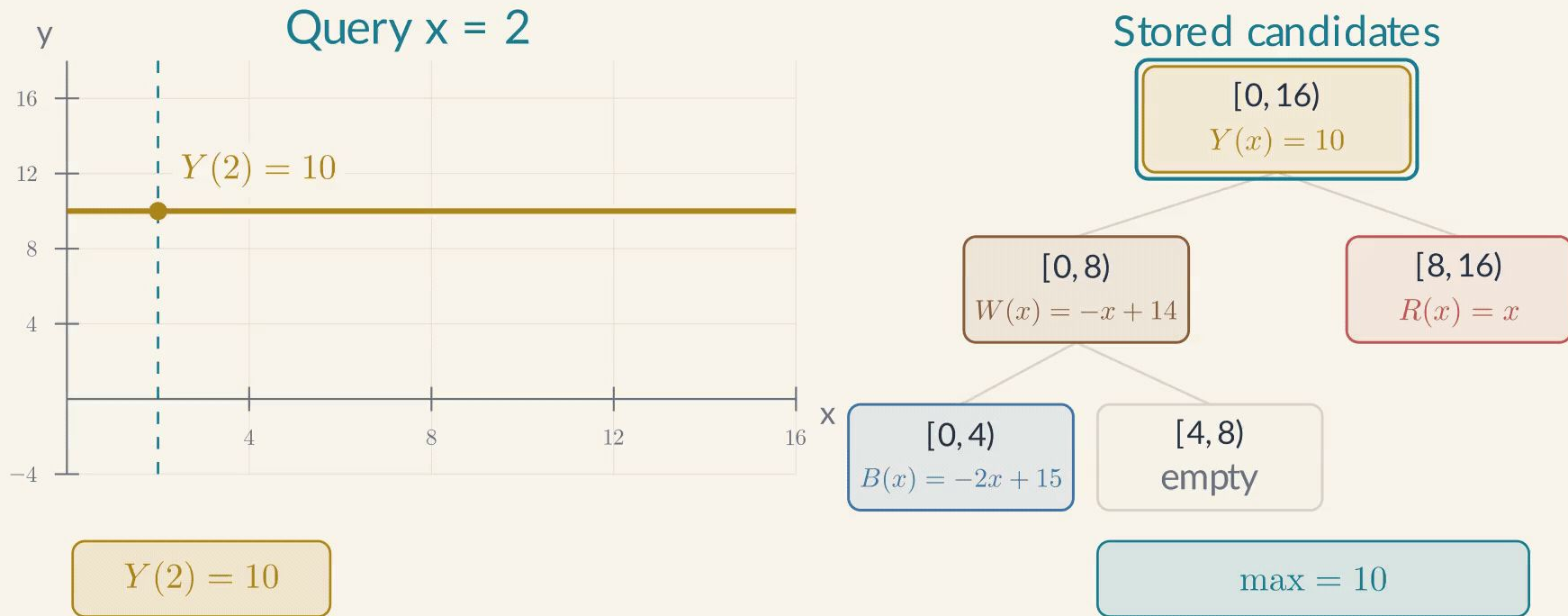
Look at every stored range containing  $x$   
Evaluate the line stored for that range at  $x$ .  
Take the maximum of all these values.

$$\text{query}(x) = \max_{\substack{\text{stored ranges } I \\ x \in I}} f_I(x)$$

$f_I$  is the line stored for range  $I$ .

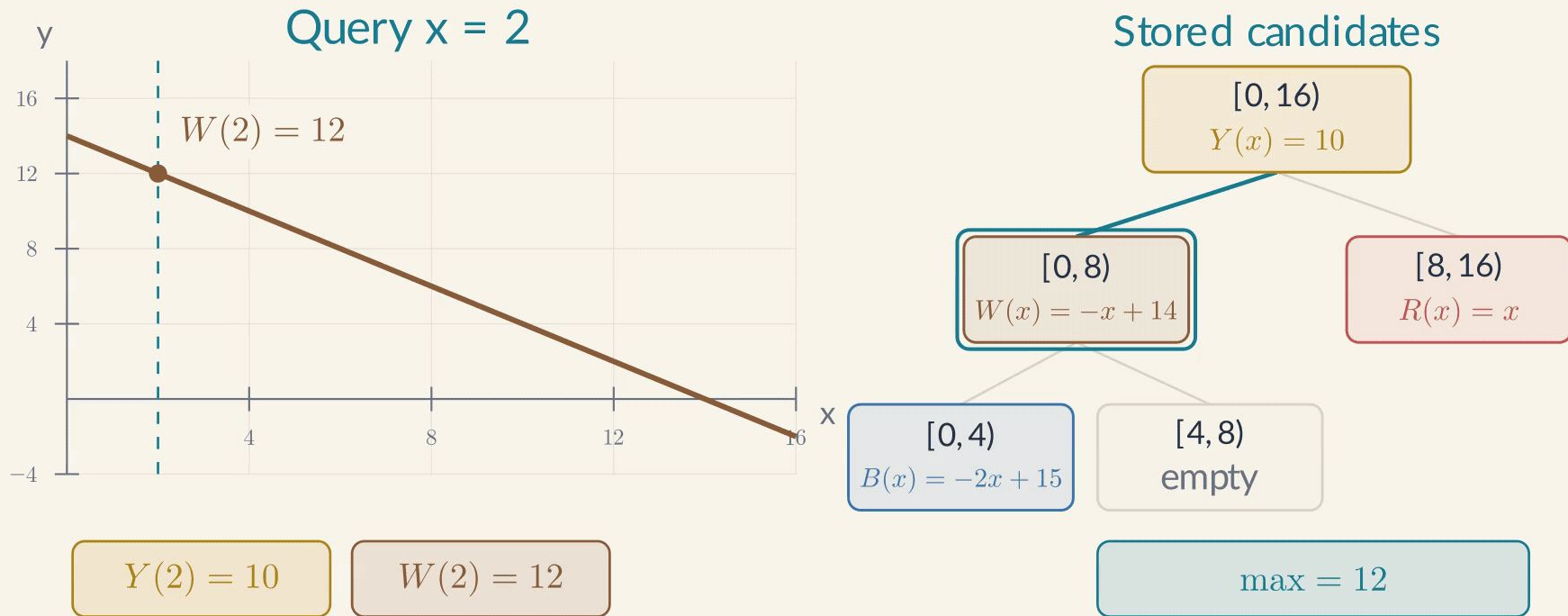
Visit the ranges containing  $x$ , evaluate their lines at  $x$ , and take the maximum.

# Li Chao Queries



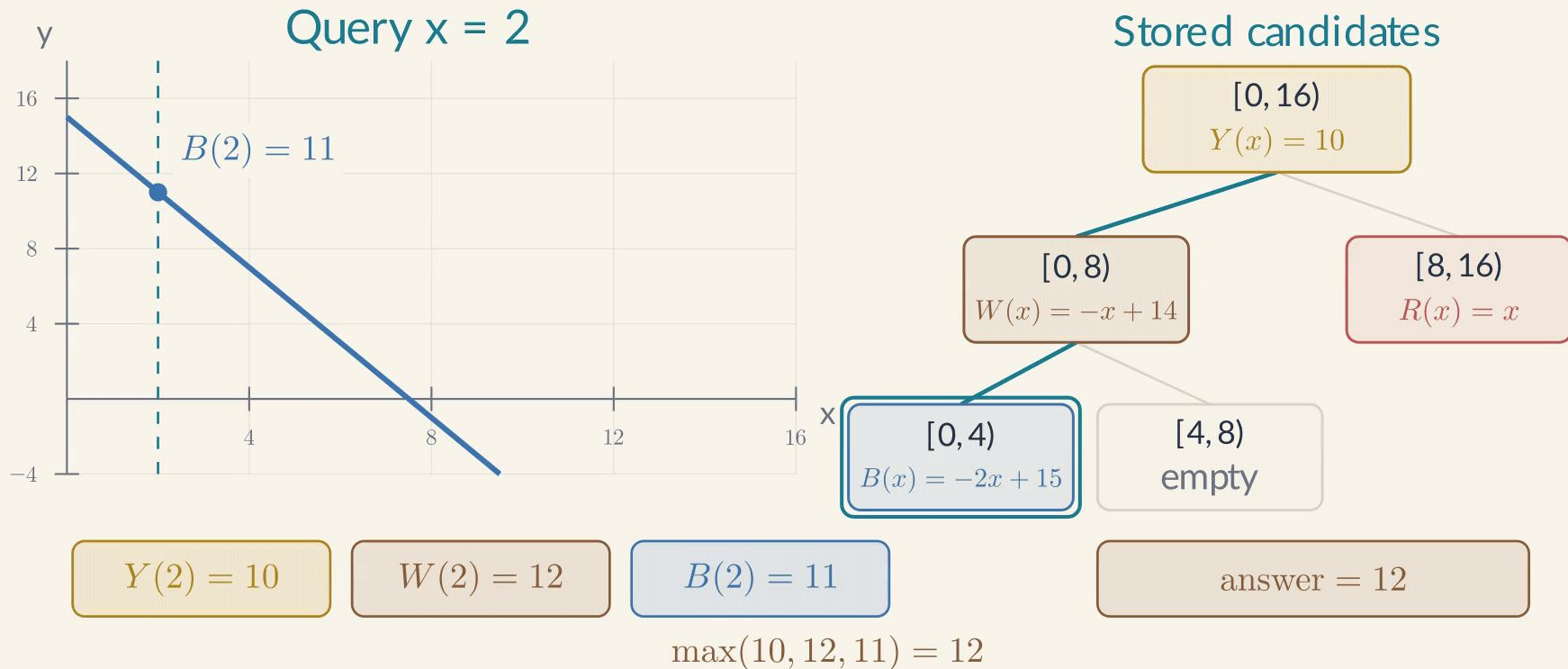
Visit  $[0, 8)$ : brown at  $x = 2$  gives 12. Update the maximum to 12.

# Li Chao Queries



Visit  $[0, 8)$ : brown at  $x = 2$  gives 12. Update the maximum to 12.

# Li Chao Queries



The next child is empty. Return  $\max(10, 12, 11) = 12$ .

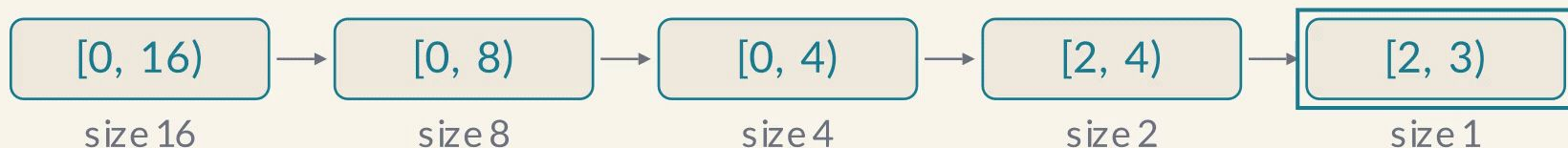
# Li Chao Queries

Exactly one half contains  $x$

$[0, \text{end})$ ,  $n = \text{end}$

$n$  is the number of integer query positions.

$x = 2$



$n \longrightarrow n/2 \longrightarrow n/4 \longrightarrow \dots \longrightarrow 1$

$[x, x + 1)$  Query:  $O(\log n)$

An empty child lets us stop earlier.

Halving reaches  $[x, x+1)$  in  $O(\log n)$  steps.

# Closing thoughts

---