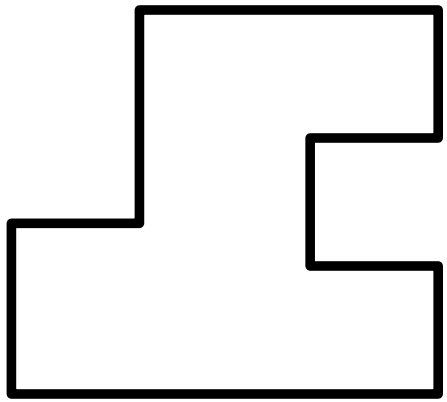
Minimum Rectilinear Polygons for Given Angle Sequences

W. Evans[1], K. Fleszar[2], P. Kindermann[2],
N. Saeedi[1], C.-S. Shin[3], A. Wolff[2]

1. UBC, Canada
2. Würzburg Univ., Germany
3. HUFS, Korea

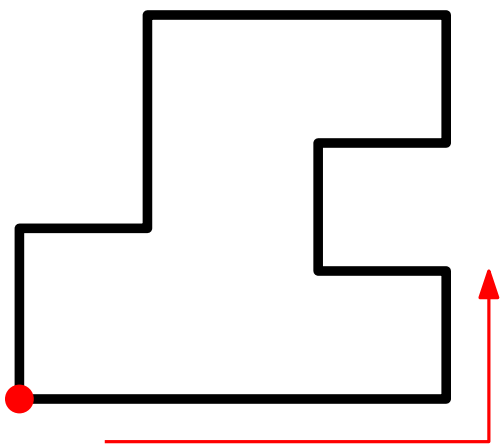
Problems

Rectilinear polygon P



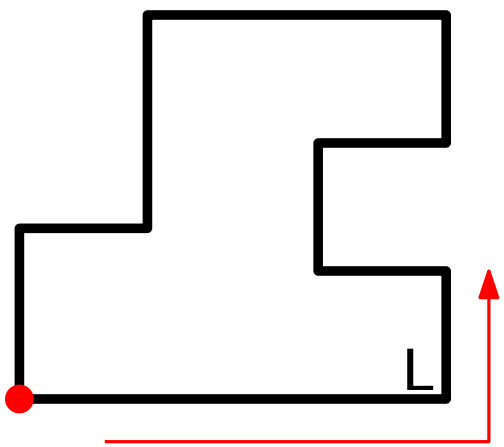
Problems

Rectilinear polygon P



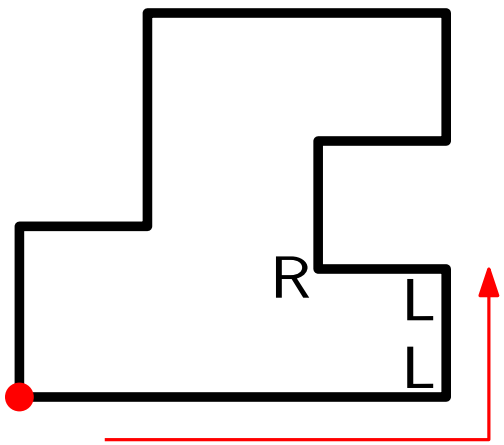
Problems

Rectilinear polygon P



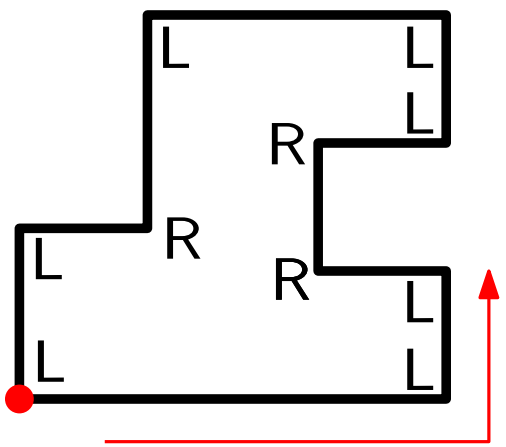
Problems

Rectilinear polygon P



Problems

Rectilinear polygon P ⟶ An angle sequence



⟶ LLRRLLLLRLL

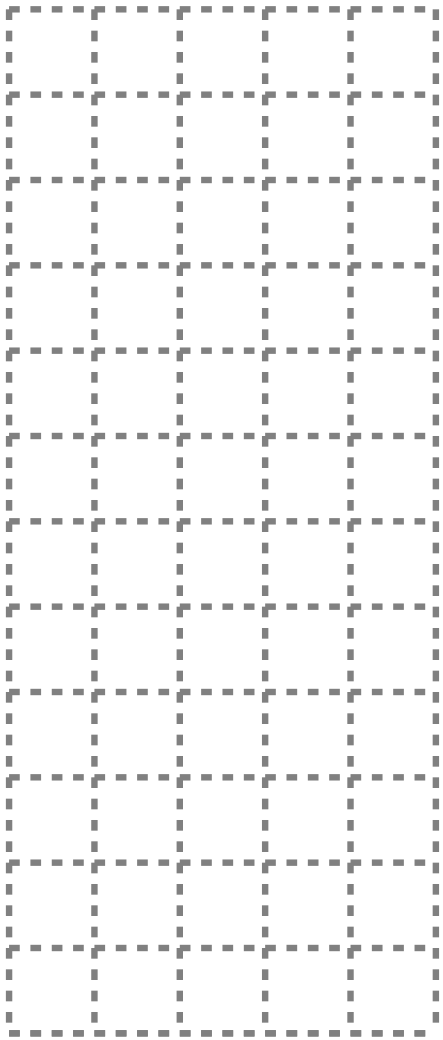
Problems

Rectilinear polygon P ← Angle sequence

← LLRRLLLLRLL

Problems

Rectilinear polygon P ← Angle sequence

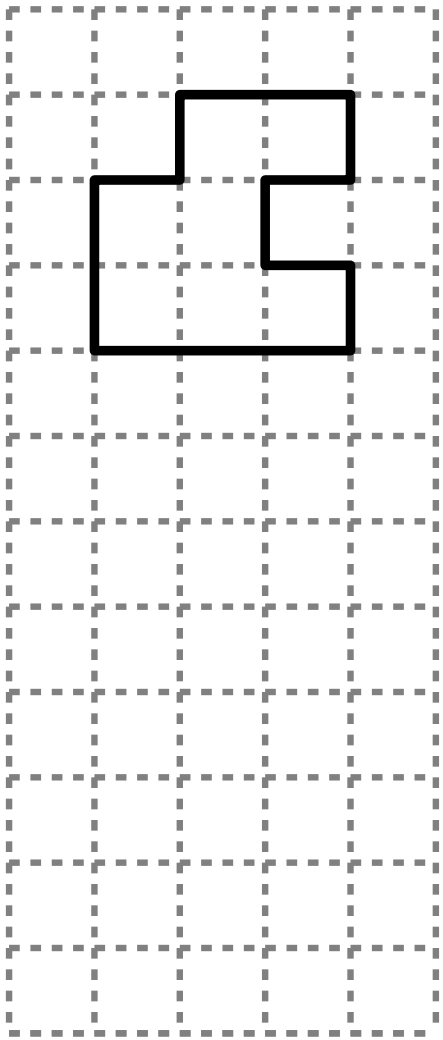


← LLRRLLLLRLL

on integer grid

Problems

Rectilinear polygon P ← Angle sequence

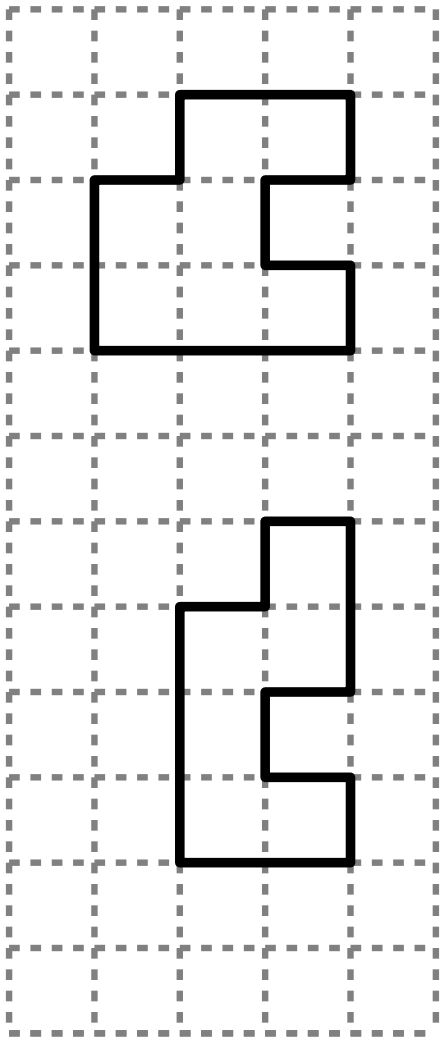


← LLRRLLLLRLL

on integer grid

Problems

Rectilinear polygon P ← Angle sequence

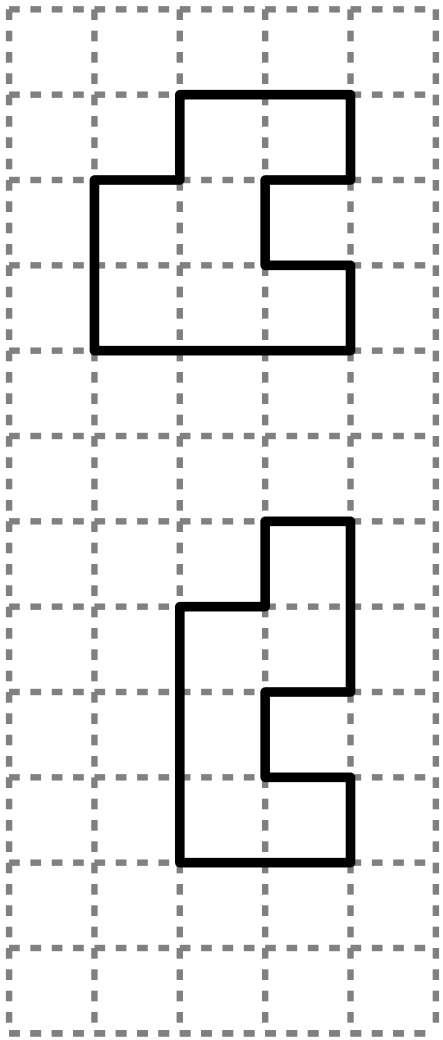


← LLRRLLLLRLL

on integer grid

Problems

Rectilinear polygon P ← Angle sequence



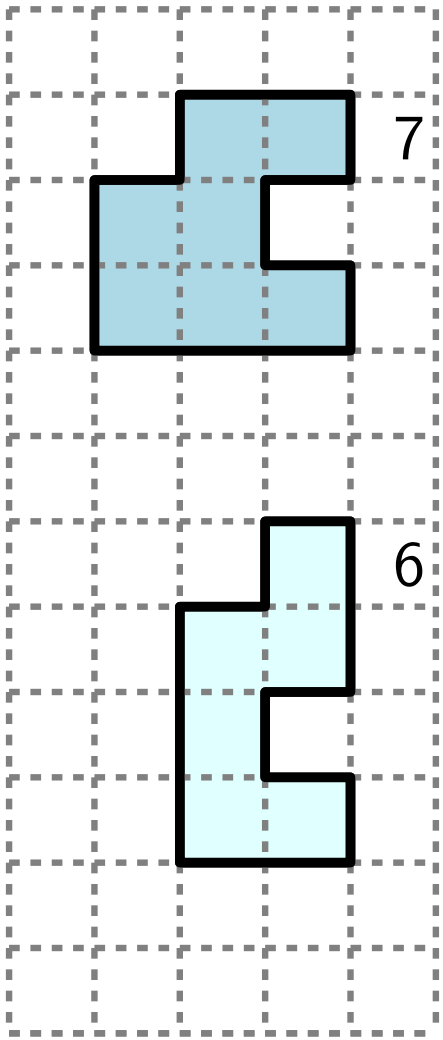
← on integer grid

LLRRLLLLRLL

Realize a given angle sequence as a rectilinear polygpon on the integer grid with ...

Problems

Rectilinear polygon P ← Angle sequence



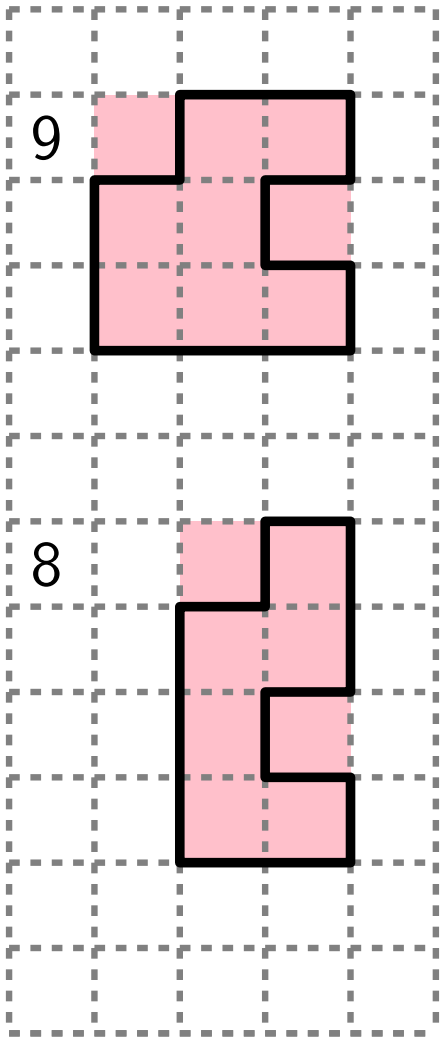
← LLRRLLLLRLL

on integer grid

Realize a given angle sequence as a rectilinear polygpon on the integer grid with minimum area

Problems

Rectilinear polygon P ← Angle sequence



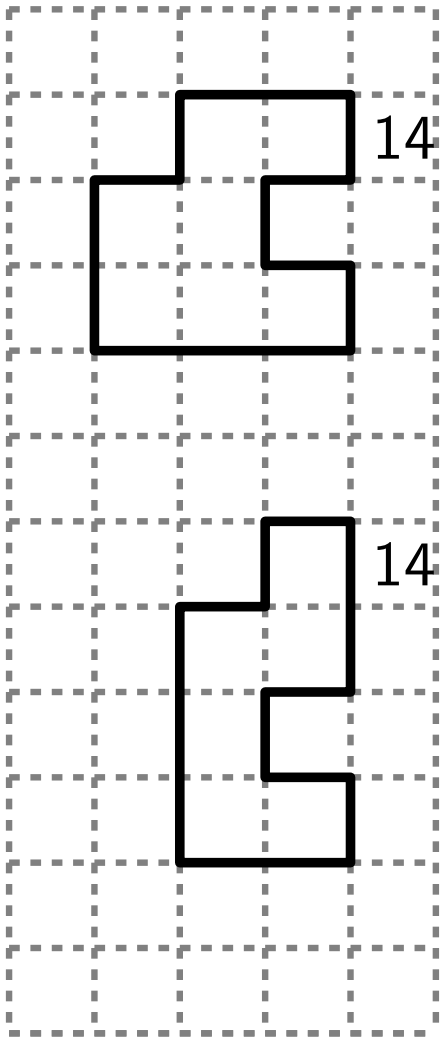
← on integer grid

LLRRLLLRLL

Realize a given angle sequence as a rectilinear polygpon on the integer grid with minimum bounding box

Problems

Rectilinear polygon P ← Angle sequence



← on integer grid

LLRRLLLLRLL

Realize a given angle sequence as a rectilinear polygpon on the integer grid with minimum perimeter

Previous Work

$P(S) =$ an optimal polygon that realizes an angle seq. S of length n

Previous Work

$P(S) =$ an optimal polygon that realizes an angle seq. S of length n

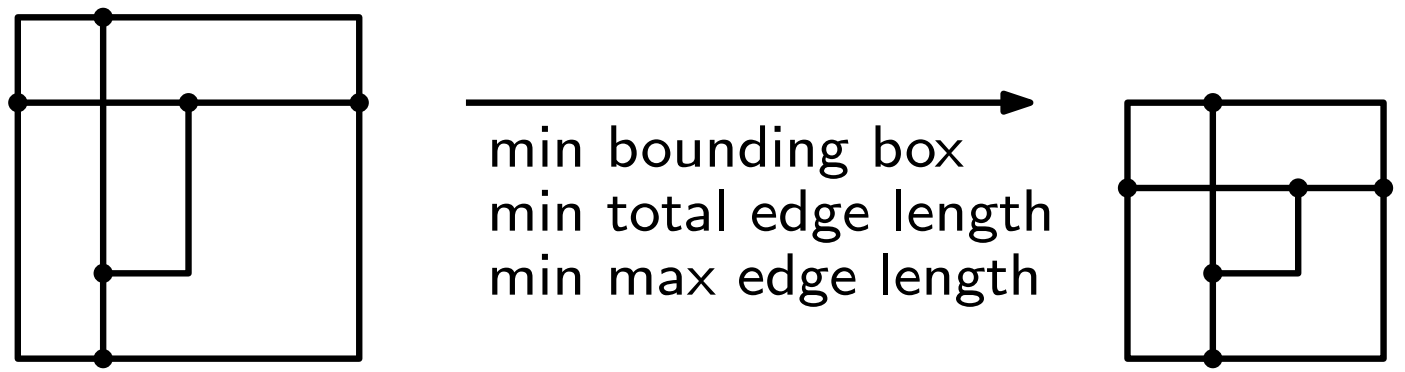
1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

Previous Work

$P(S)$ = an optimal polygon that realizes an angle seq. S of length n

1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

   Patrignani (2001):



Orthogonal representation

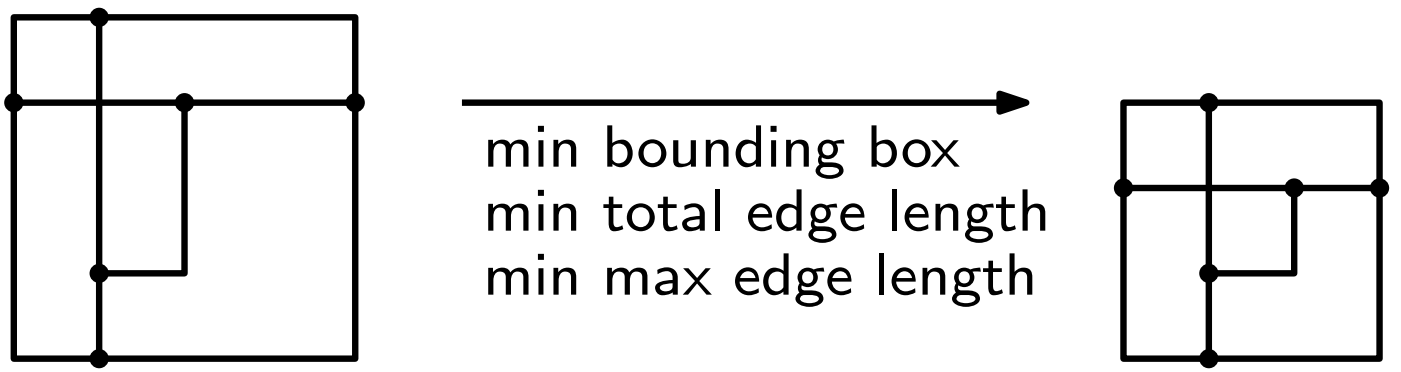
Compaction

Previous Work

$P(S)$ = an optimal polygon that realizes an angle seq. S of length n

1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

   Patrignani (2001): showed all three criteria are NP-complete



Orthogonal representation

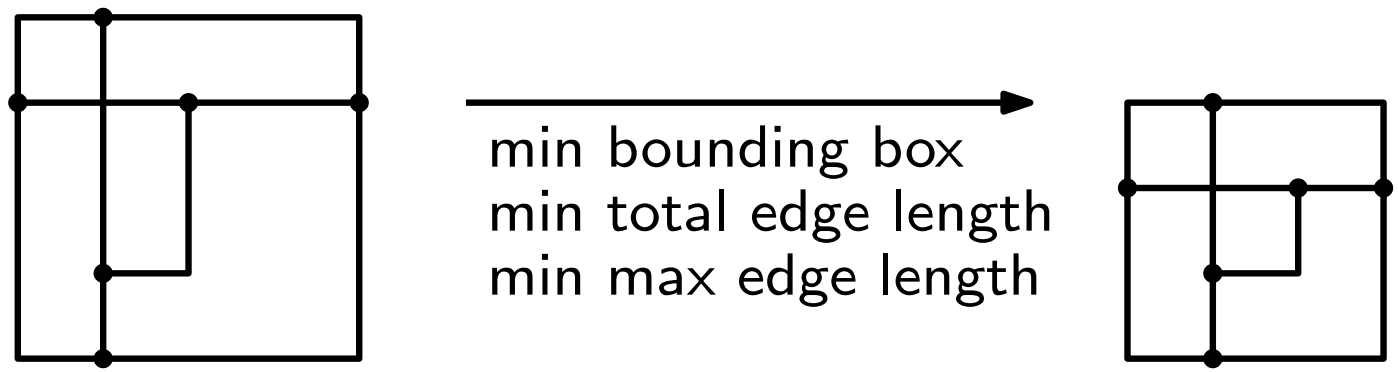
Compaction

Previous Work

$P(S)$ = an optimal polygon that realizes an angle seq. S of length n

1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

   Patrignani (2001): showed all three criteria are NP-complete



Orthogonal representation — Compaction

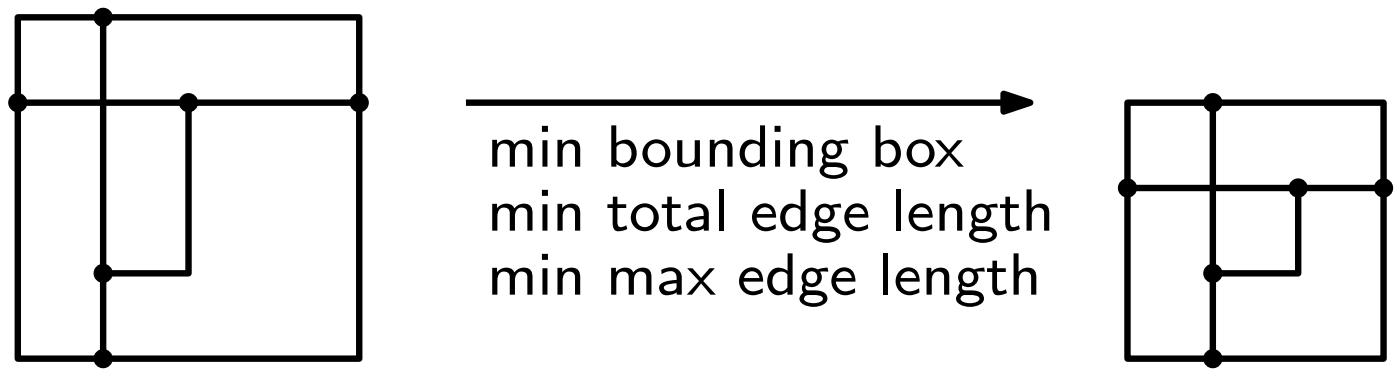
Our bounding box problem corresponds to the special case that the input graph is a simple cycle.

Previous Work

$P(S)$ = an optimal polygon that realizes an angle seq. S of length n

1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

   Patrignani (2001): showed all three criteria are NP-complete



Orthogonal representation Compaction

Our bounding box problem corresponds to the special case that the input graph is a simple cycle.

Open!

Previous Work

$P(S) =$ an optimal polygon that realizes an angle seq. S of length n

1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

2. Area minimization: Bae, Okamoto, Shin (2012)
   - Bound $\delta(n)$, the minimum area of $P(S)$ for a given length n

Previous Work

$P(S) =$ an optimal polygon that realizes an angle seq. S of length n

1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

2. Area minimization: Bae, Okamoto, Shin (2012)
   - Bound $\delta(n)$, the minimum area of $P(S)$ for a given length n

   $\delta(n) = \frac{n}{2} - 1$ if $n \equiv 4 \bmod 8$, $\frac{n}{2}$ otherwise

Previous Work

$P(S) =$ an optimal polygon that realizes an angle seq. S of length n

1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

2. Area minimization: Bae, Okamoto, Shin (2012)
   - Bound $\delta(n)$, the minimum area of $P(S)$ for a given length n

     $\delta(n) = \frac{n}{2} - 1$ if $n \equiv 4 \bmod 8$, $\frac{n}{2}$ otherwise

   - Bound $\Delta(n)$, the maximum of $P(S)$ for a given length n

     $\Delta(n) = \frac{1}{8}(n-2)(n+4)$

Previous Work

$P(S) =$ an optimal polygon that realizes an angle seq. S of length n

1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

2. Area minimization: Bae, Okamoto, Shin (2012)
   - Bound $\delta(n)$, the minimum area of $P(S)$ for a given length n

     $\delta(n) = \frac{n}{2} - 1$ if $n \equiv 4 \bmod 8$, $\frac{n}{2}$ otherwise

   - Bound $\Delta(n)$, the maximum of $P(S)$ for a given length n

     $\Delta(n) = \frac{1}{8}(n-2)(n+4)$ ← meaning that one can realize any S in this area, also tight

Previous Work

$P(S) =$ an optimal polygon that realizes an angle seq. S of length n

1. How hard it is to compute $P(S)$? Looks hard, but no proof yet not even for the minimum bounding box.

2. Area minimization: Bae, Okamoto, Shin (2012)
   - Bound $\delta(n)$, the minimum area of $P(S)$ for a given length n

     $\delta(n) = \frac{n}{2} - 1$ if $n \equiv 4 \bmod 8$, $\frac{n}{2}$ otherwise

   - Bound $\Delta(n)$, the maximum of $P(S)$ for a given length n

     $\Delta(n) = \frac{1}{8}(n-2)(n+4)$ ← meaning that one can realize any S in this area, also tight

- No other related results

Our results

1. Show that computing $P(S)$ under three criteria (area, bounding box, perimeter) are NP-hard.

Our results

1. Show that computing $P(S)$ under three criteria (area, bounding box, perimeter) are NP-hard.

2. Give efficient algorithms for special types of angle sequences:

Our results

1. Show that computing $P(S)$ under three criteria (area, bounding box, perimeter) are NP-hard.

2. Give efficient algorithms for special types of angle sequences:

	Area	Bounding Box	Perimeter
x-monotone			

Our results

1. Show that computing $P(S)$ under three criteria (area, bounding box, perimeter) are NP-hard.

2. Give efficient algorithms for special types of angle sequences:

	Area	Bounding Box	Perimeter
x-monotone	$O(n^4)$	$O(n^3)$	$O(n^2)$

Our results

1. Show that computing $P(S)$ under three criteria (area, bounding box, perimeter) are NP-hard.

2. Give efficient algorithms for special types of angle sequences:

	Area	Bounding Box	Perimeter
x-monotone	$O(n^4)$	$O(n^3)$	$O(n^2)$
xy-monotone			

Our results

1. Show that computing $P(S)$ under three criteria (area, bounding box, perimeter) are NP-hard.

2. Give efficient algorithms for special types of angle sequences:

	Area	Bounding Box	Perimeter
x-monotone	$O(n^4)$	$O(n^3)$	$O(n^2)$
xy-monotone	$O(n)$	$O(n)$	$O(n)$

Our results

1. Show that computing $P(S)$ under three criteria (area, bounding box, perimeter) are NP-hard.

2. Give efficient algorithms for special types of angle sequences:

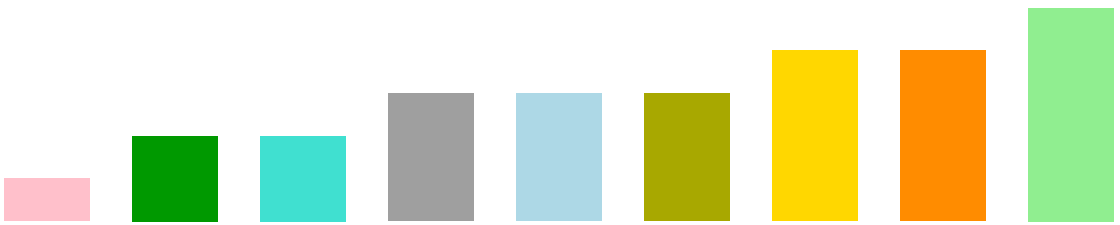
	Area	Bounding Box	Perimeter
x-monotone	$O(n^4)$	$O(n^3)^\star$	$O(n^2)^\star$
xy-monotone	$O(n)^\star$	$O(n)$	$O(n)$

Computing Polygons with Bounding Boxes of Minimum Area

NP-Hardness

We reduce from 3-Partition:

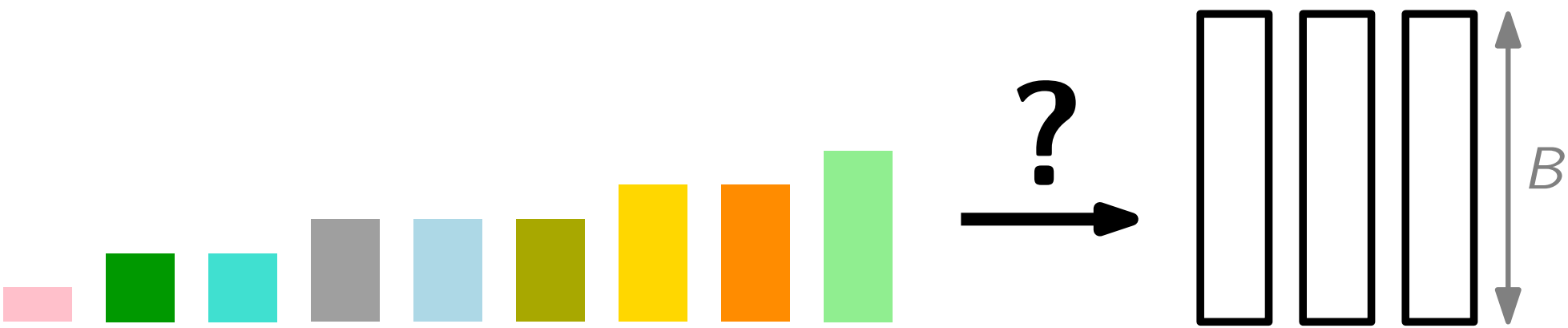
> Given a multiset S of $n = 3m$ integers with $\sum S = mB$, is there a partition of S into m subsets $S_1, \ldots, S_m$ such that $\sum S_i = B$ for each i?



NP-Hardness

We reduce from 3-Partition:

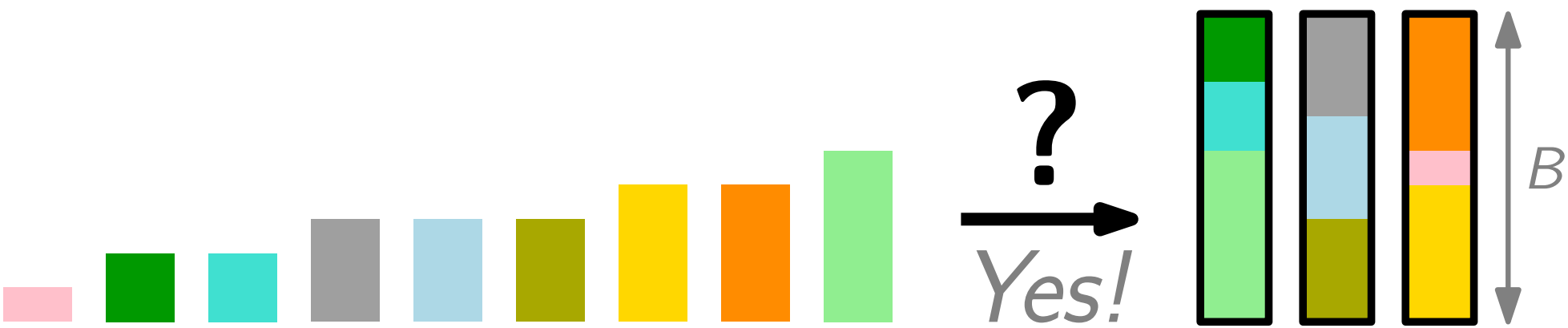
> Given a multiset S of $n = 3m$ integers with $\sum S = mB$, is there a partition of S into m subsets $S_1, \ldots, S_m$ such that $\sum S_i = B$ for each i?



NP-Hardness

We reduce from 3-Partition:

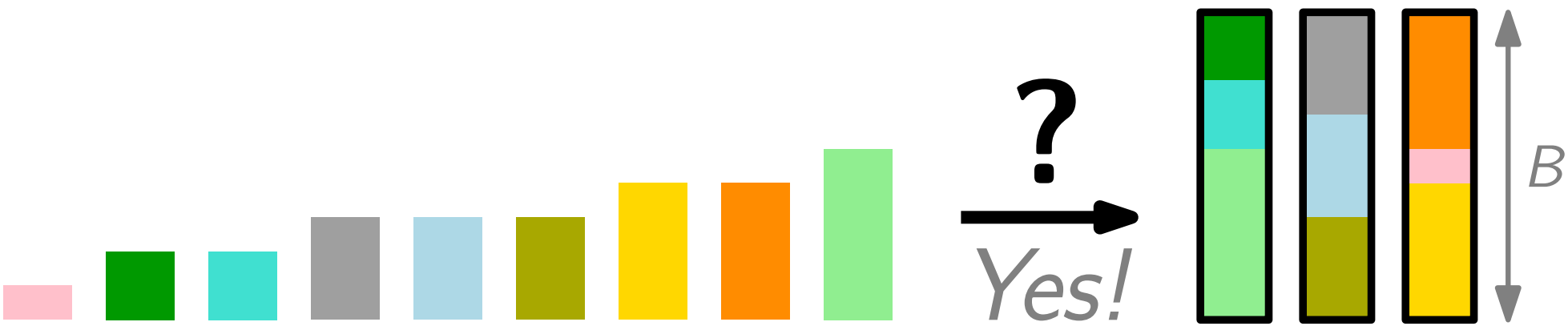
> Given a multiset S of $n = 3m$ integers with $\sum S = mB$, is there a partition of S into m subsets $S_1, \ldots, S_m$ such that $\sum S_i = B$ for each i?



NP-Hardness

We reduce from 3-Partition:

Given a multiset S of $n = 3m$ integers with $\sum S = mB$, is there a partition of S into m subsets $S_1, \ldots, S_m$ such that $\sum S_i = B$ for each i?



3-Partition is NP-hard even if (i) $B = \text{poly}(m)$
(ii) for each $s \in S$, we have $B/4 < s < B/2$ ($\Rightarrow$ all $|S_i| = 3$).

Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

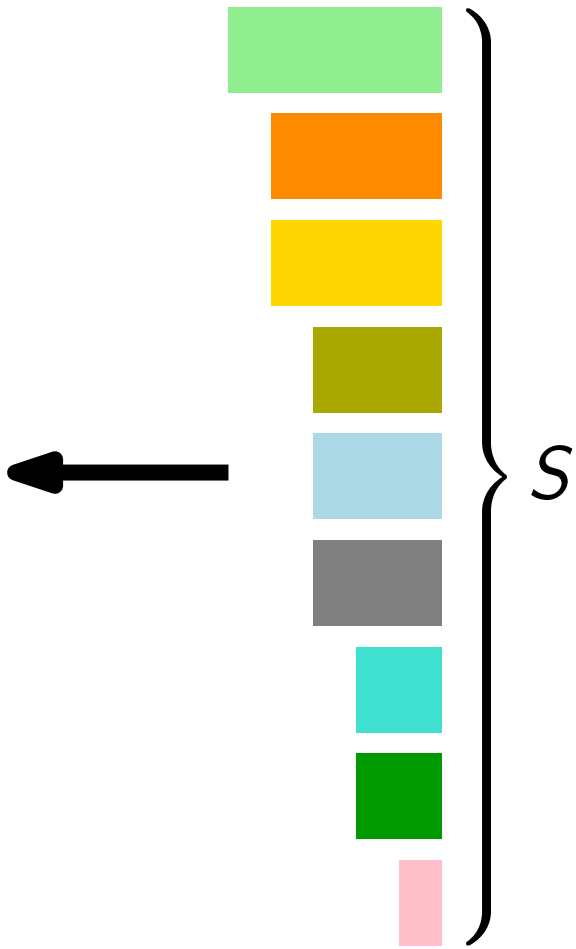
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

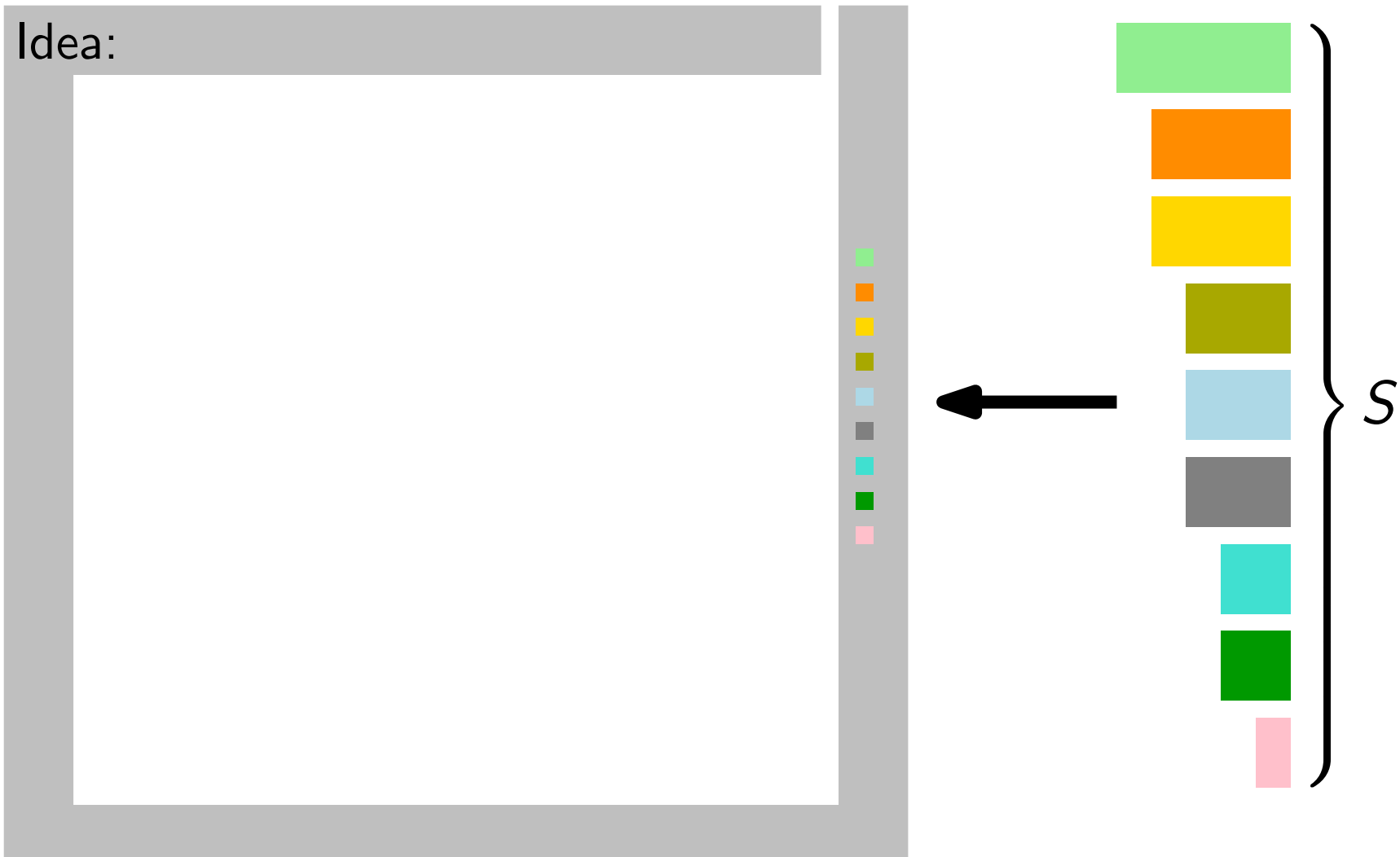
Idea:



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

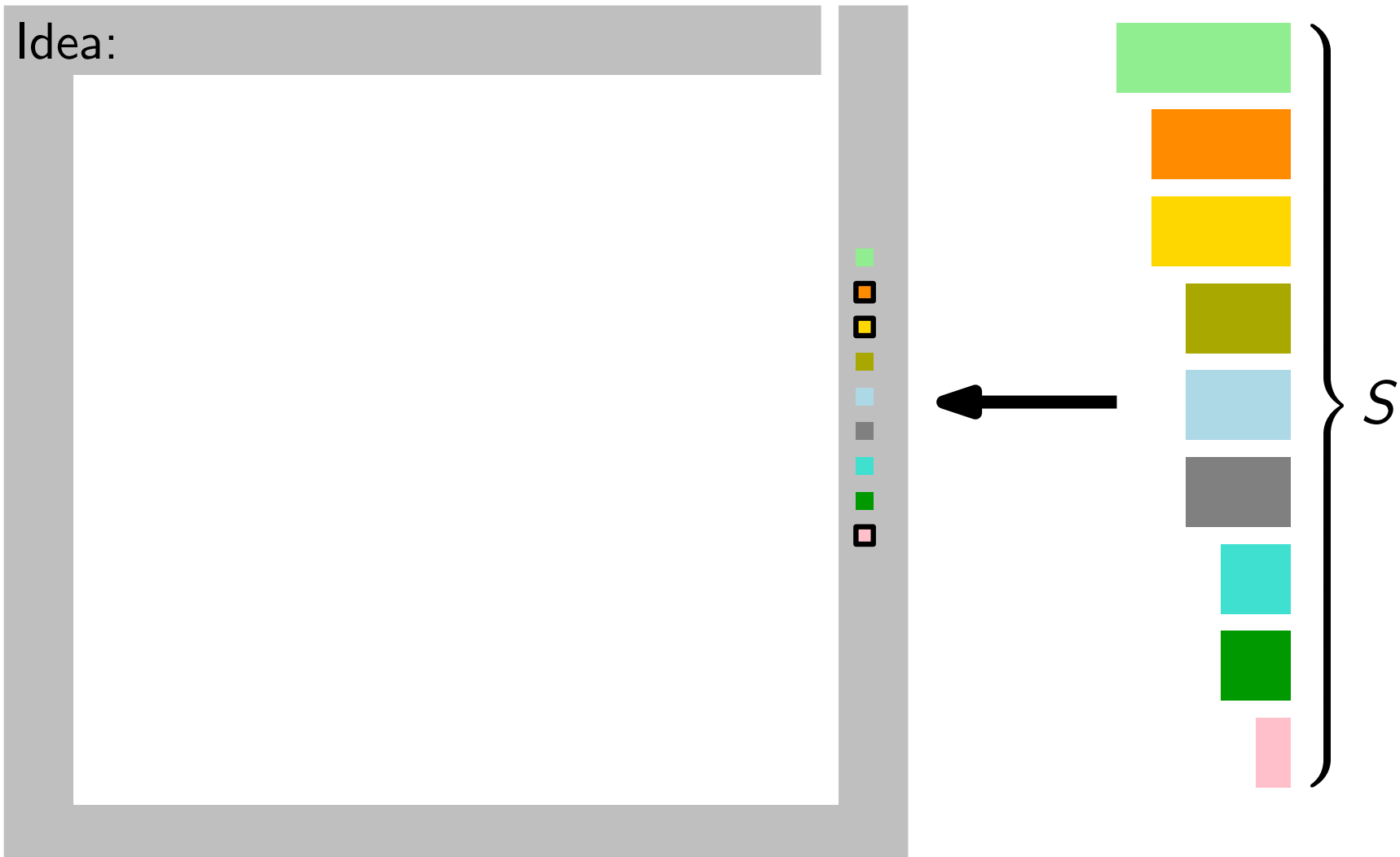
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

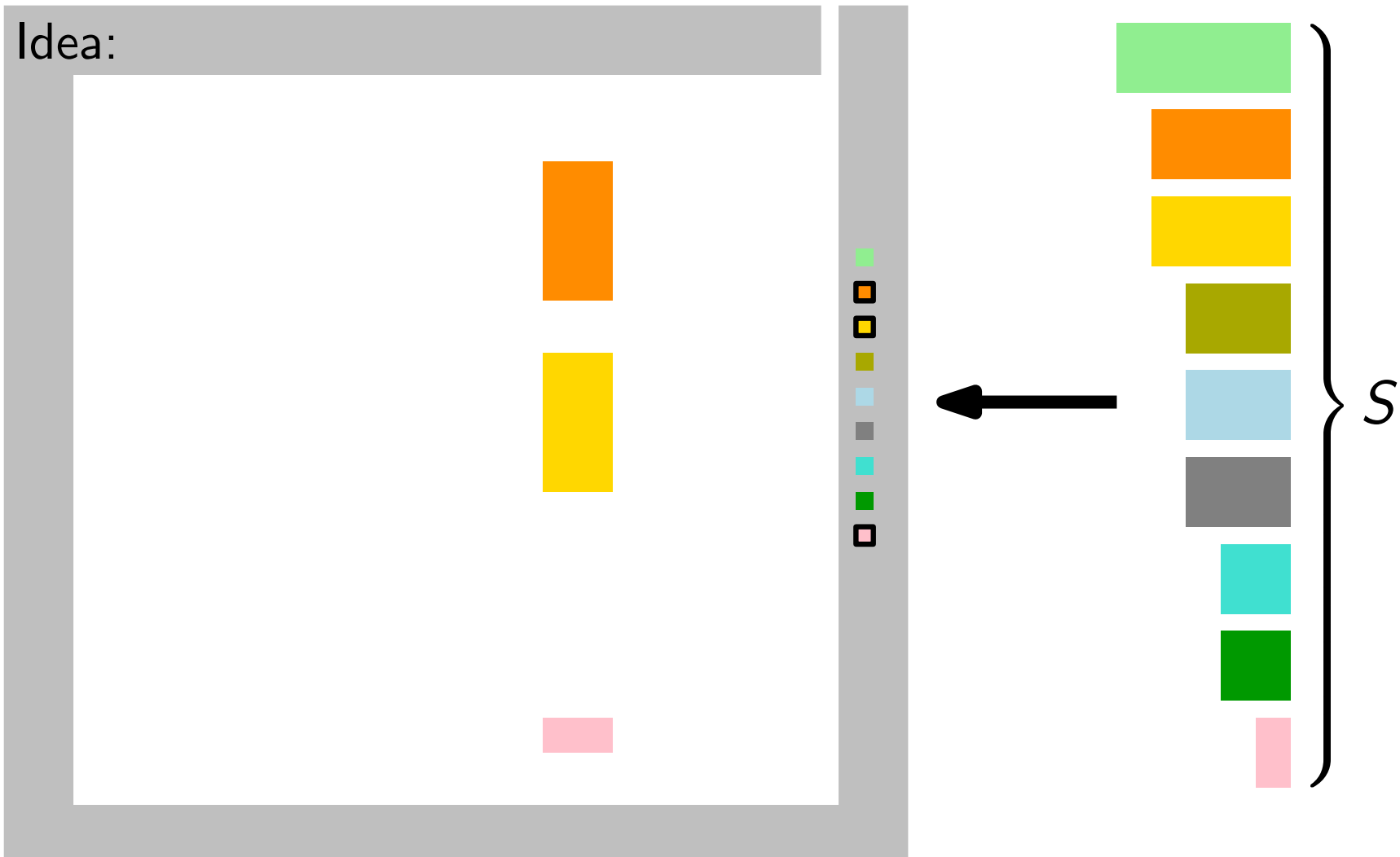
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

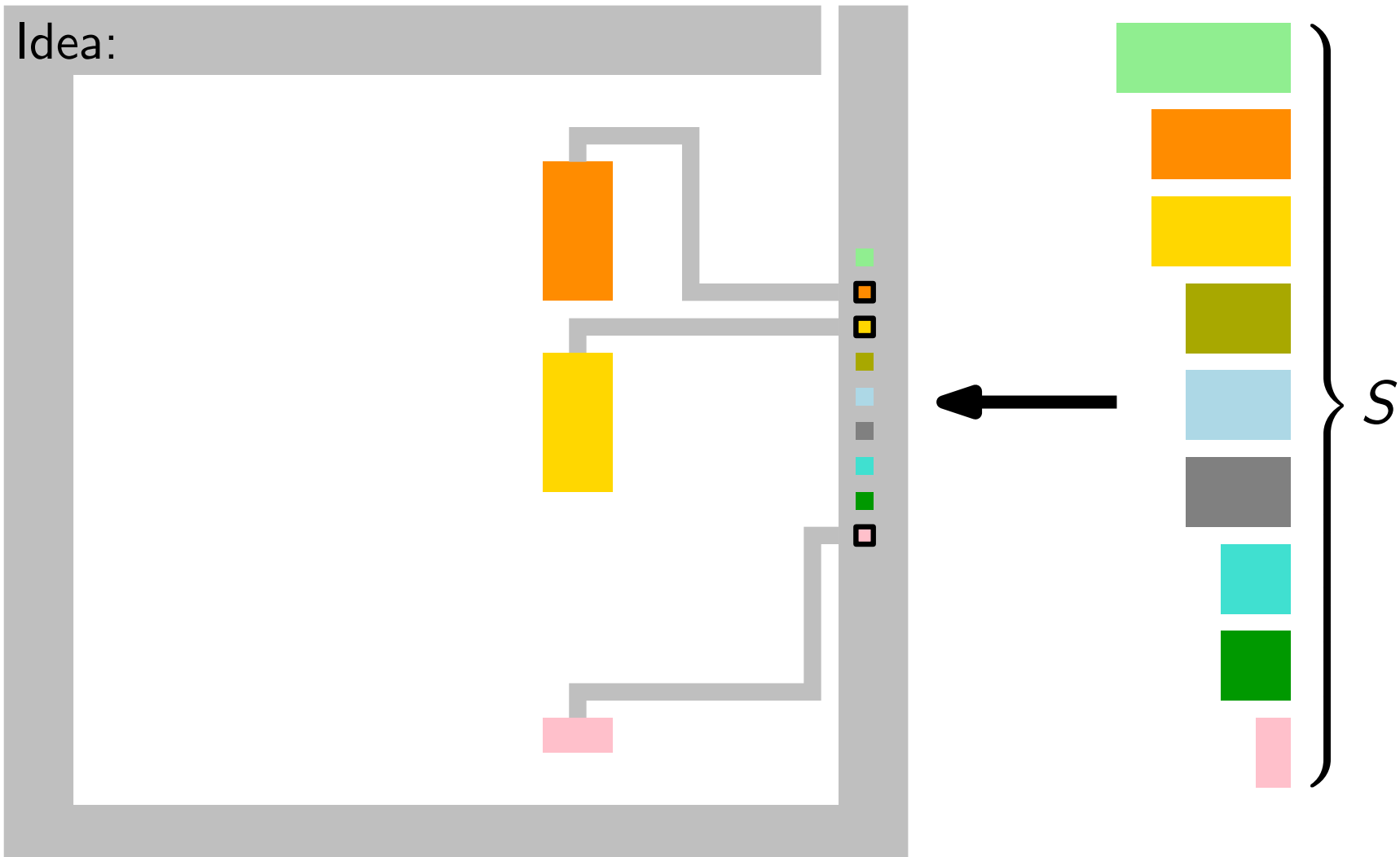
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

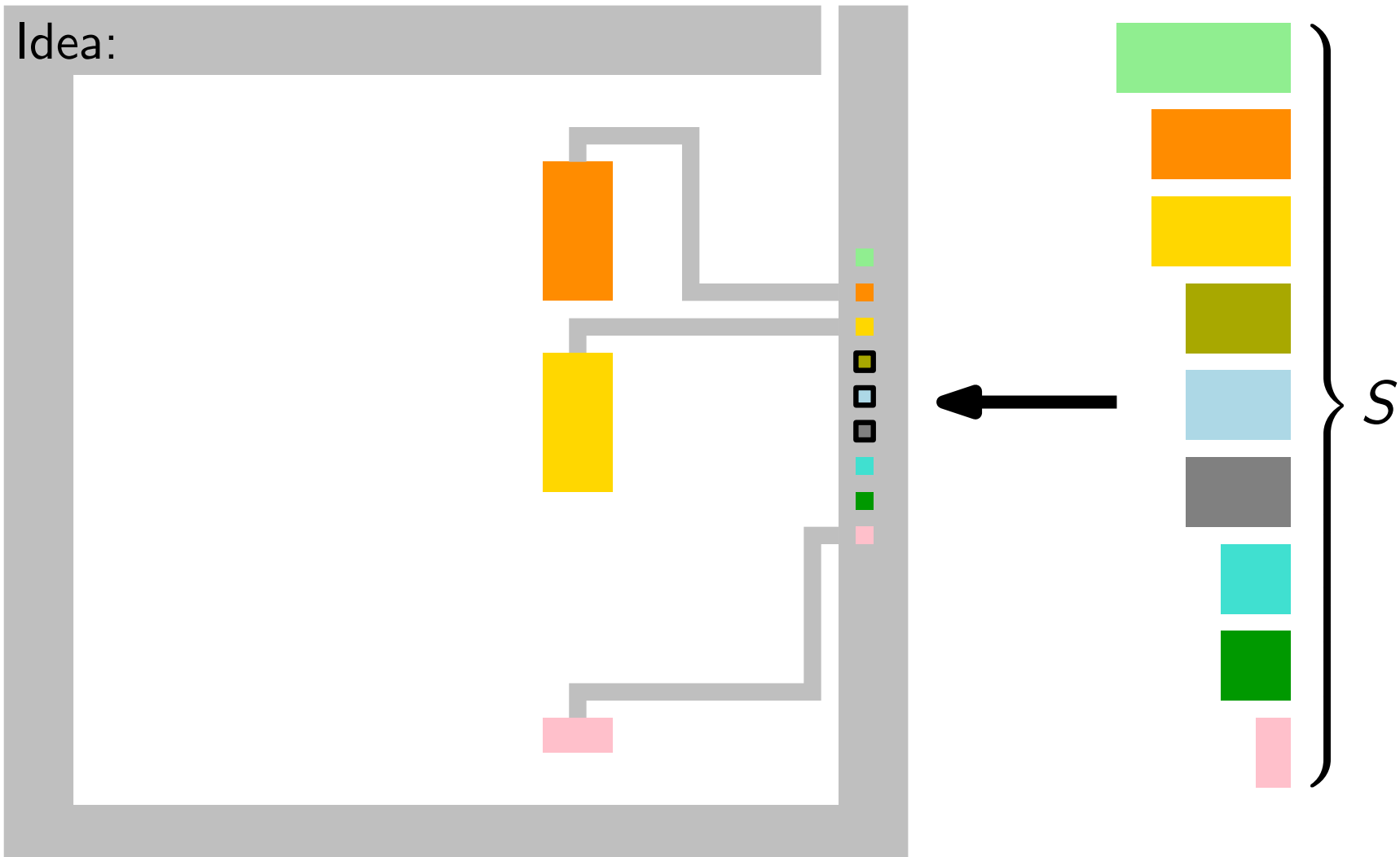
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

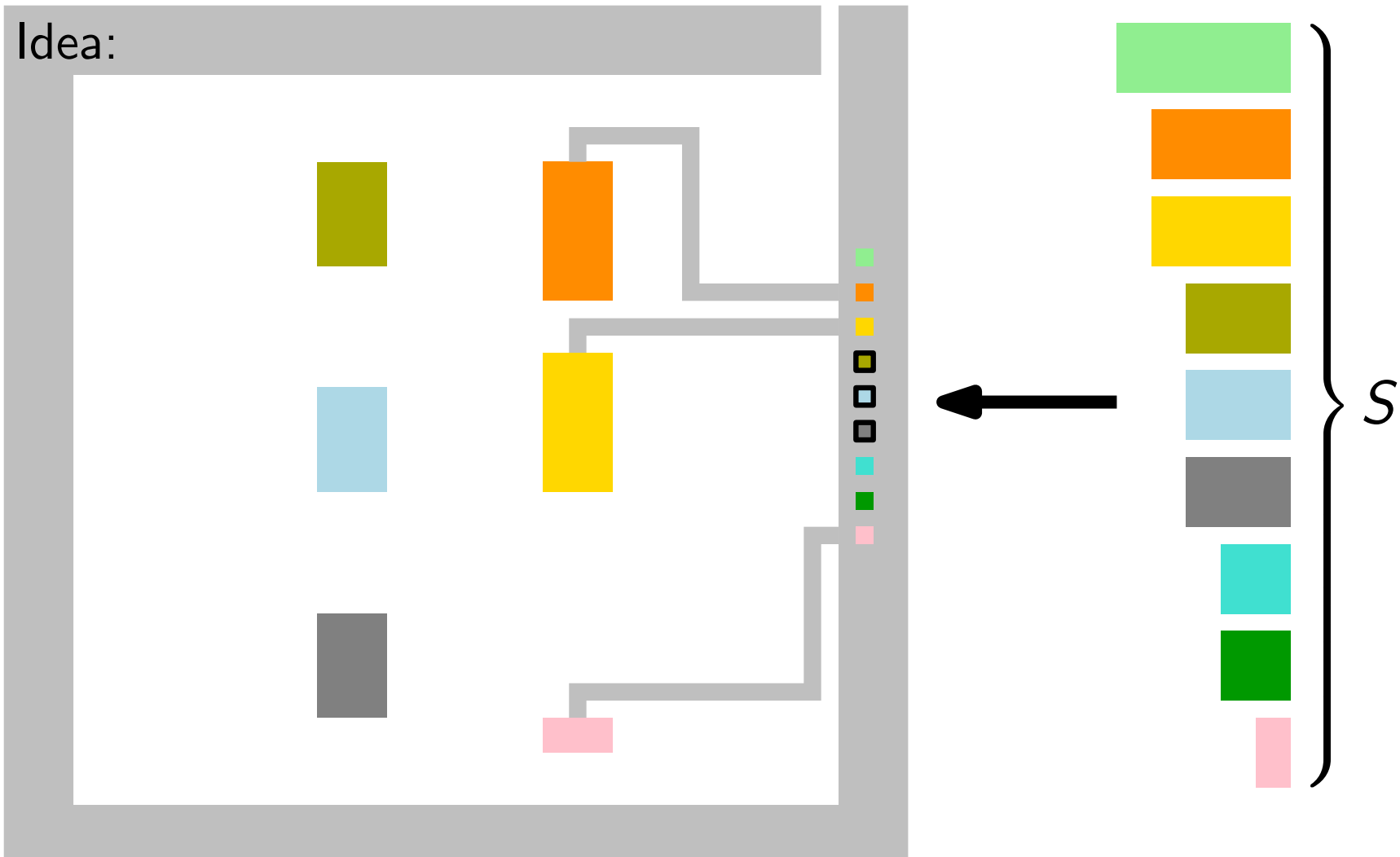
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

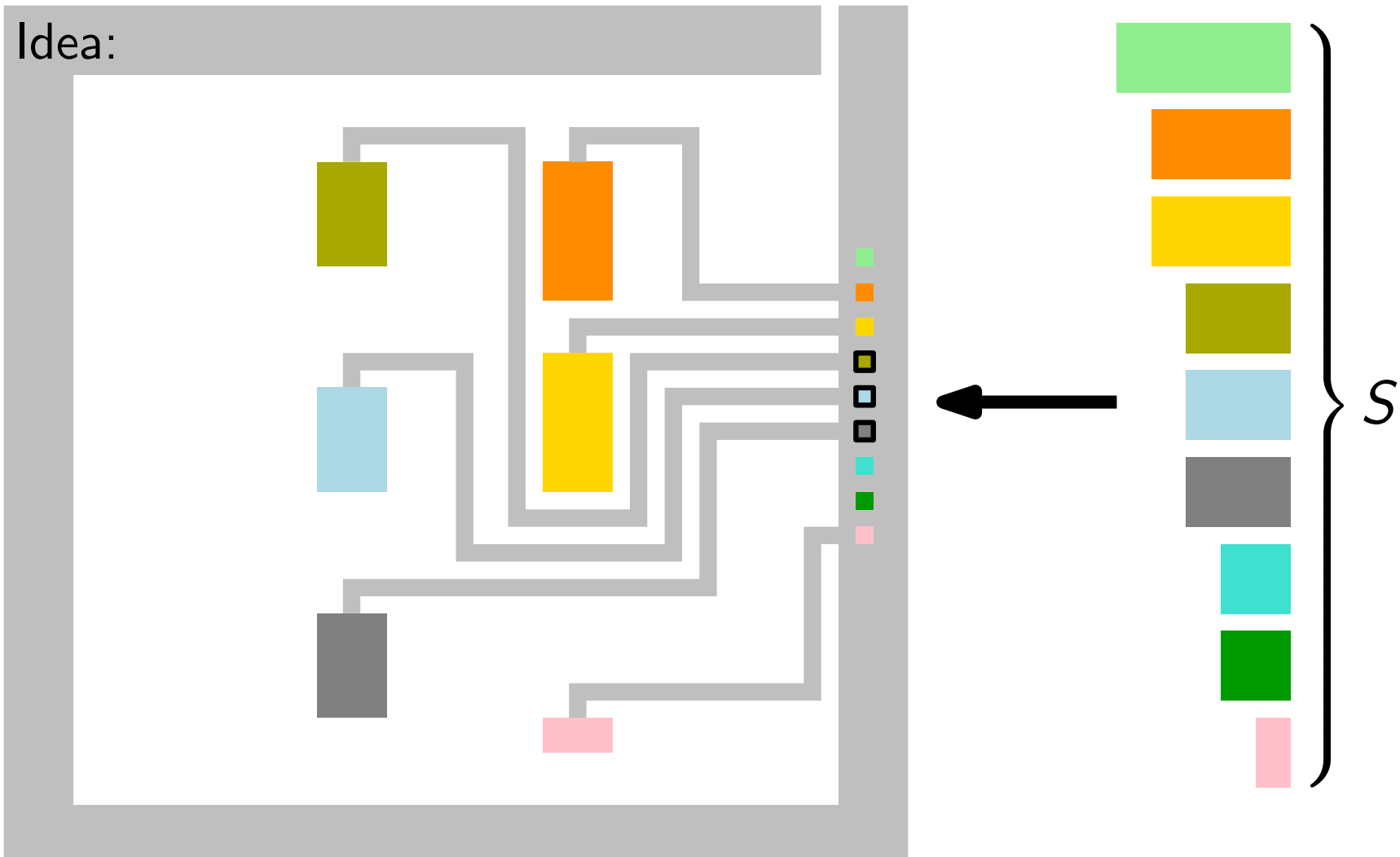
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

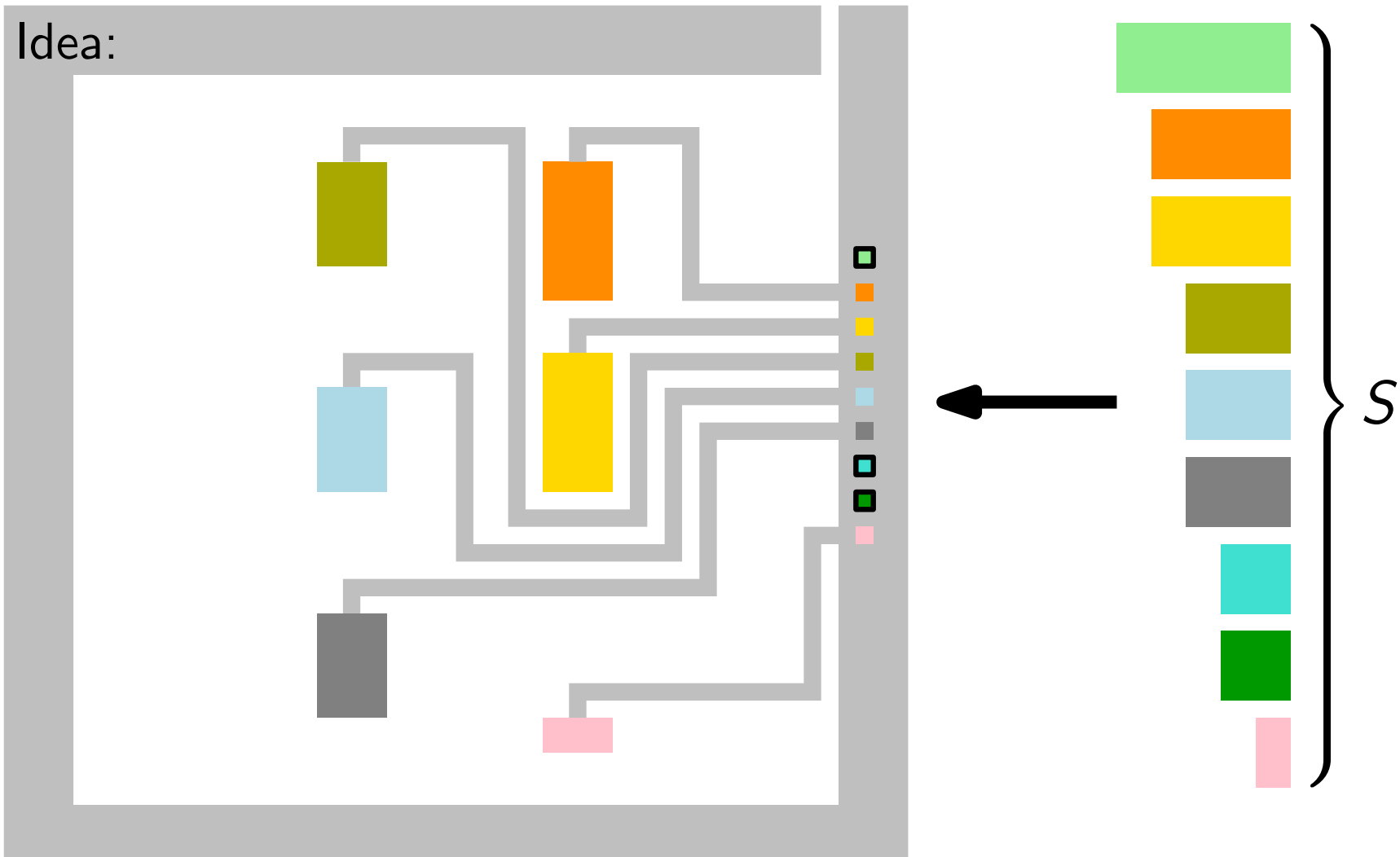
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

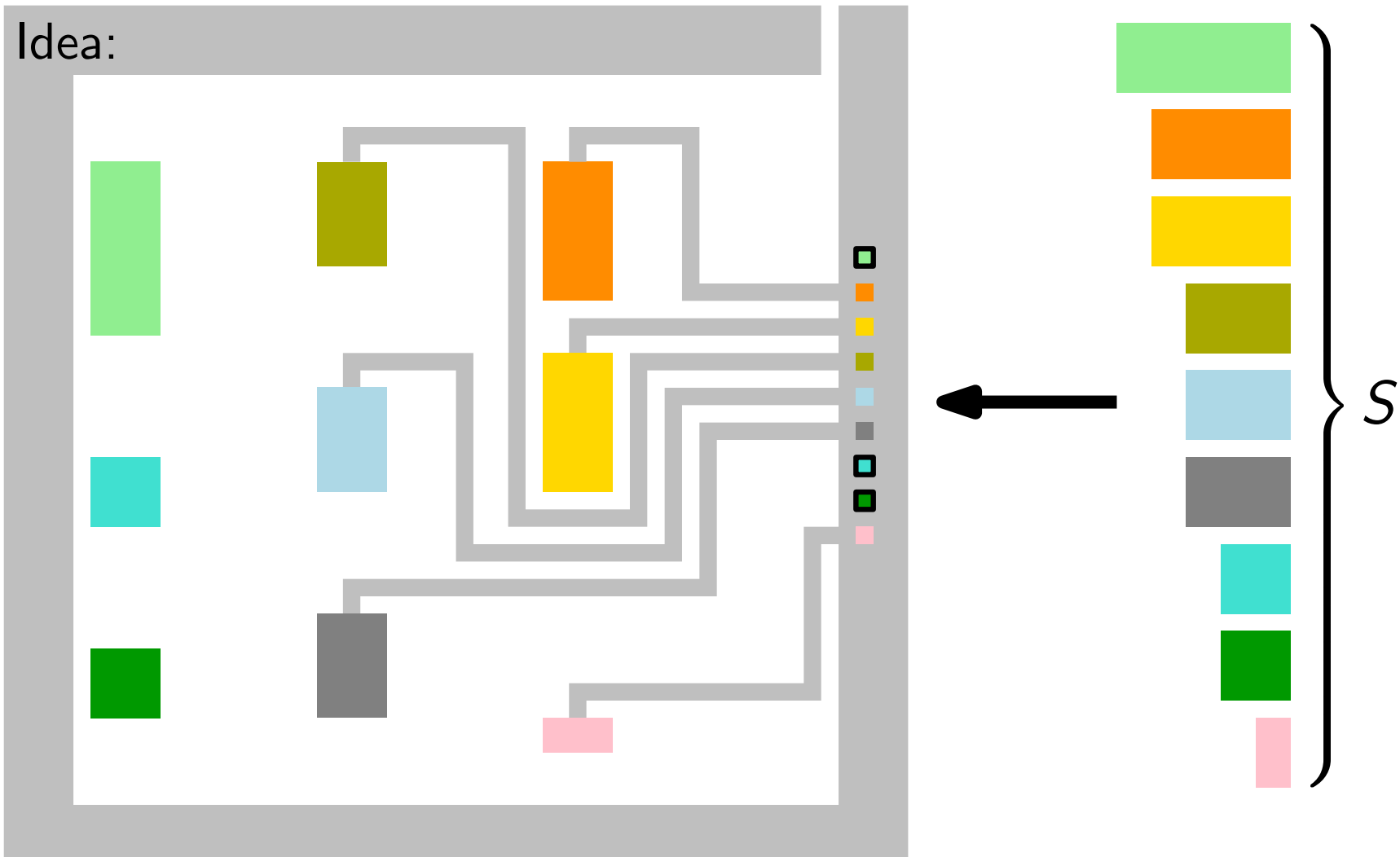
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

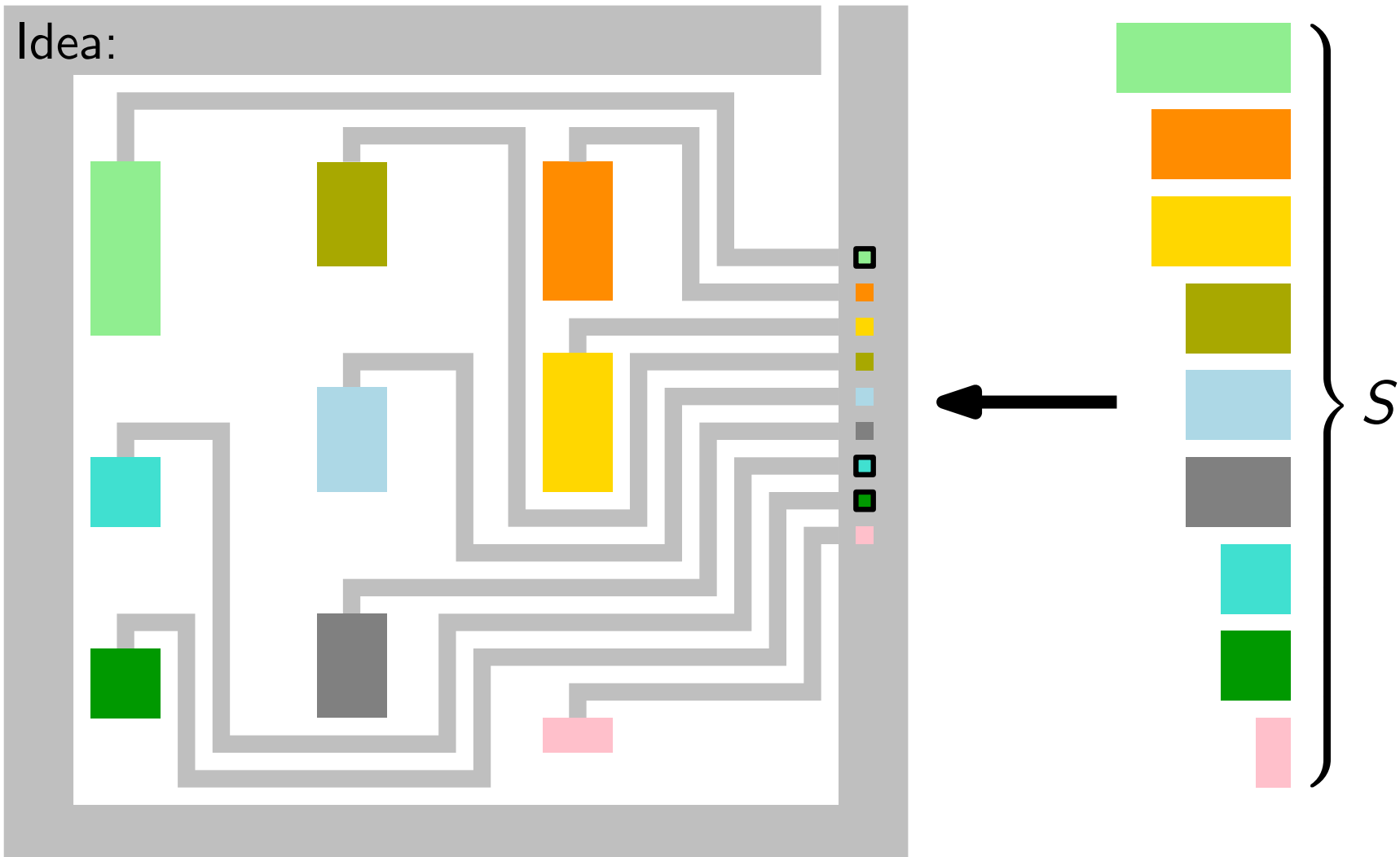
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

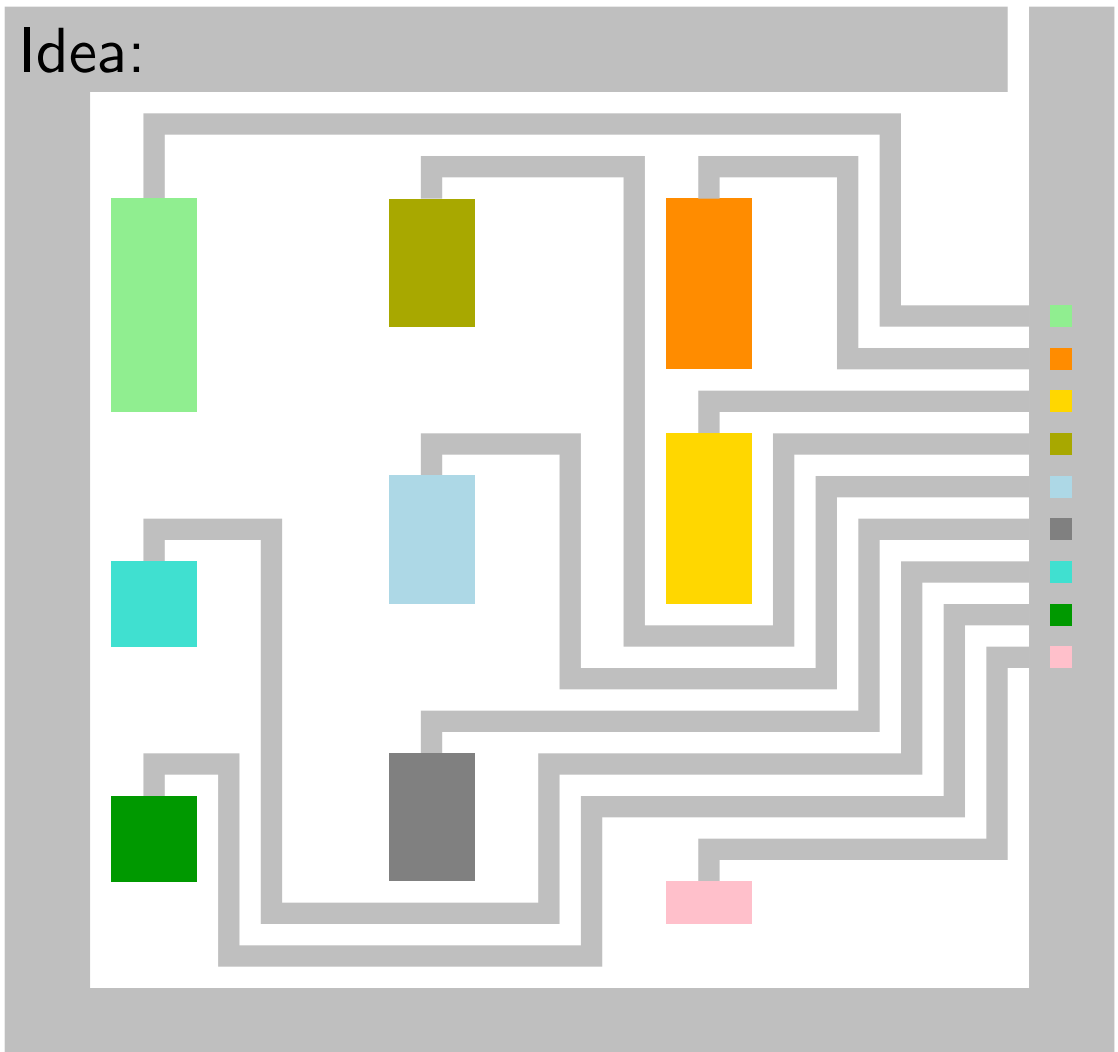


Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

Idea:

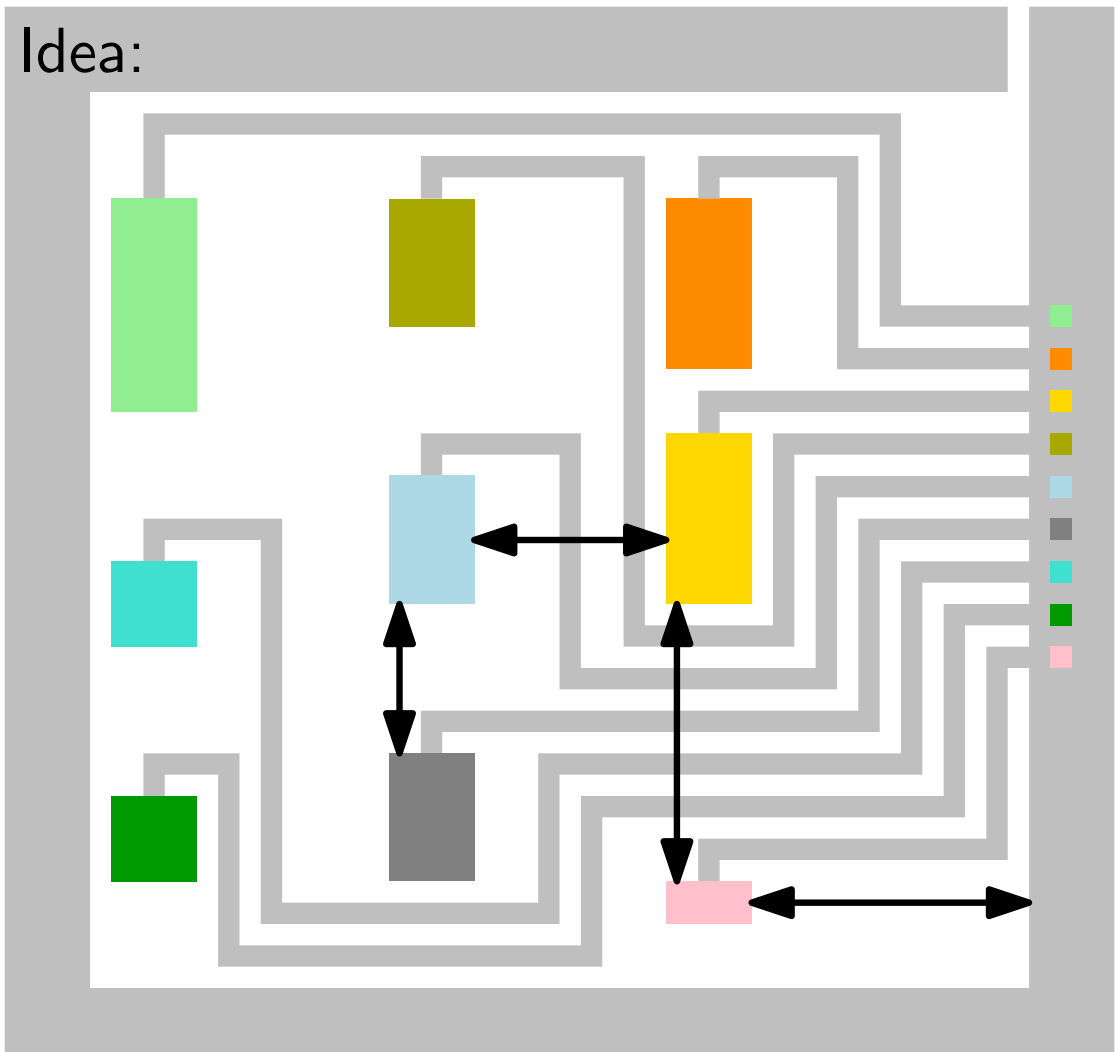


Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

Idea:



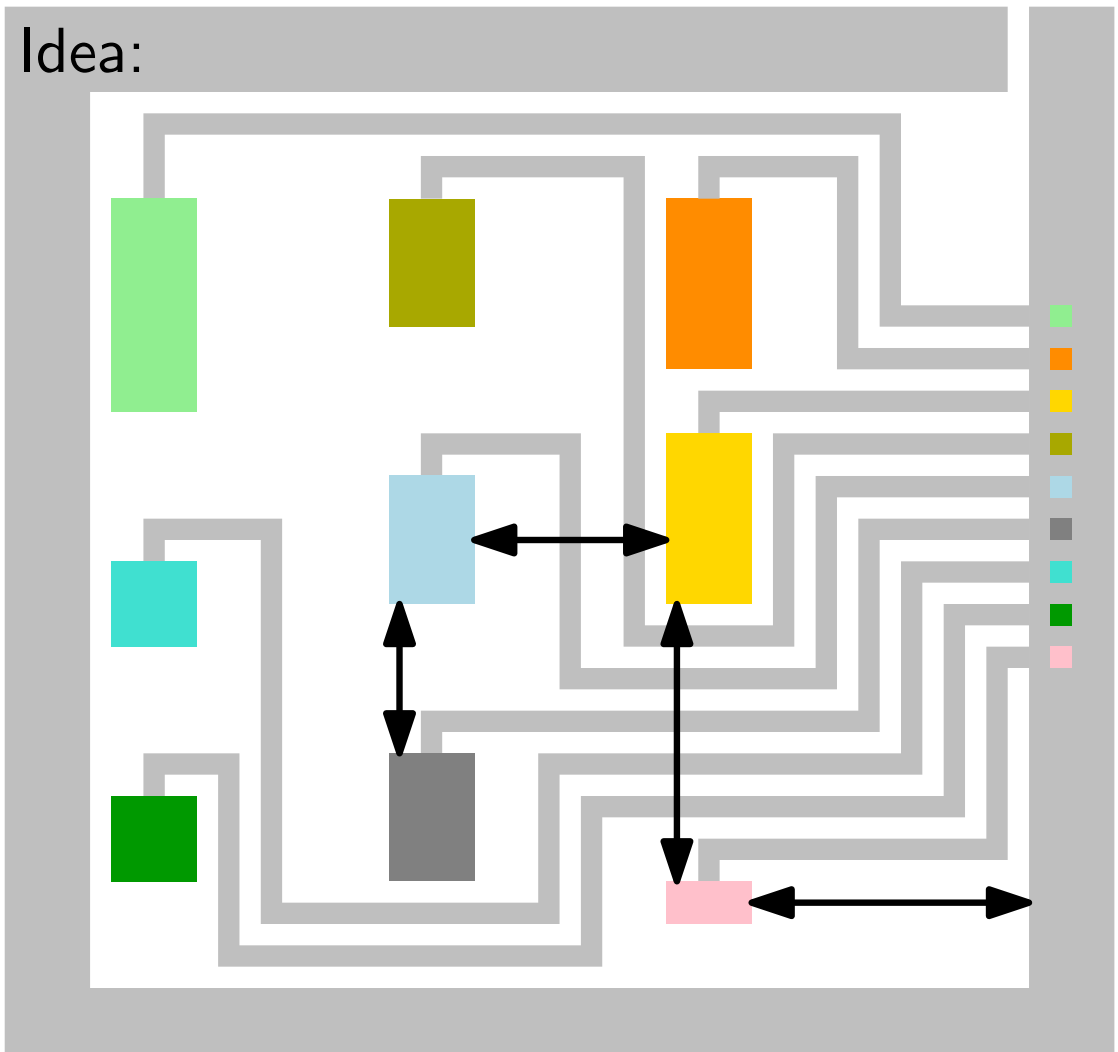
- Distance always $2 \cdot 3m$

Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

Idea:

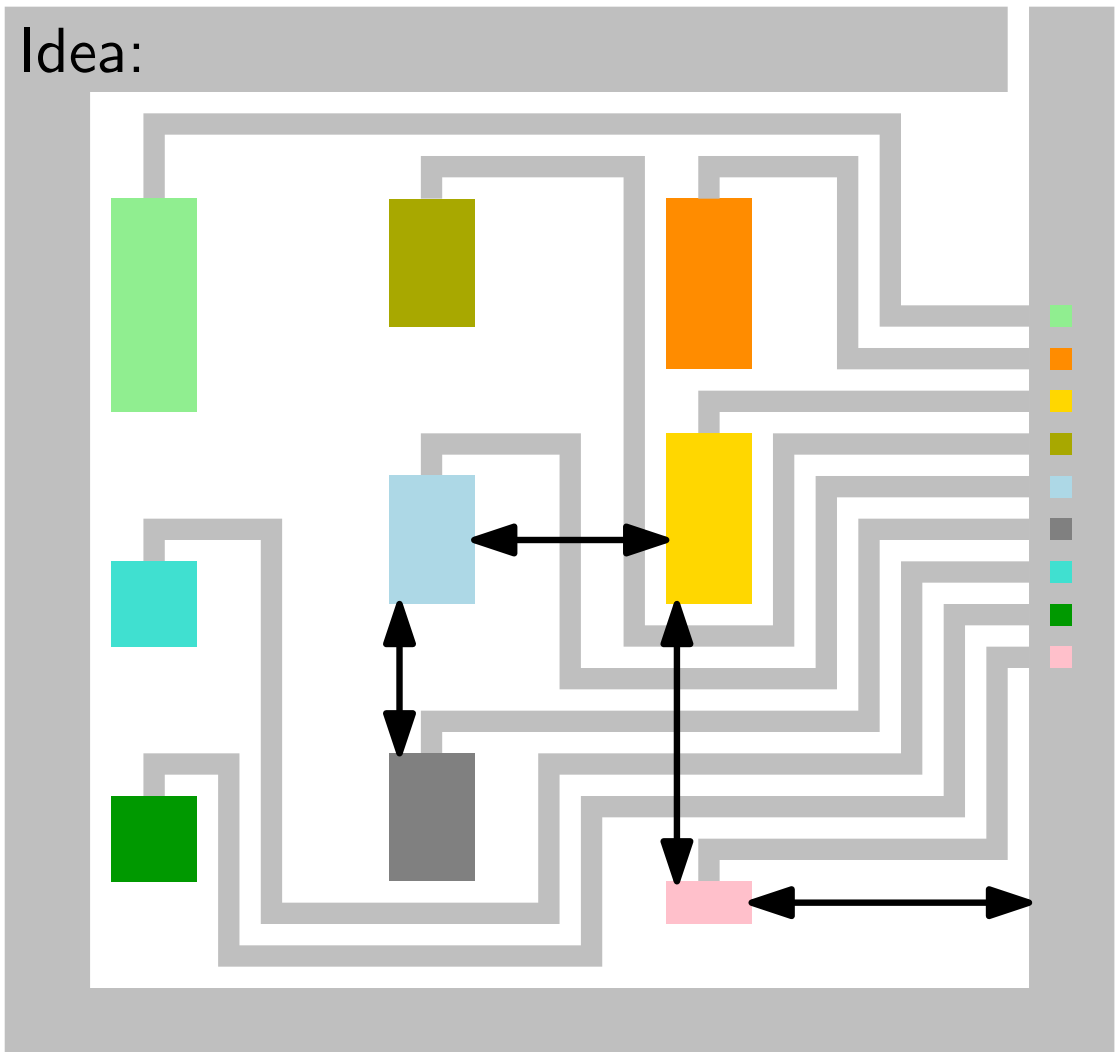


- Distance always $2 \cdot 3m$
- Blocks:

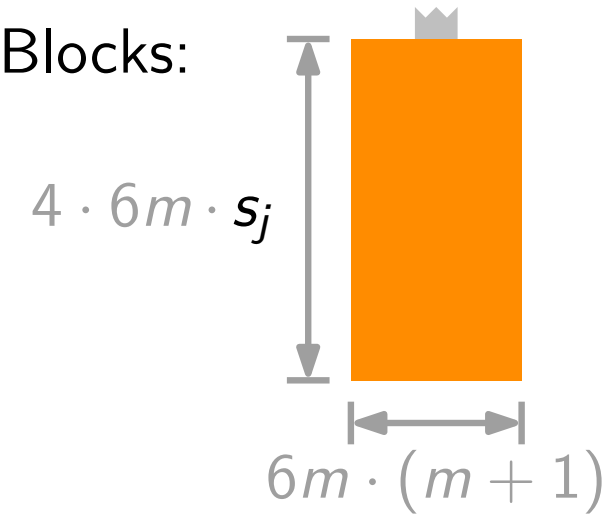
Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



- Distance always $2 \cdot 3m$
- Blocks:

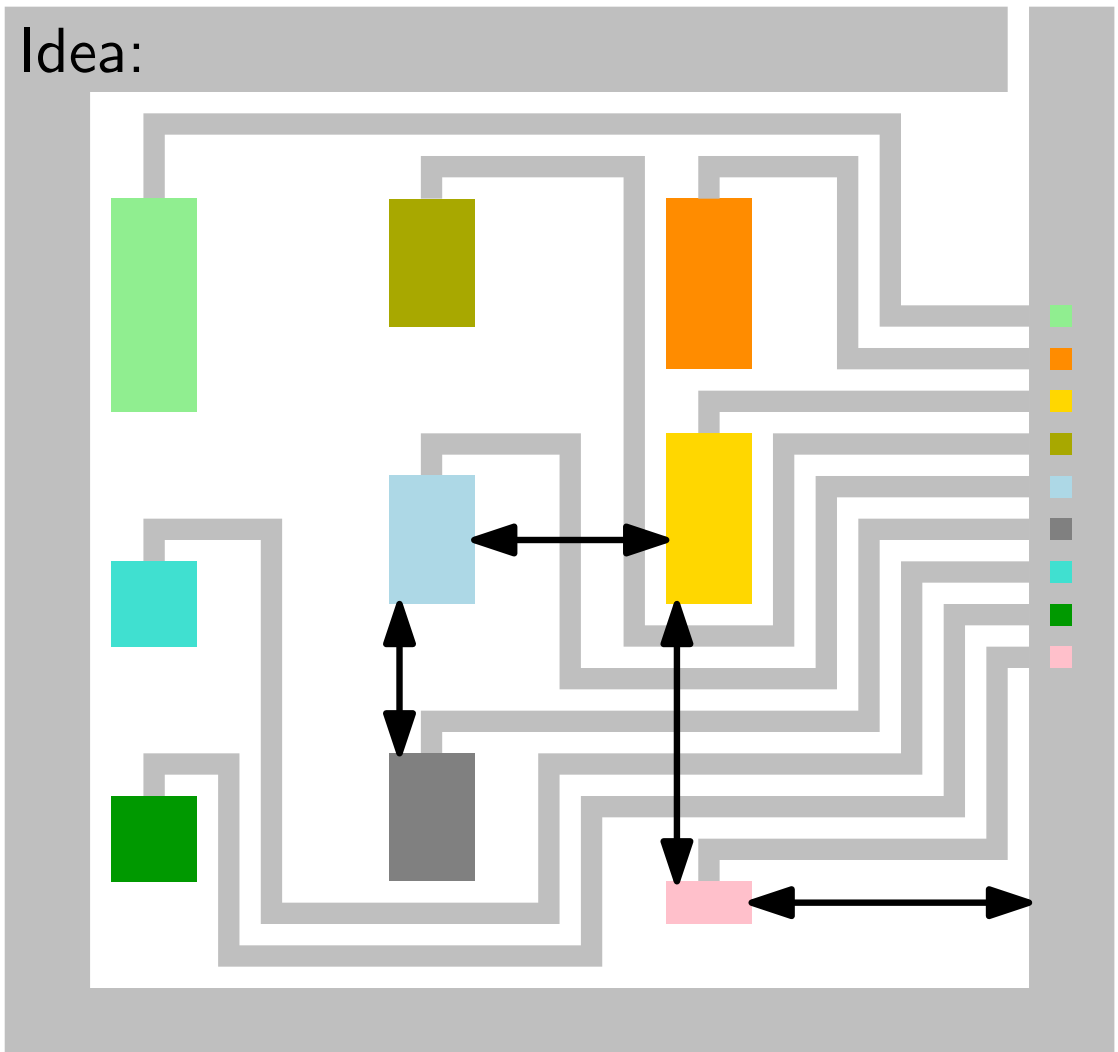


Reduction (Sketch)

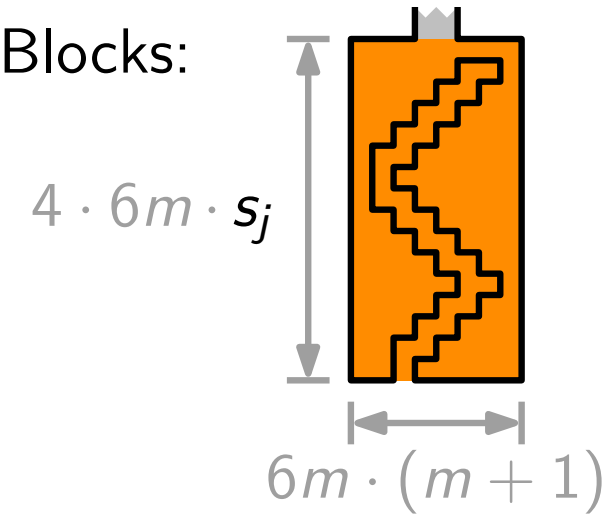
Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

Idea:



- Distance always $2 \cdot 3m$
- Blocks:

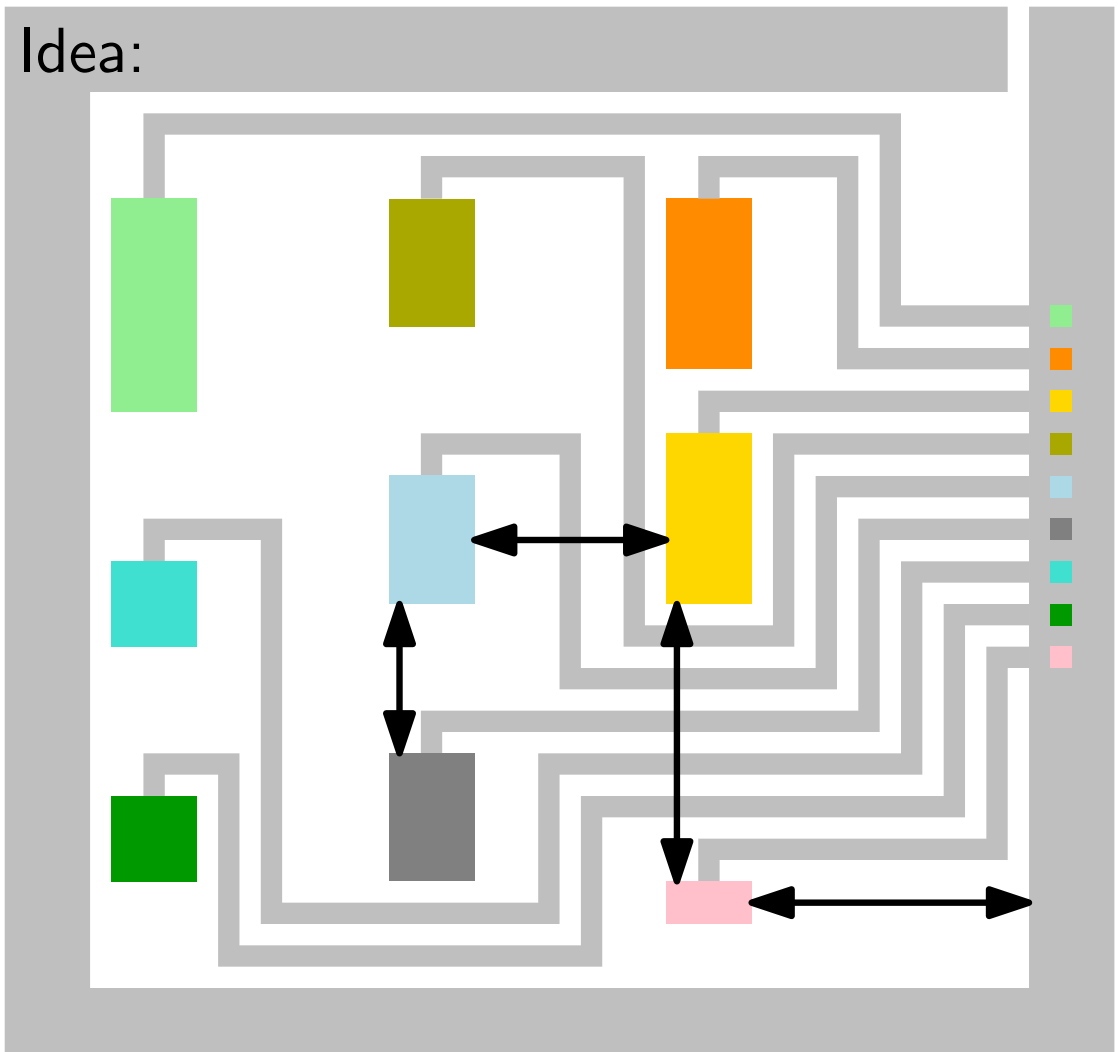


Reduction (Sketch)

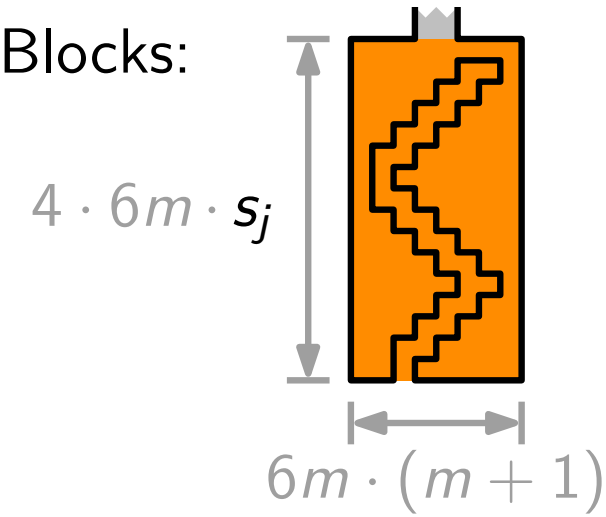
Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

Idea:



- Distance always $2 \cdot 3m$
- Blocks:



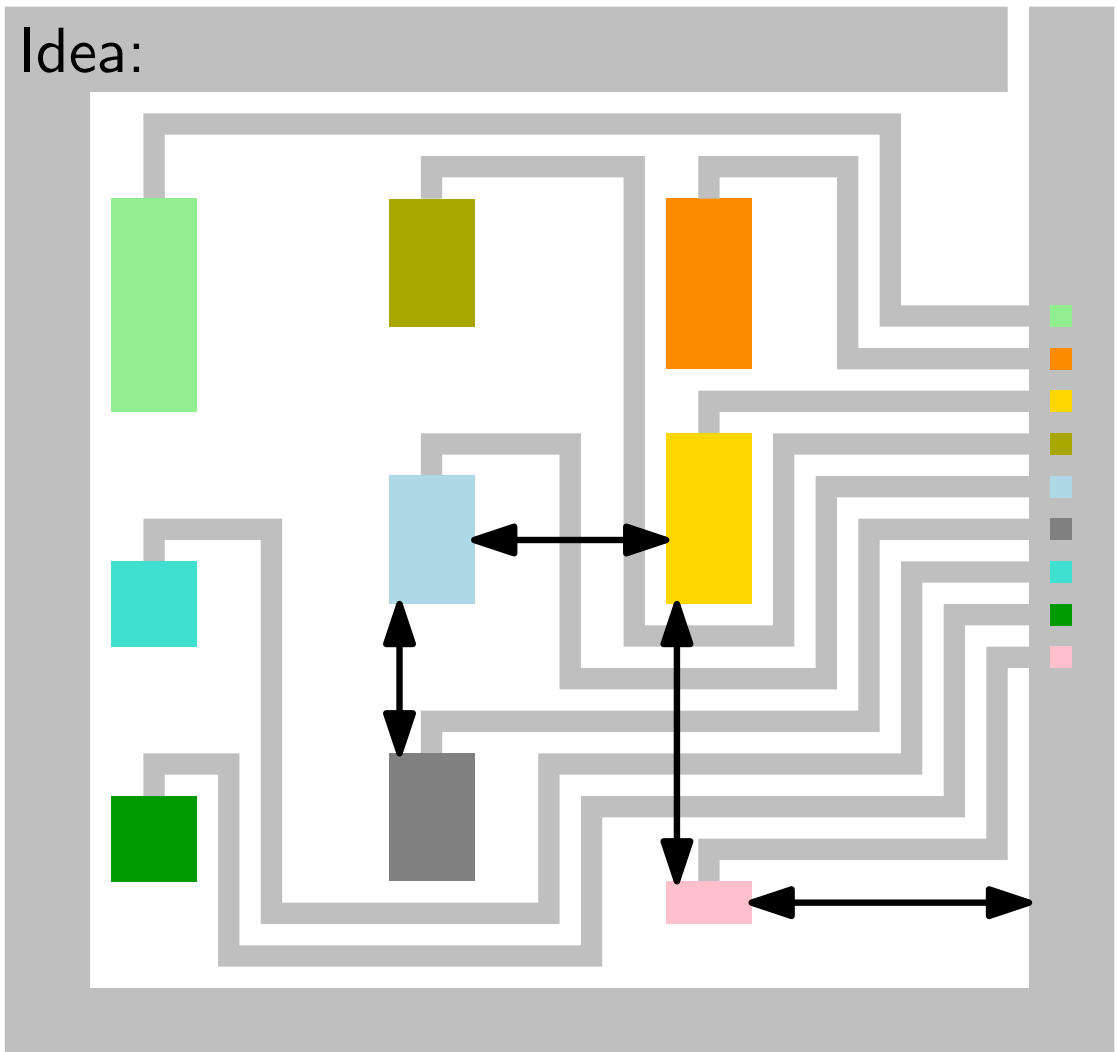
- Walls:

Reduction (Sketch)

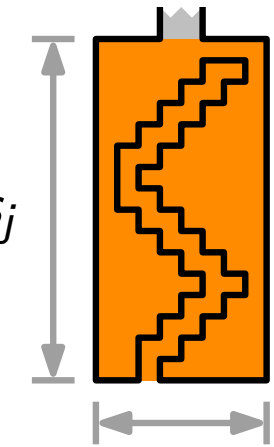
Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

Idea:



- Distance always $2 \cdot 3m$
- Blocks:



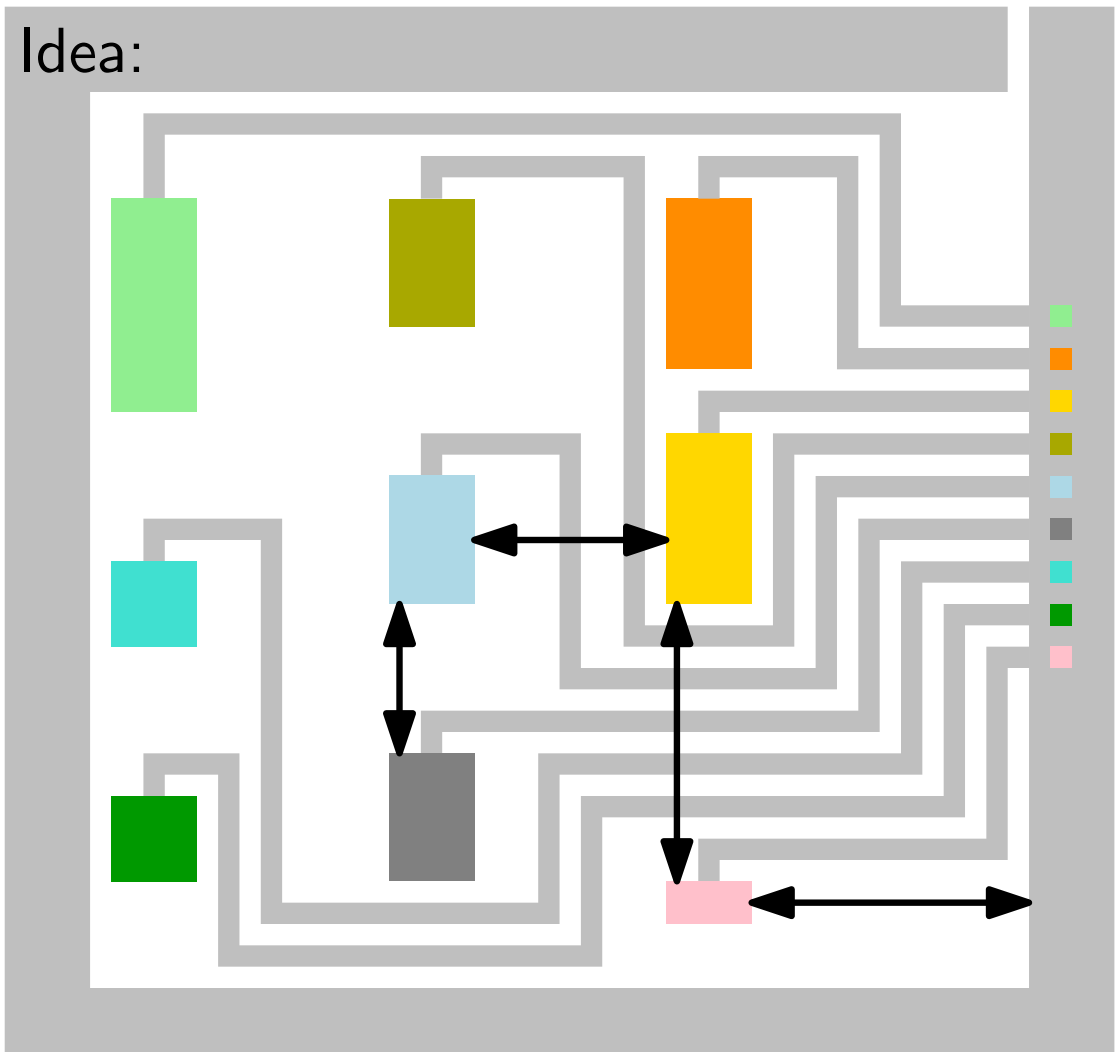
- Walls:

Reduction (Sketch)

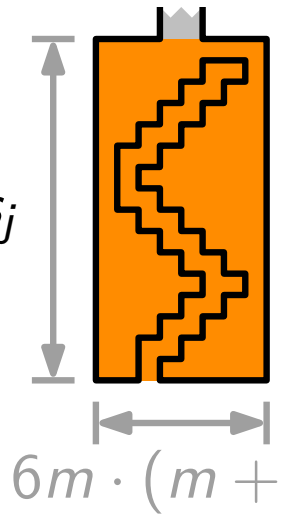
Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$

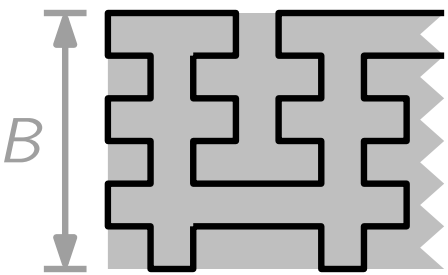
Idea:



- Distance always $2 \cdot 3m$
- Blocks:



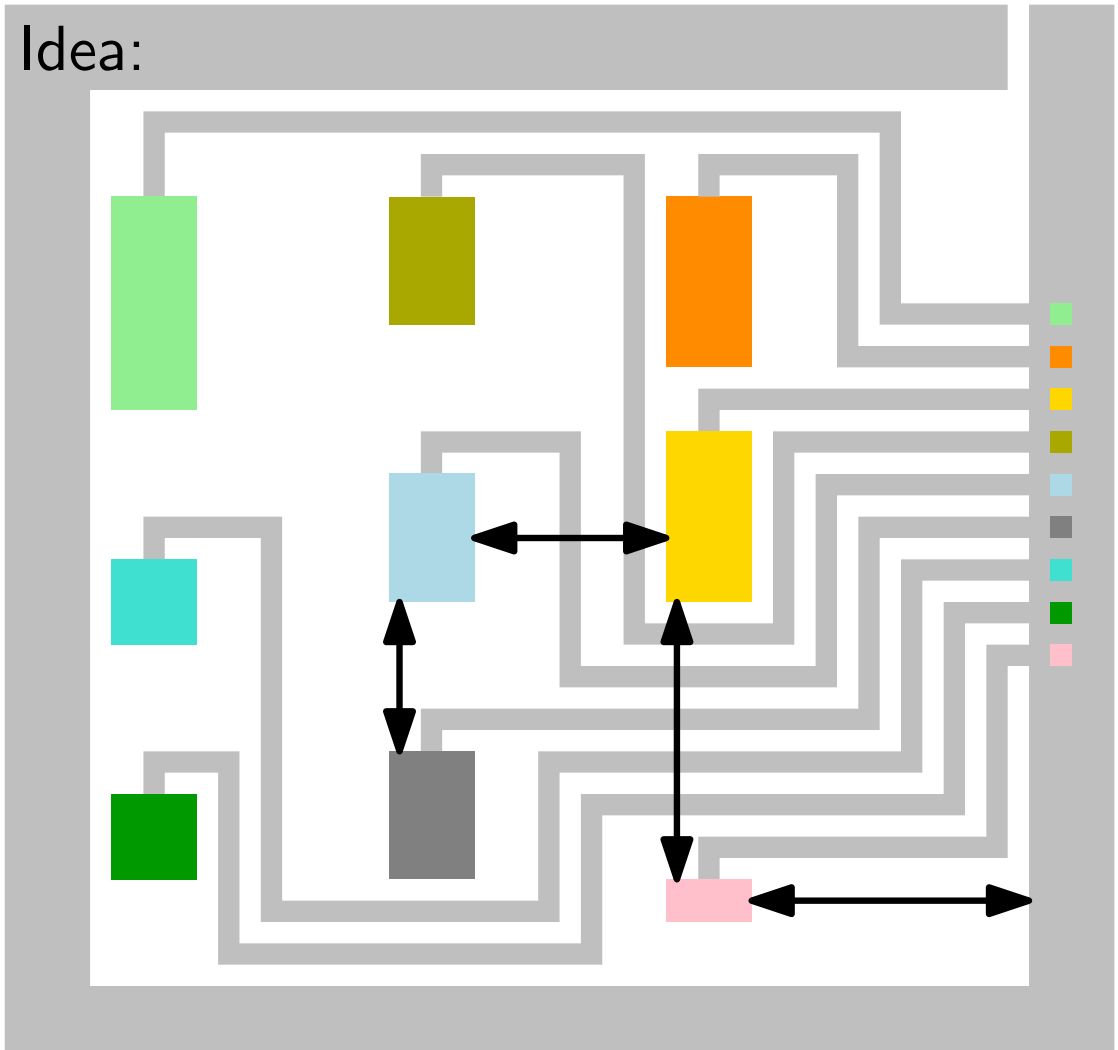
- Walls:



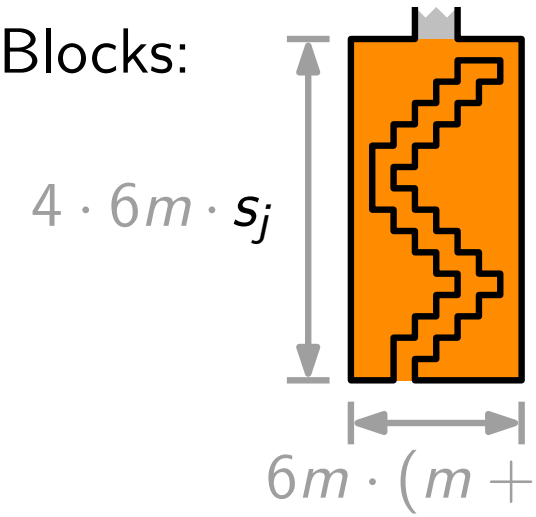
Reduction (Sketch)

Given an instance $S = (s_1, \ldots, s_{3m})$ of 3-Partition,
compute in poly(m) time $K > 0$ and an instance σ of MinBBLRSequence s.t.

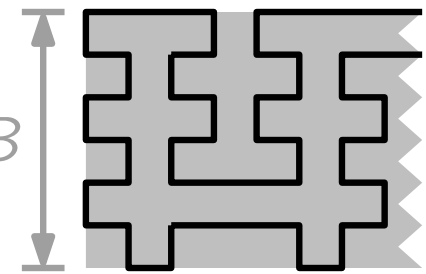
$$\text{area}(\sigma) < K \;\Leftrightarrow\; S \text{ is yes-instance}$$



- Distance always $2 \cdot 3m$
- Blocks:



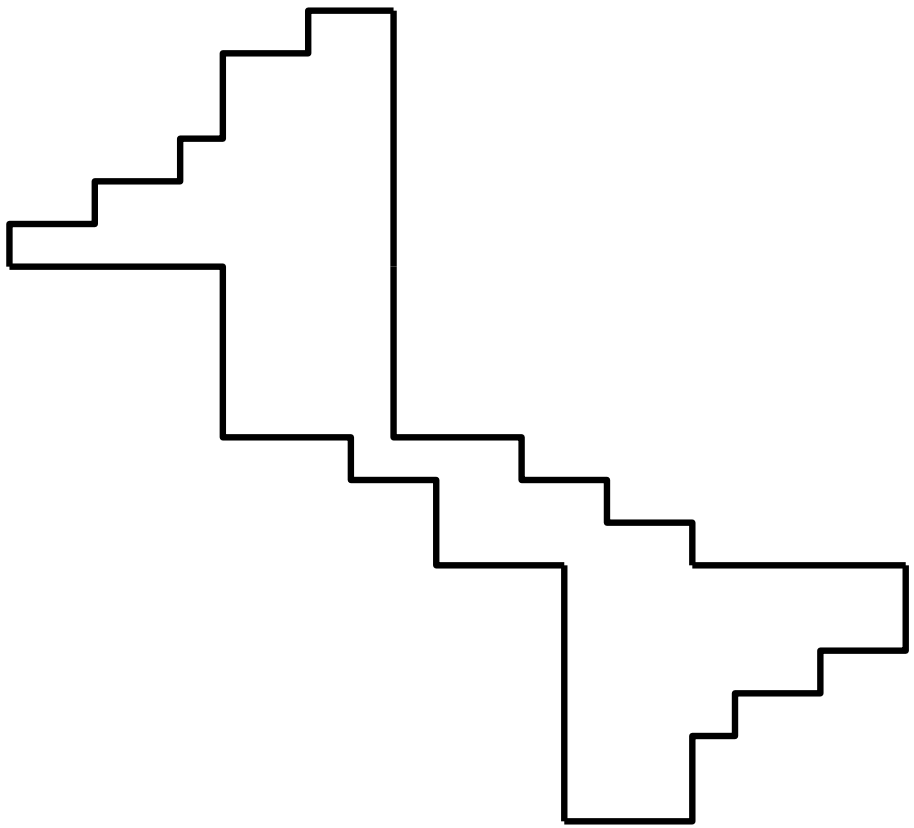
- Walls:



- reduction $\in$ poly(m, B)

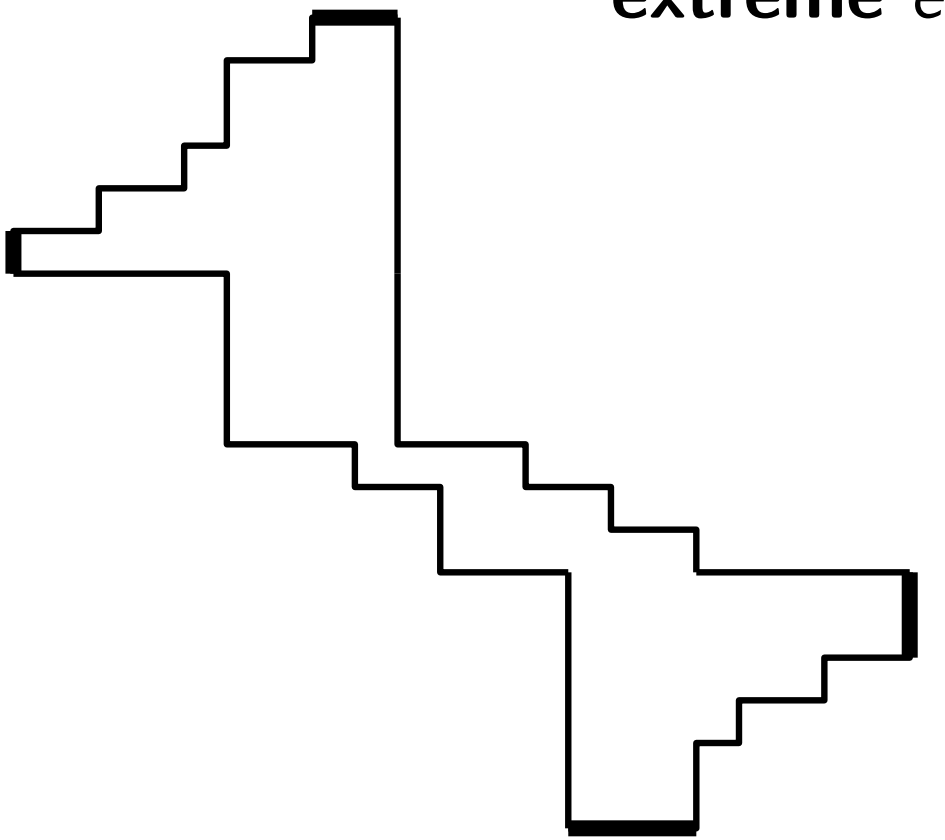
Minimizing the Area of xy-monotone Polygons

Polygon canonical if . . .



Polygon canonical if . . .

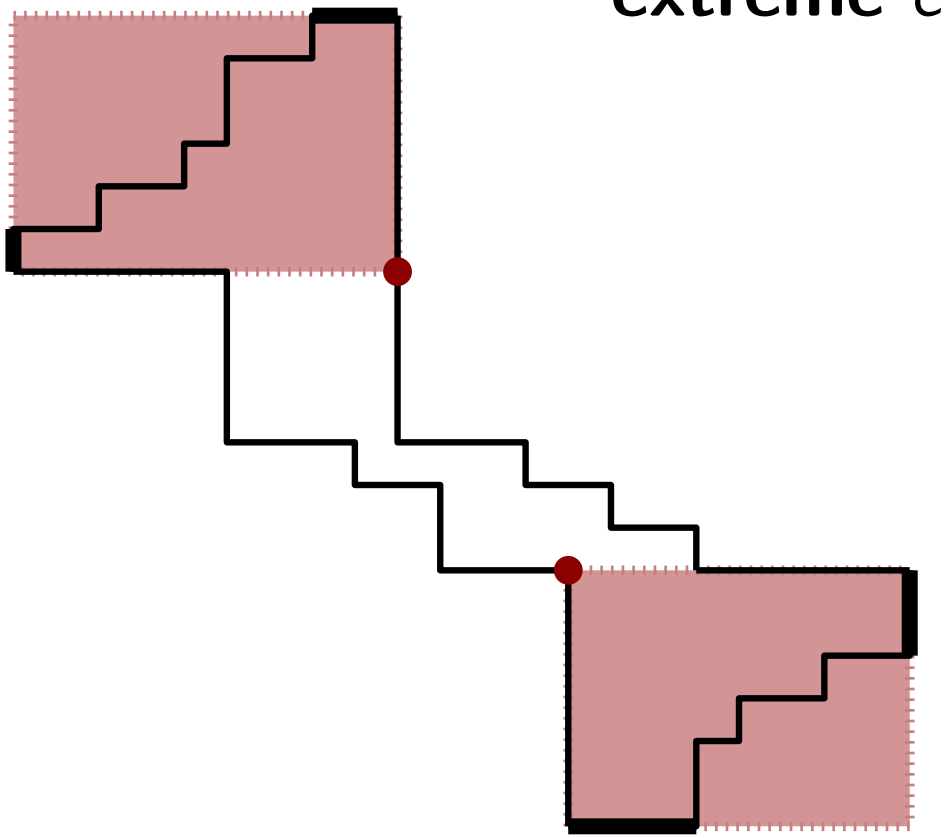
extreme *edges*



Polygon canonical if . . .

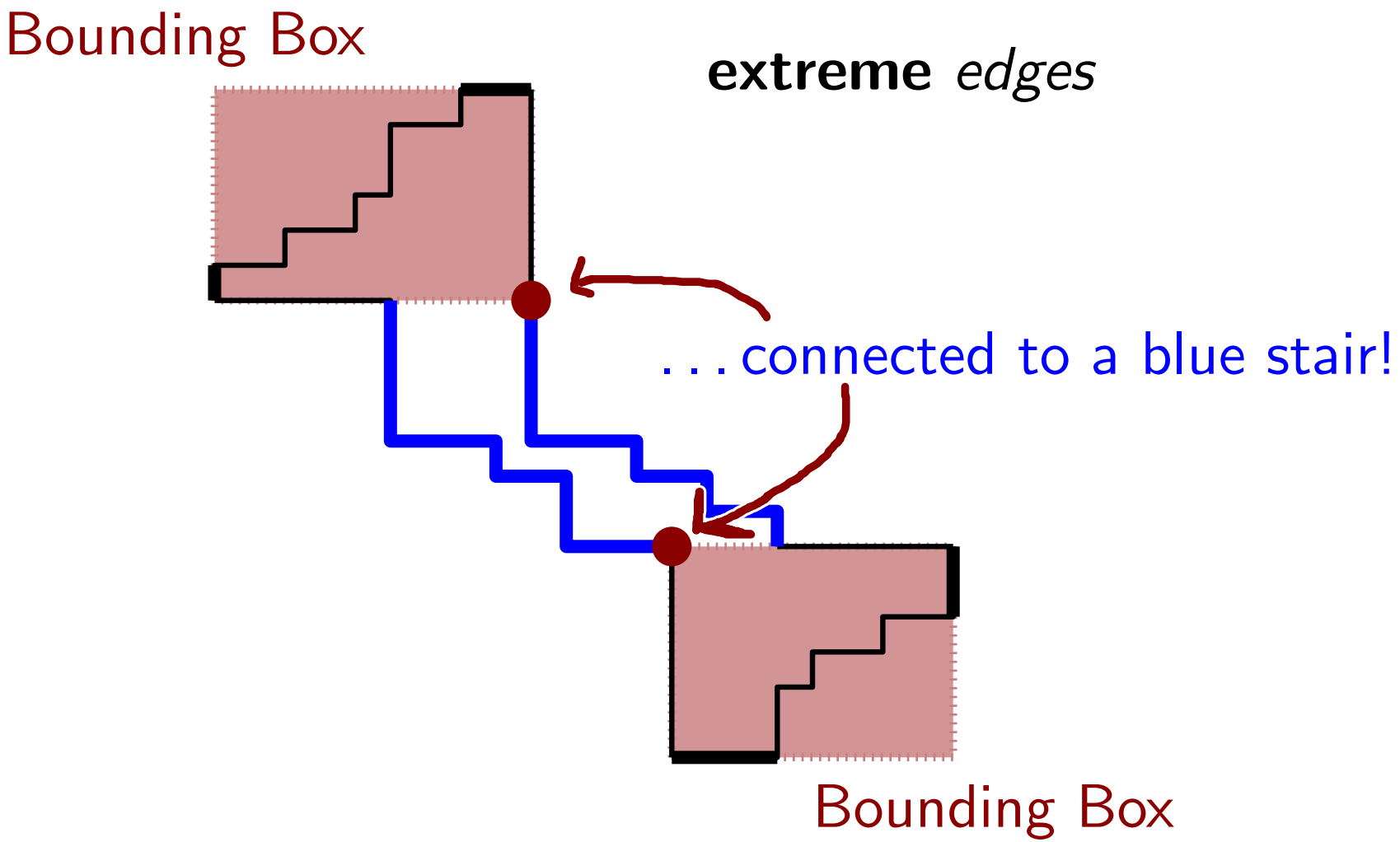
Bounding Box

extreme *edges*

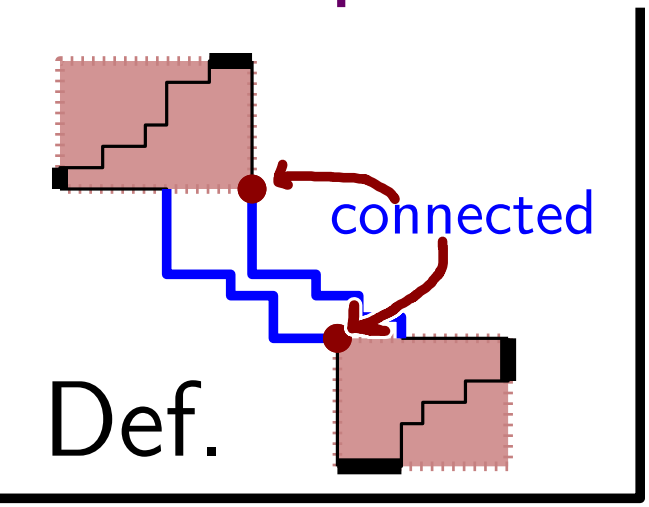


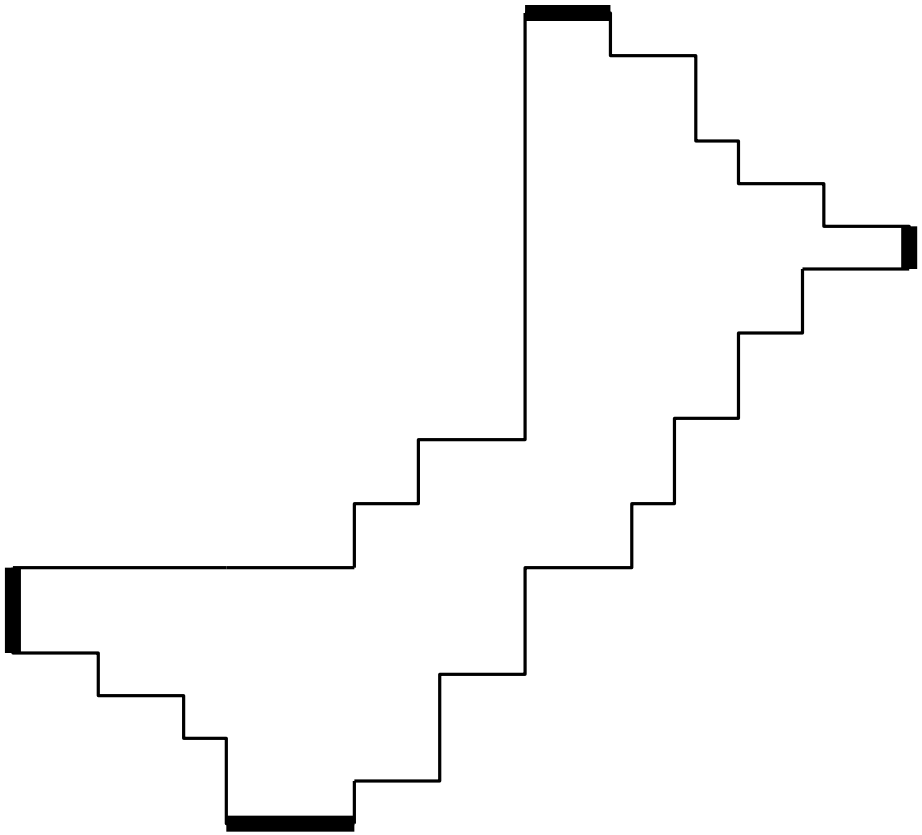
Bounding Box

Polygon canonical if . . .

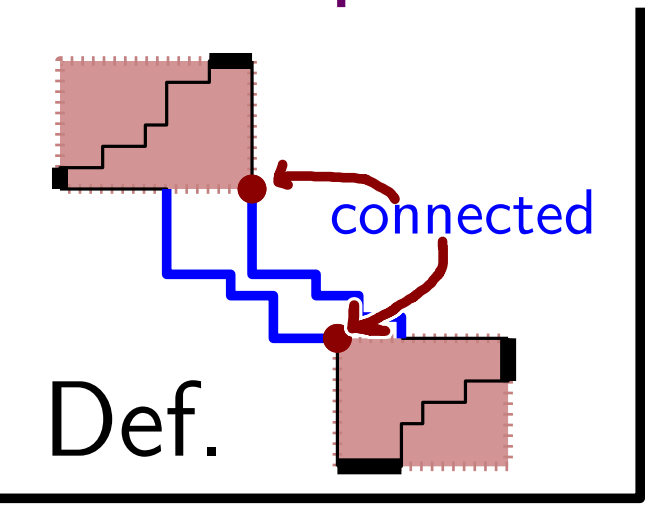


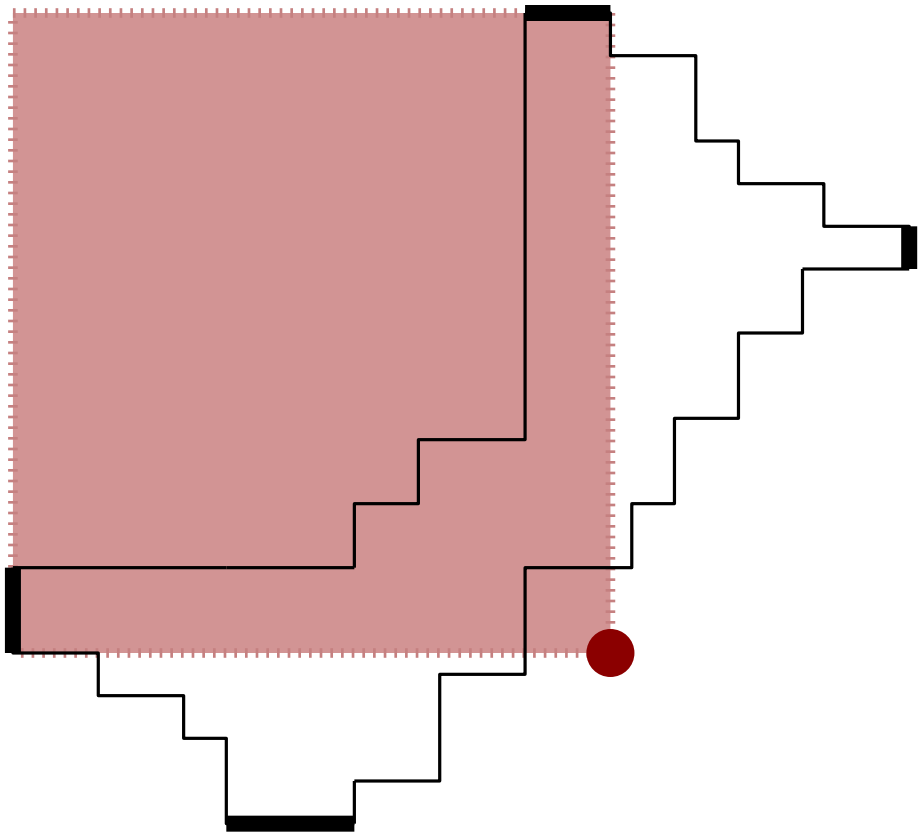
Example: Canonical Polygon



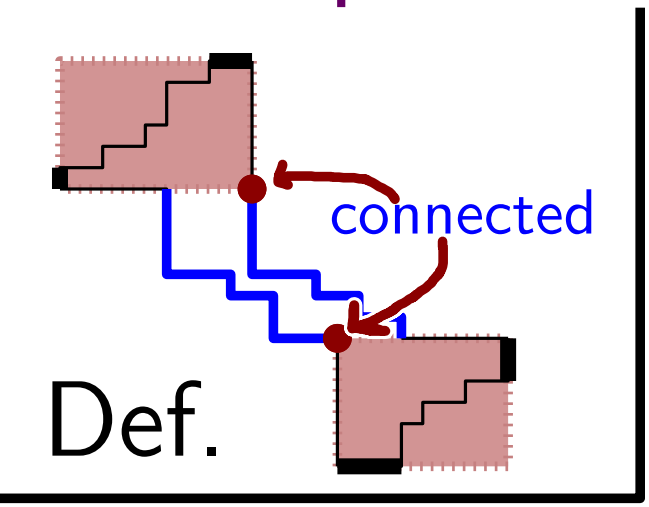


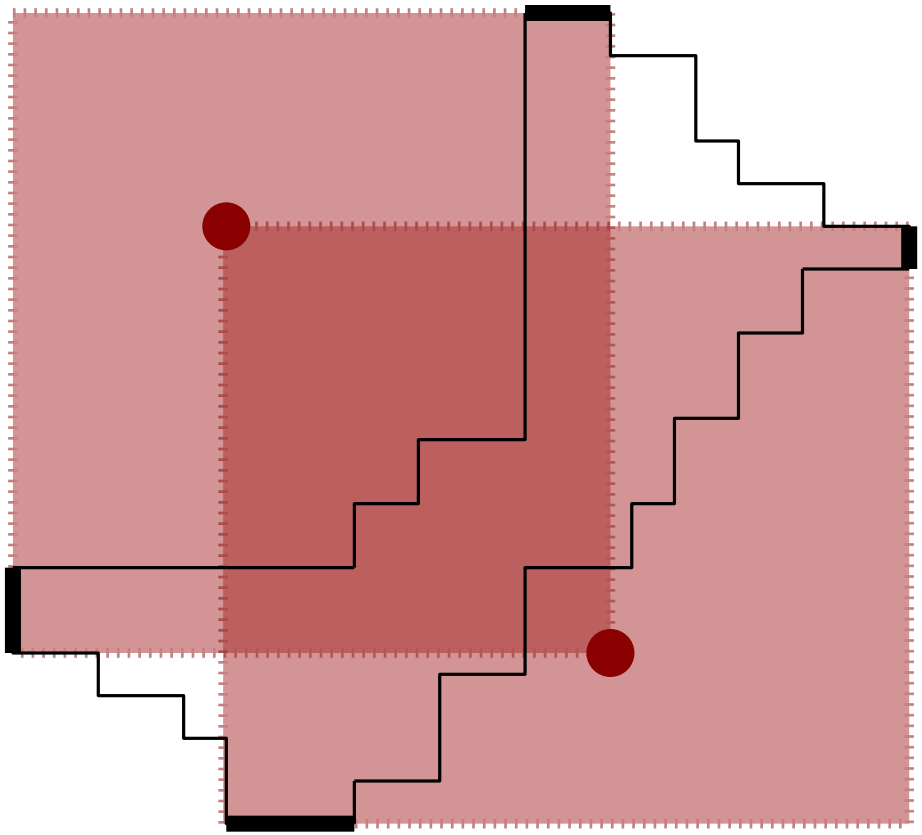
Example: Canonical Polygon



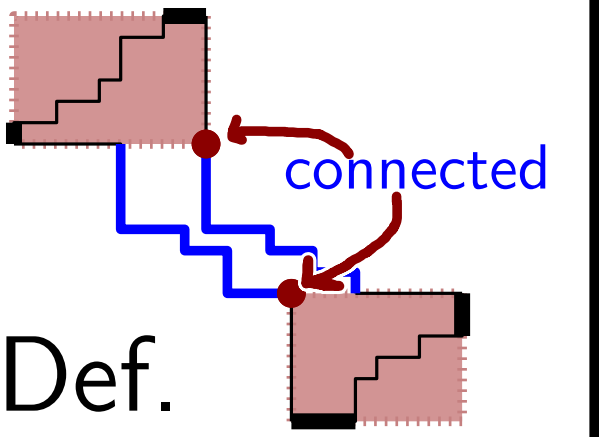


Example: Canonical Polygon

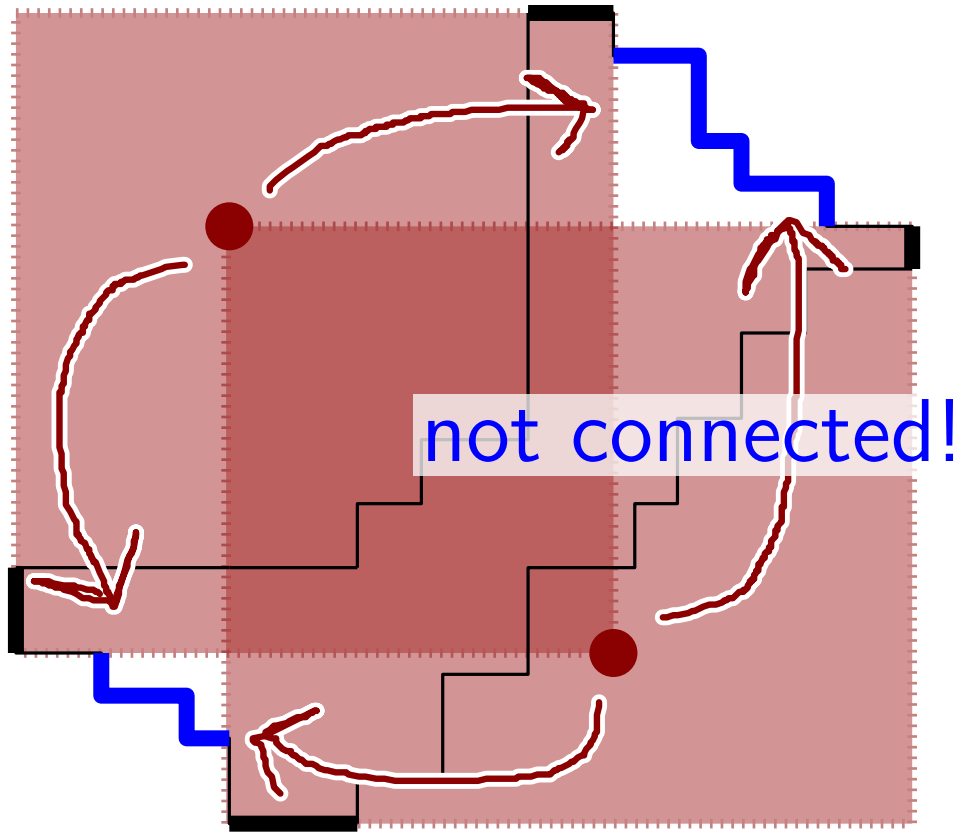




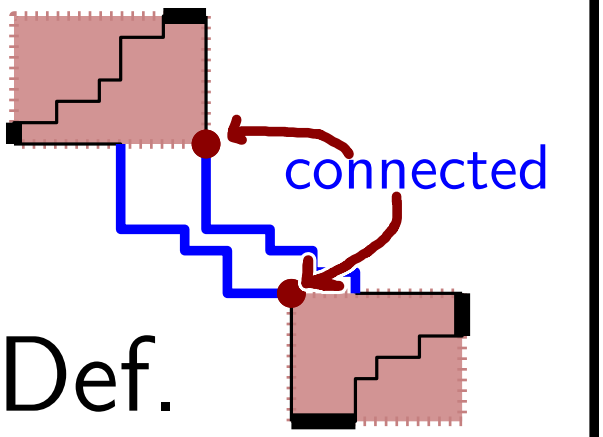
Example: Canonical Polygon



Def.

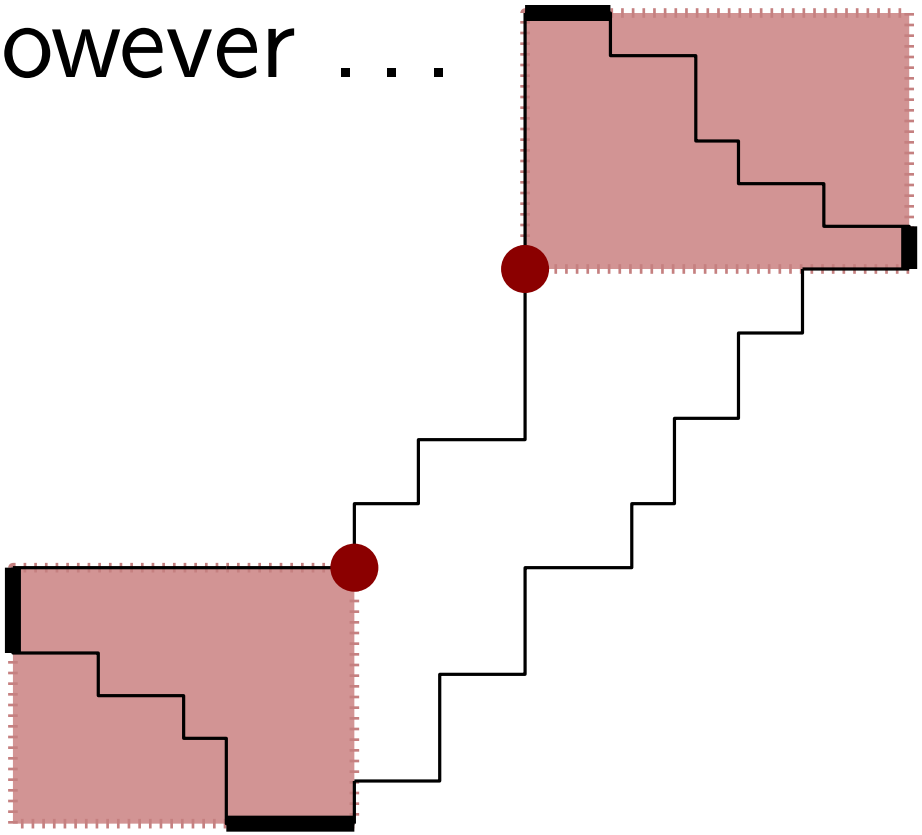


Example: Canonical Polygon

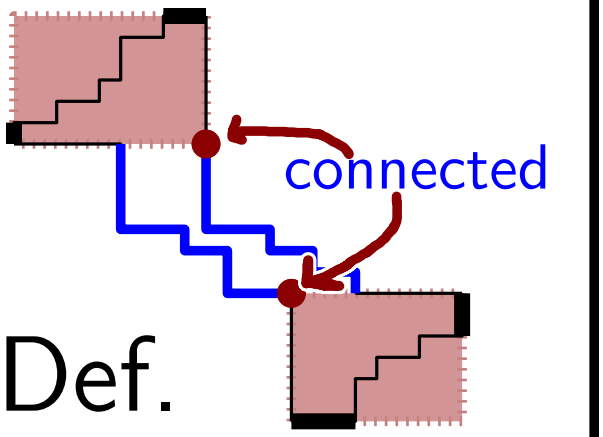


Def.

However . . .

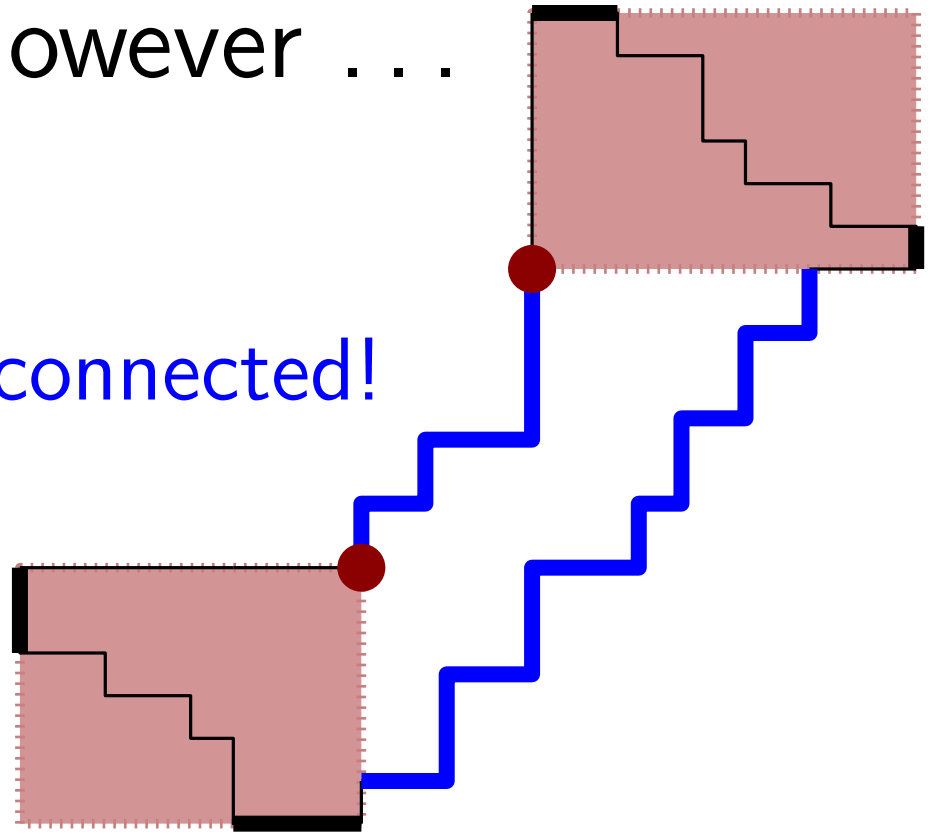


Example: Canonical Polygon

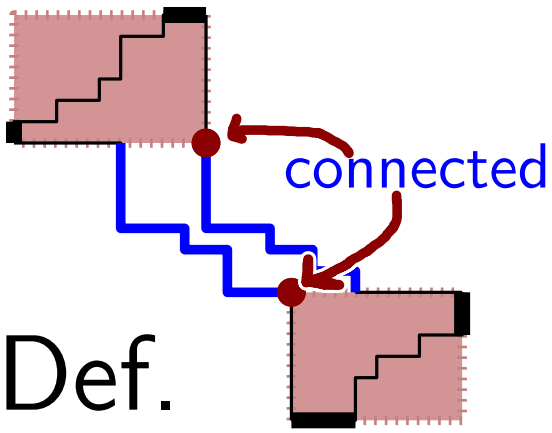


Def.

However . . .



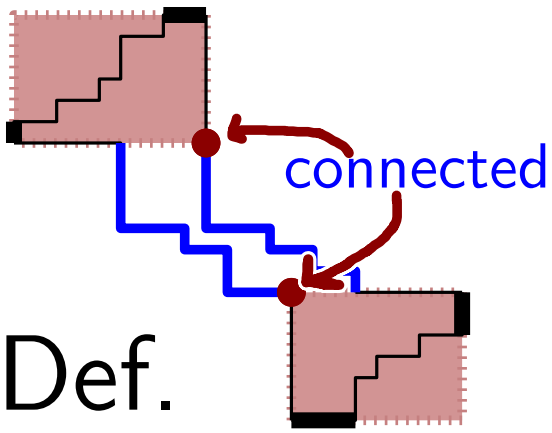
. . . is canonical.



Def.

Lemma:

There is an optimum canonical polygon!



Def.

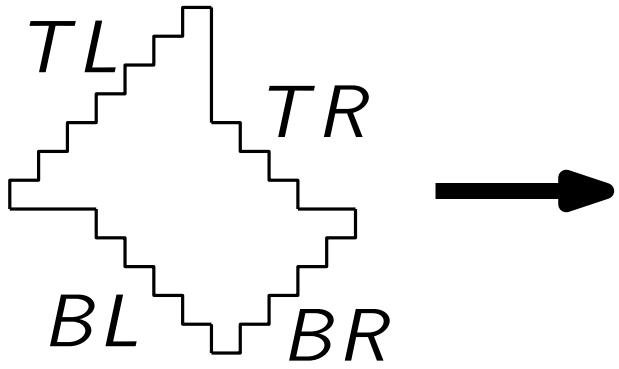
Lemma:

There is an optimum canonical polygon!

How to find it?

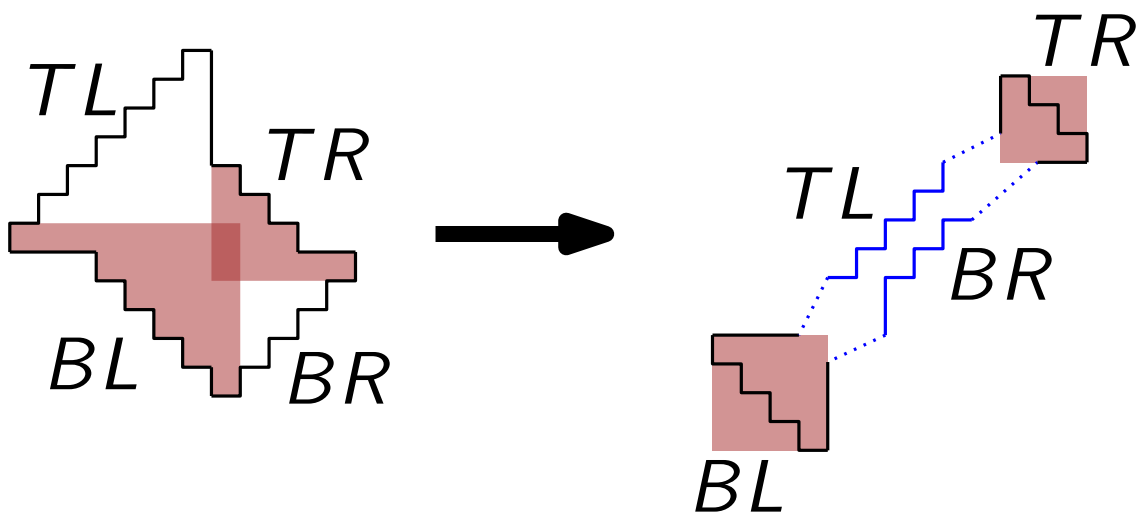
Computing Optimum Canonical Polygon

Given angle sequence . . .



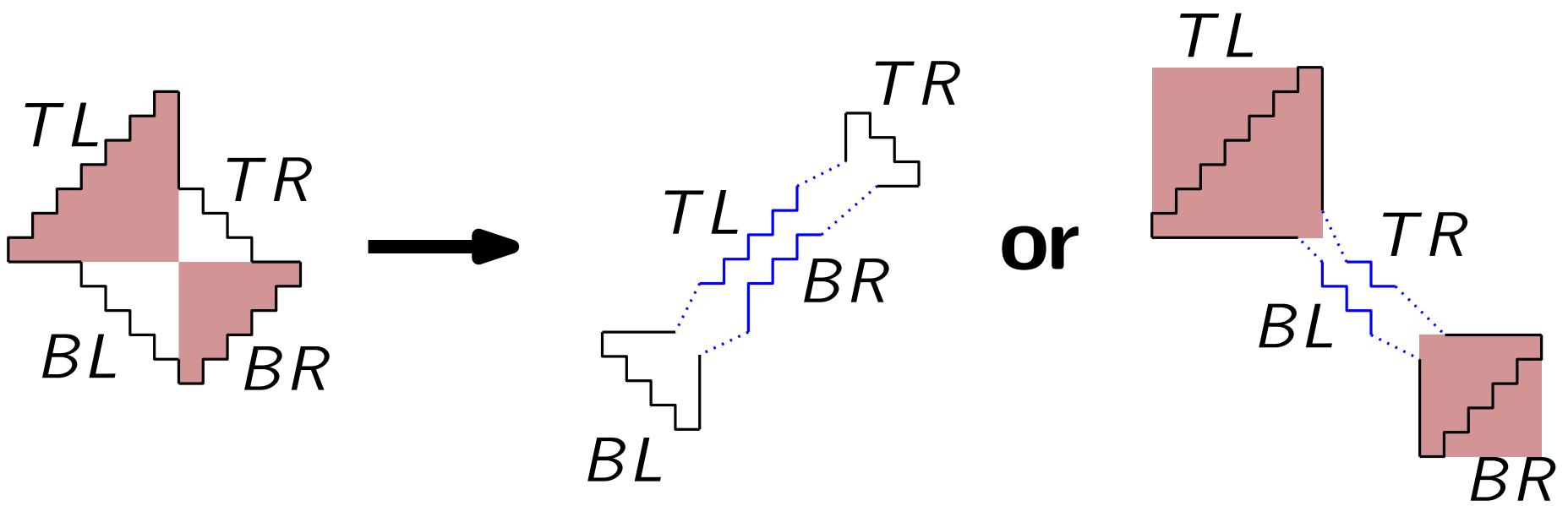
Computing Optimum Canonical Polygon

Given angle sequence . . . . . $O(1)$ candidates for OPT.



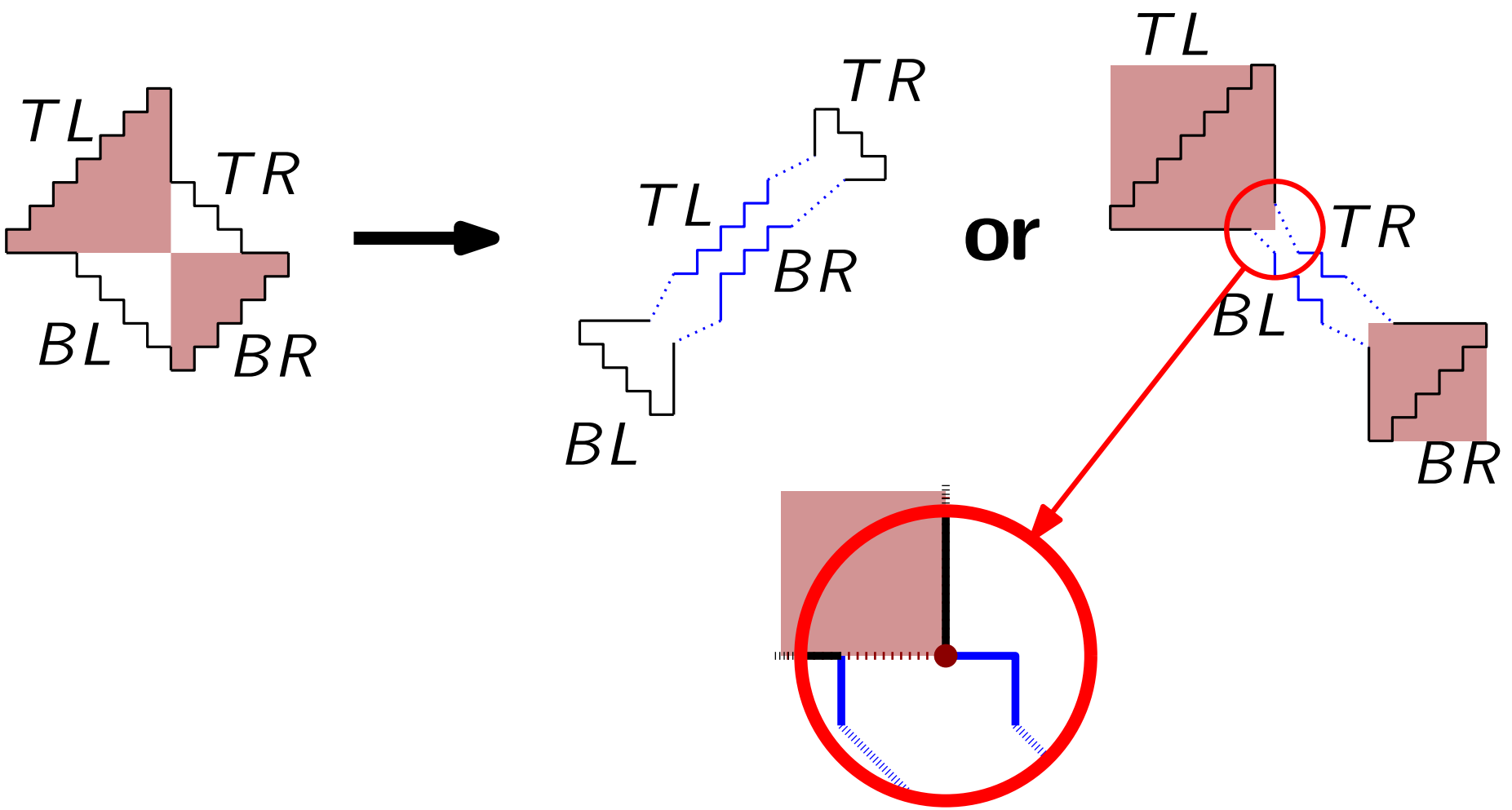
Computing Optimum Canonical Polygon

Given angle sequence . . . . . $O(1)$ candidates for OPT.



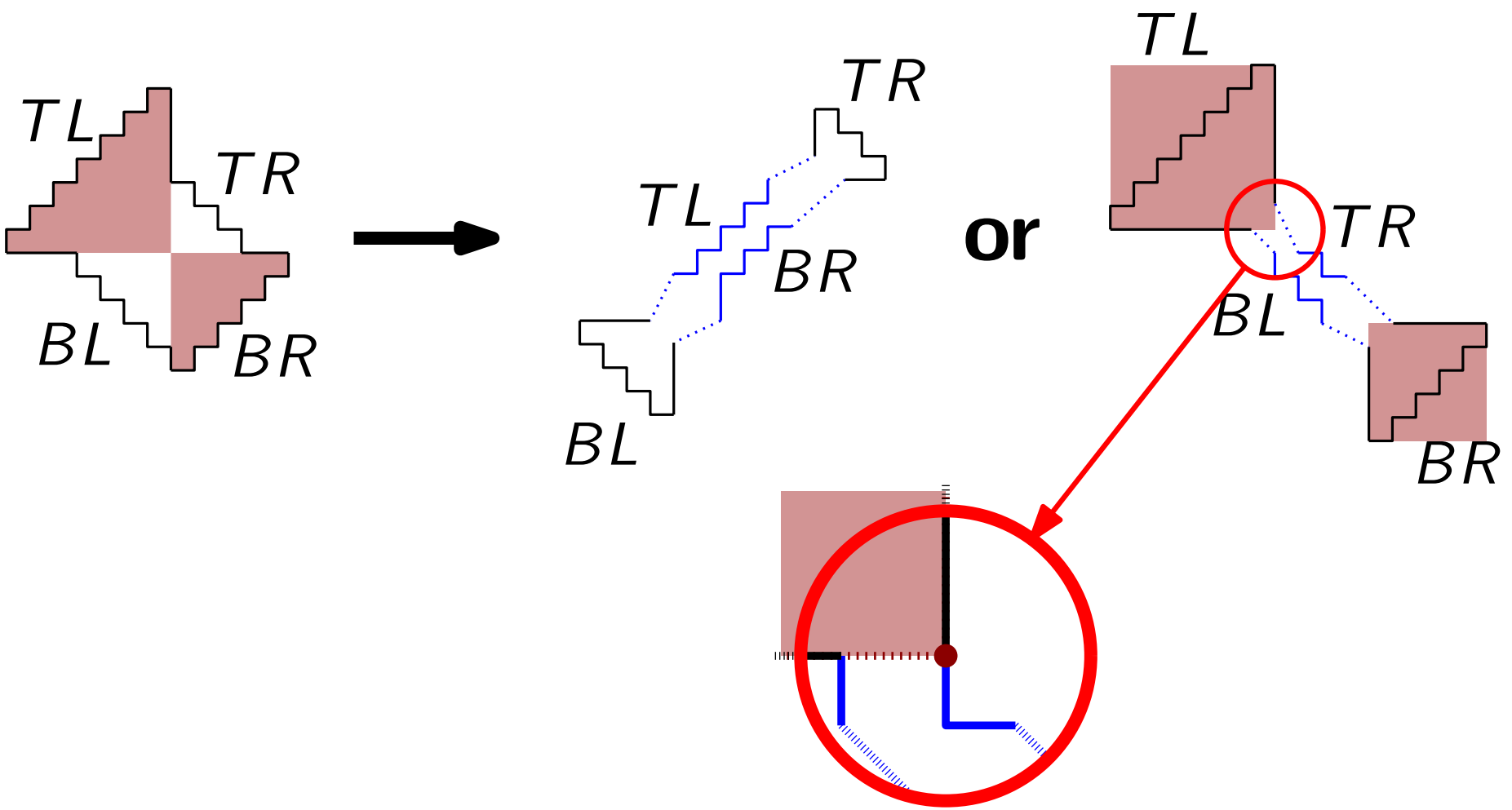
Computing Optimum Canonical Polygon

Given angle sequence . . . . . $O(1)$ candidates for OPT.



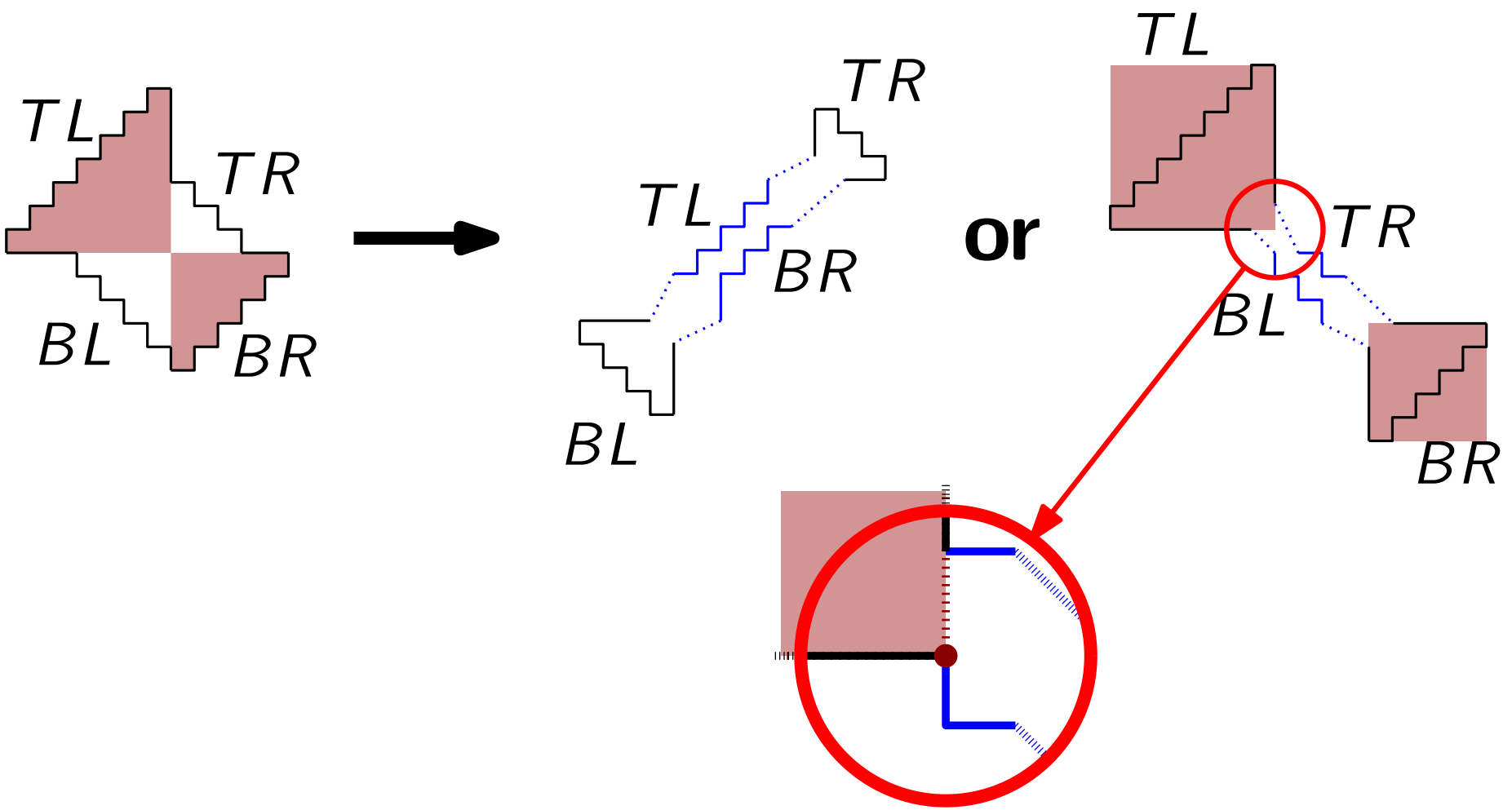
Computing Optimum Canonical Polygon

Given angle sequence . . . . . $O(1)$ candidates for OPT.



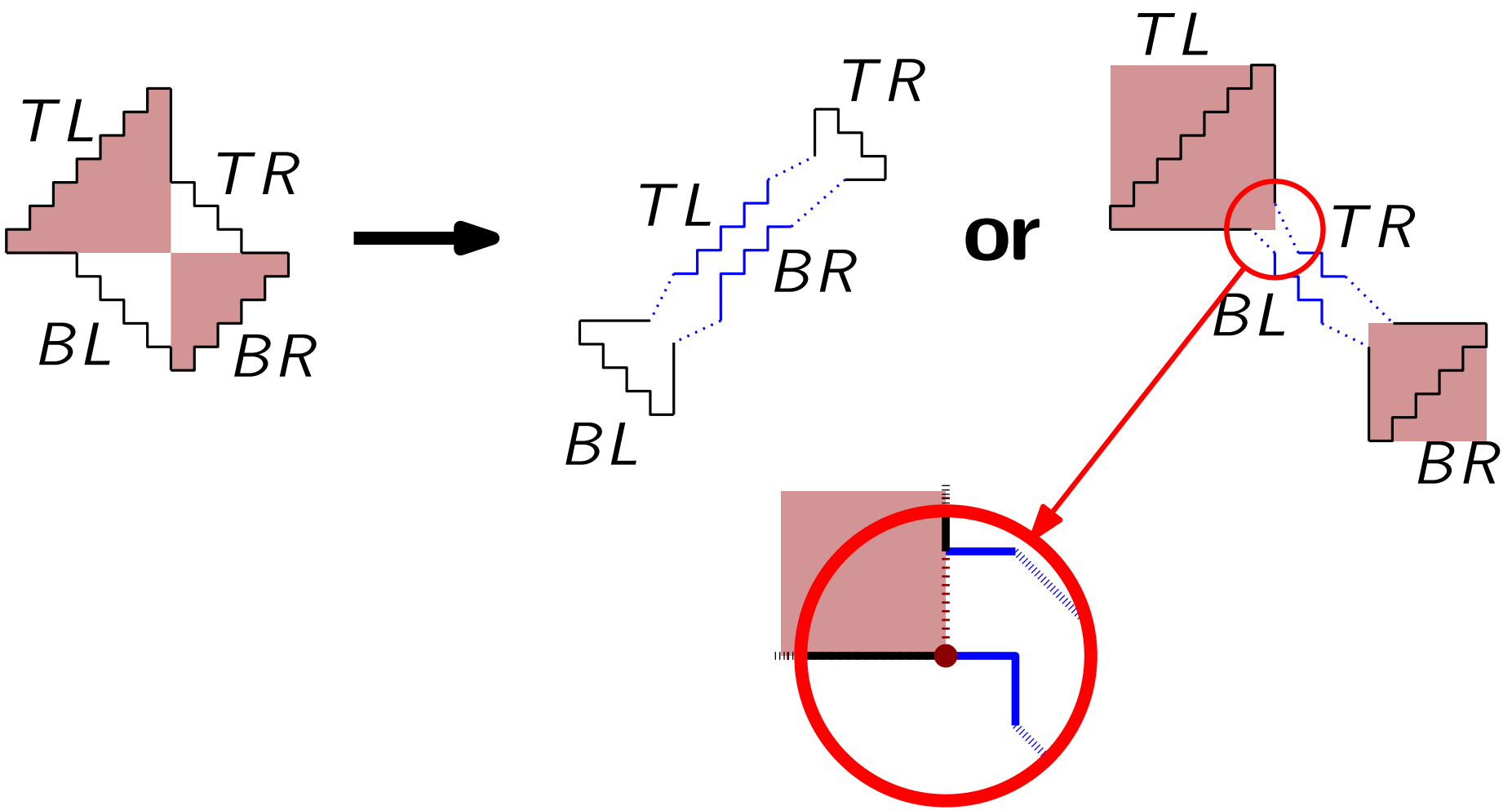
Computing Optimum Canonical Polygon

Given angle sequence . . . . . $O(1)$ candidates for OPT.



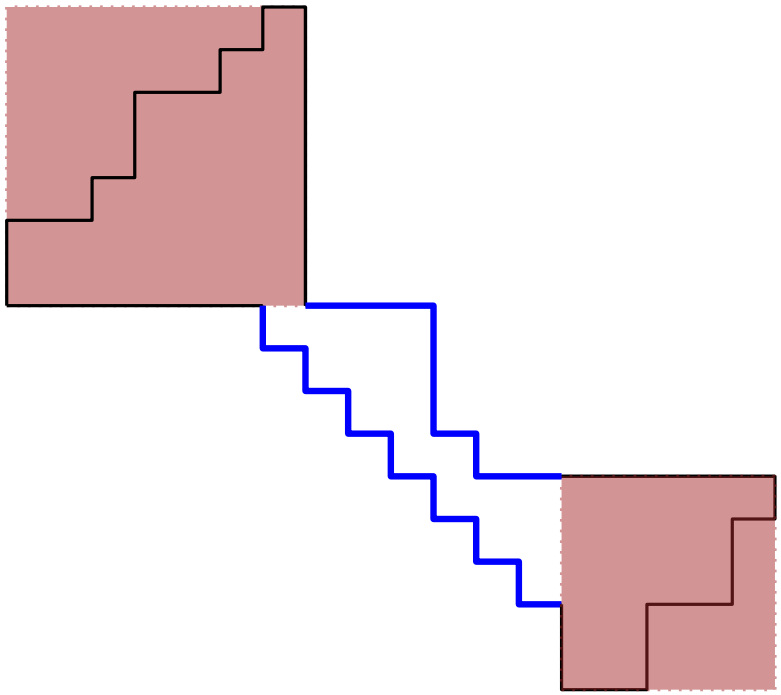
Computing Optimum Canonical Polygon

Given angle sequence . . . . . $O(1)$ candidates for OPT.



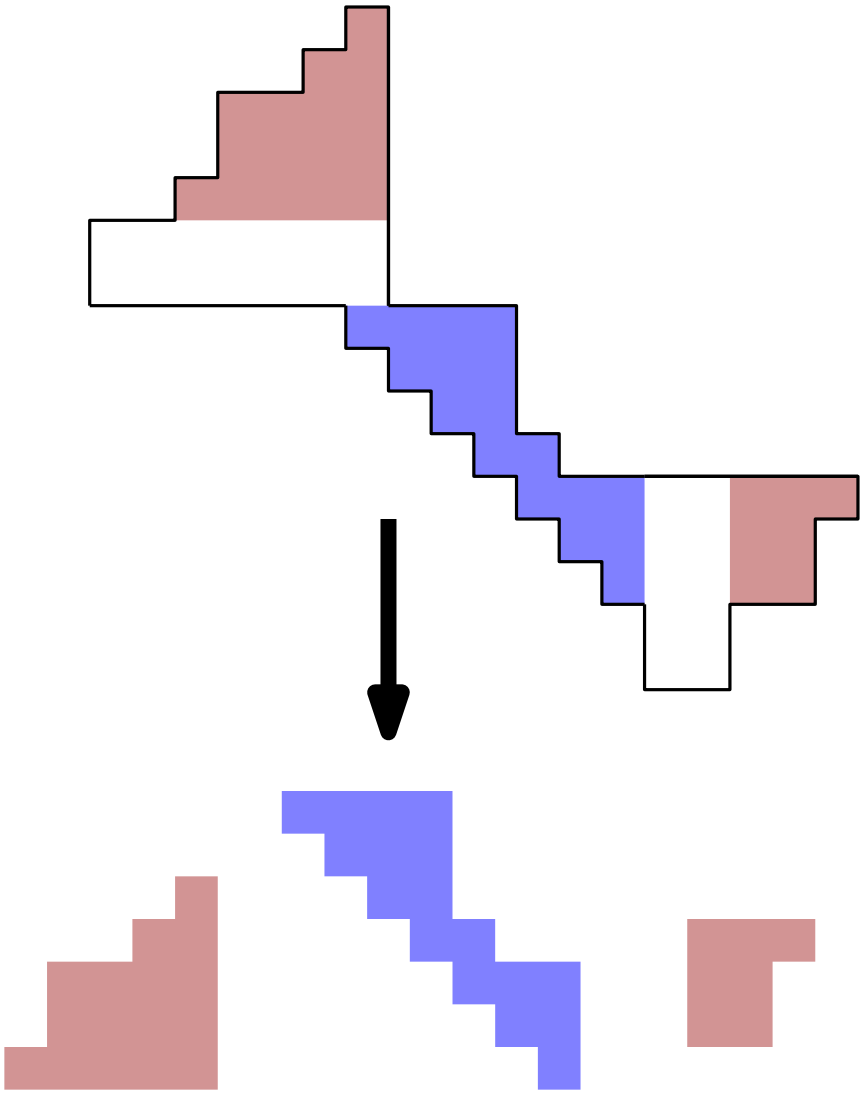
Computing Area of a Candidate

Given a candidate . . .



Computing Area of a Candidate

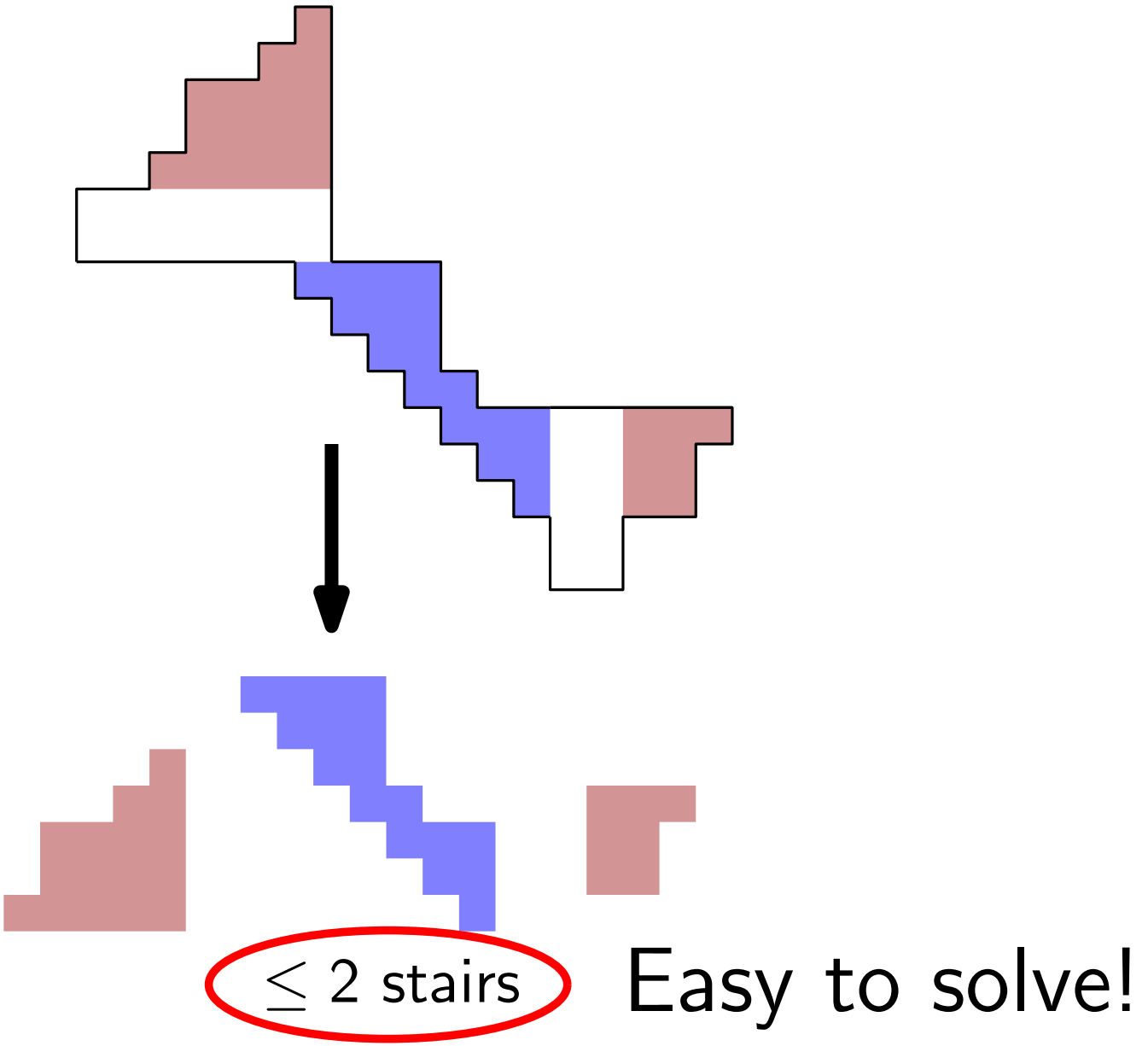
Given a candidate . . .



≤ 2 stairs

Computing Area of a Candidate

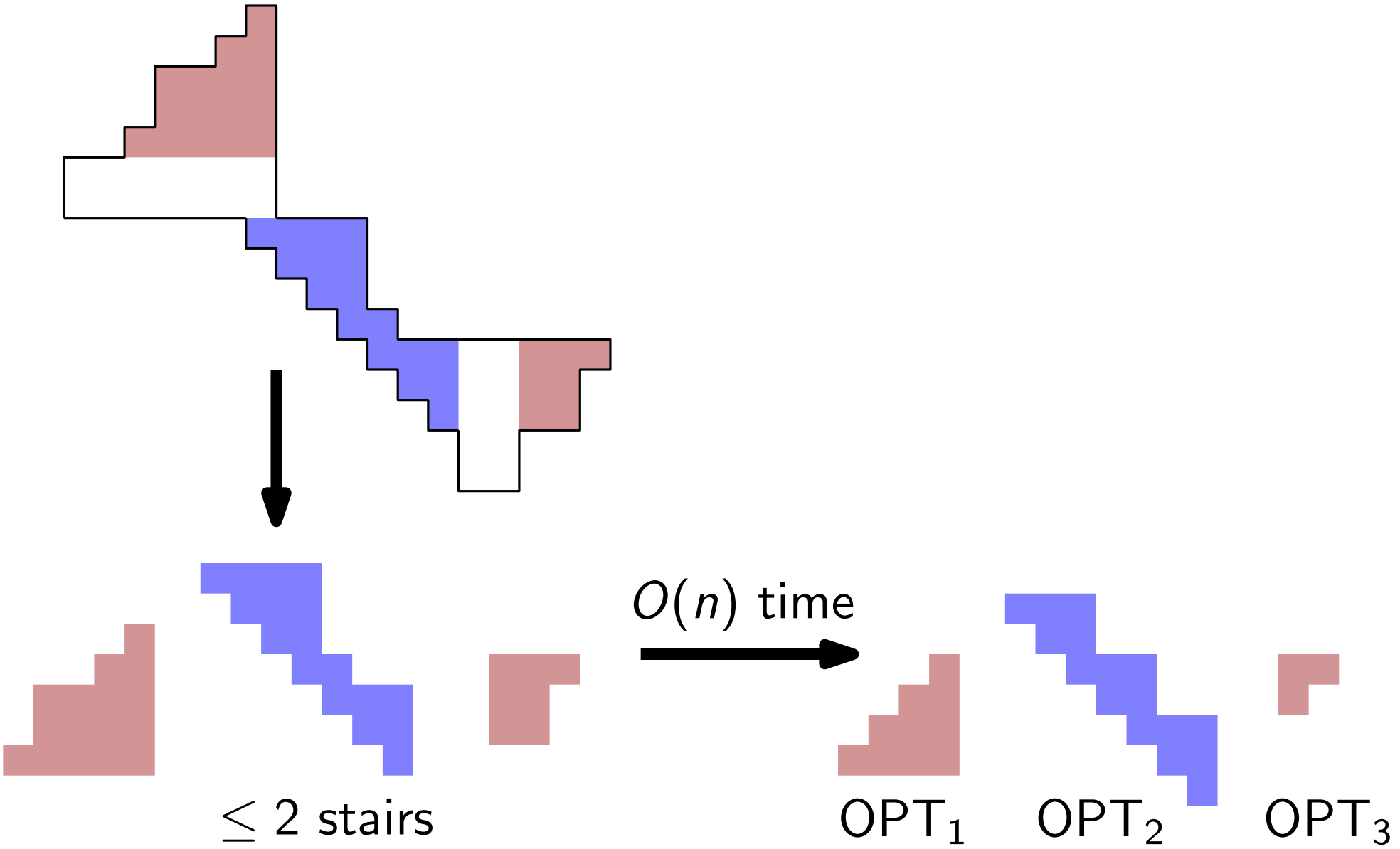
Given a candidate . . .



Easy to solve!

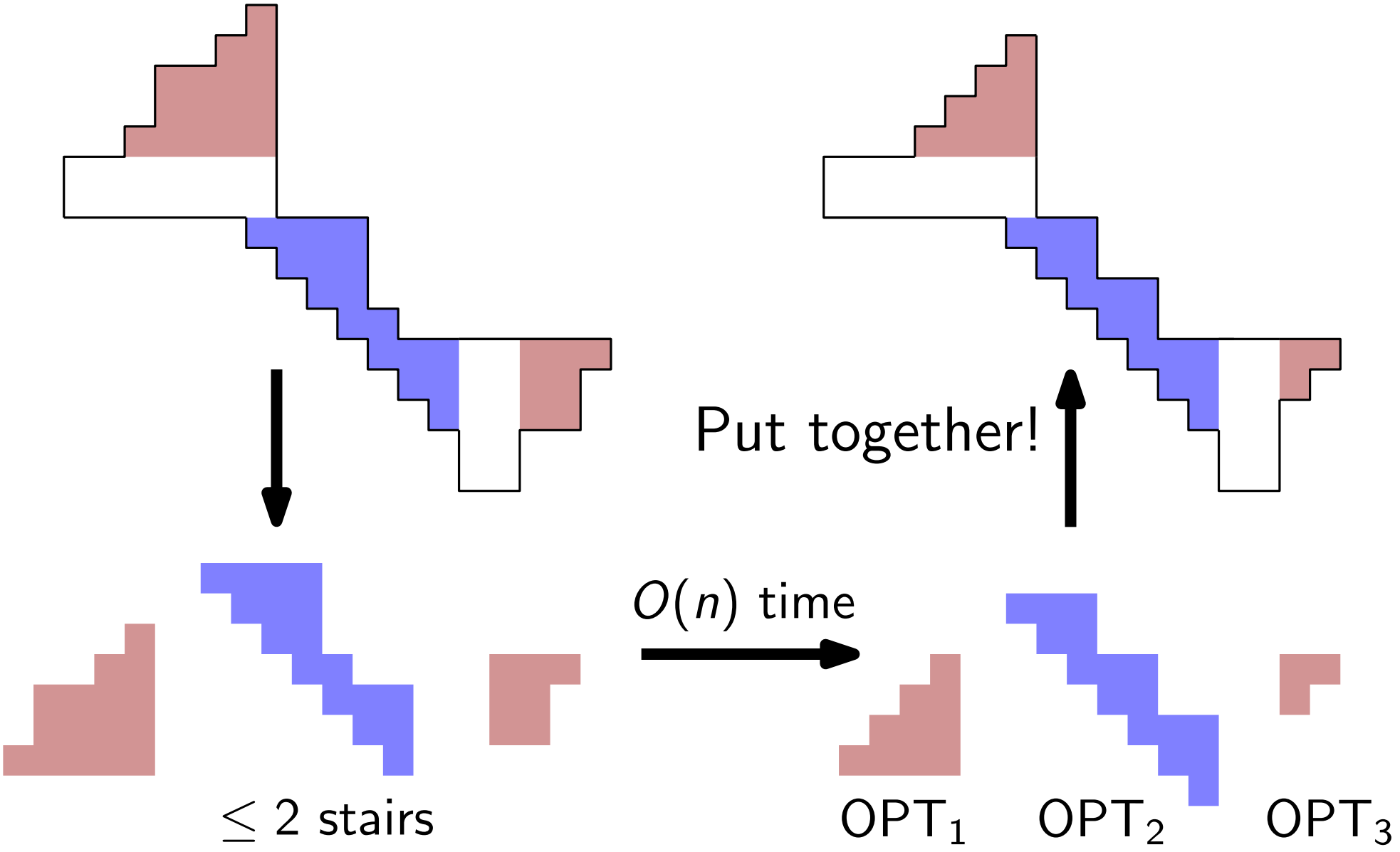
Computing Area of a Candidate

Given a candidate . . .



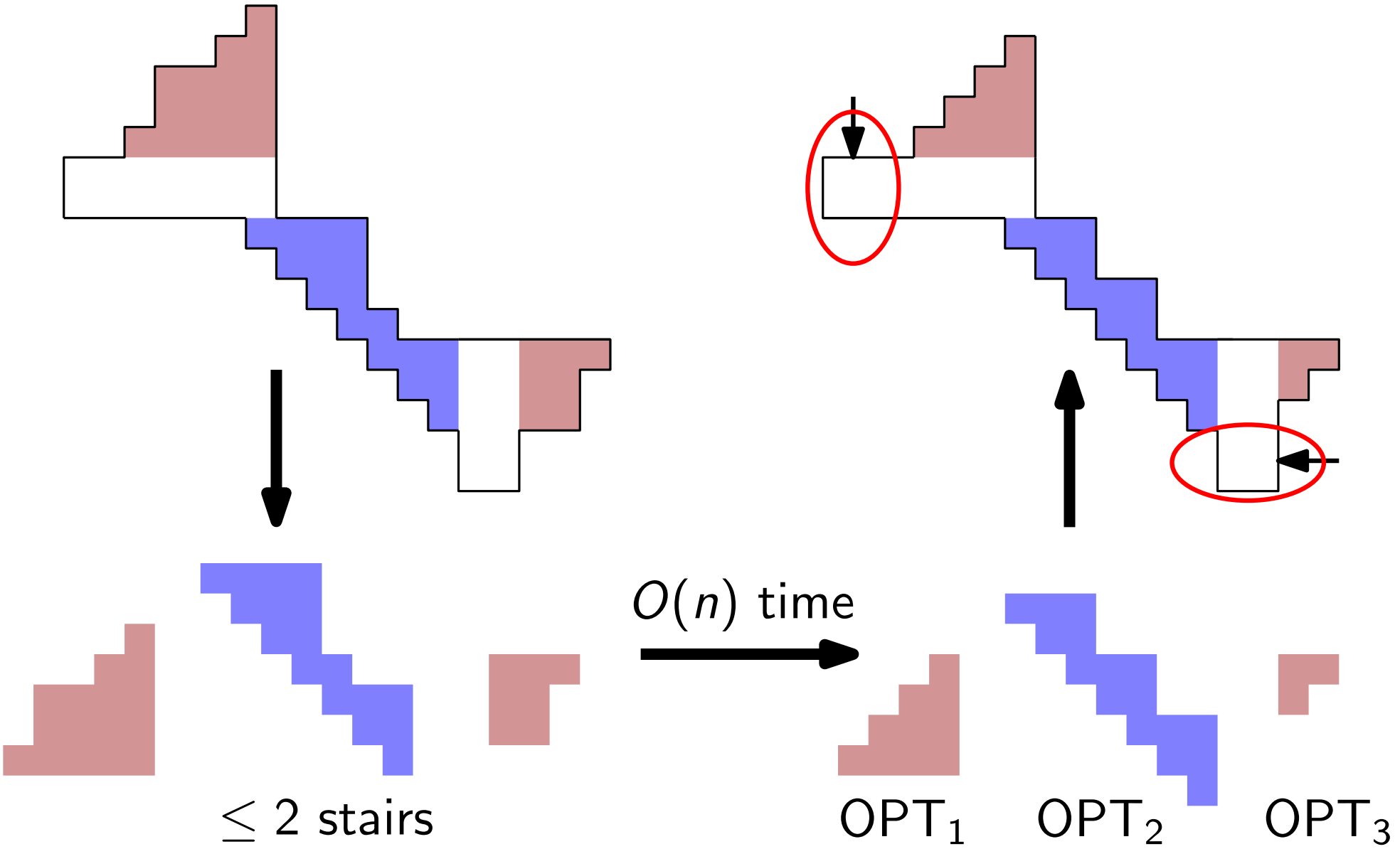
Computing Area of a Candidate

Given a candidate . . .



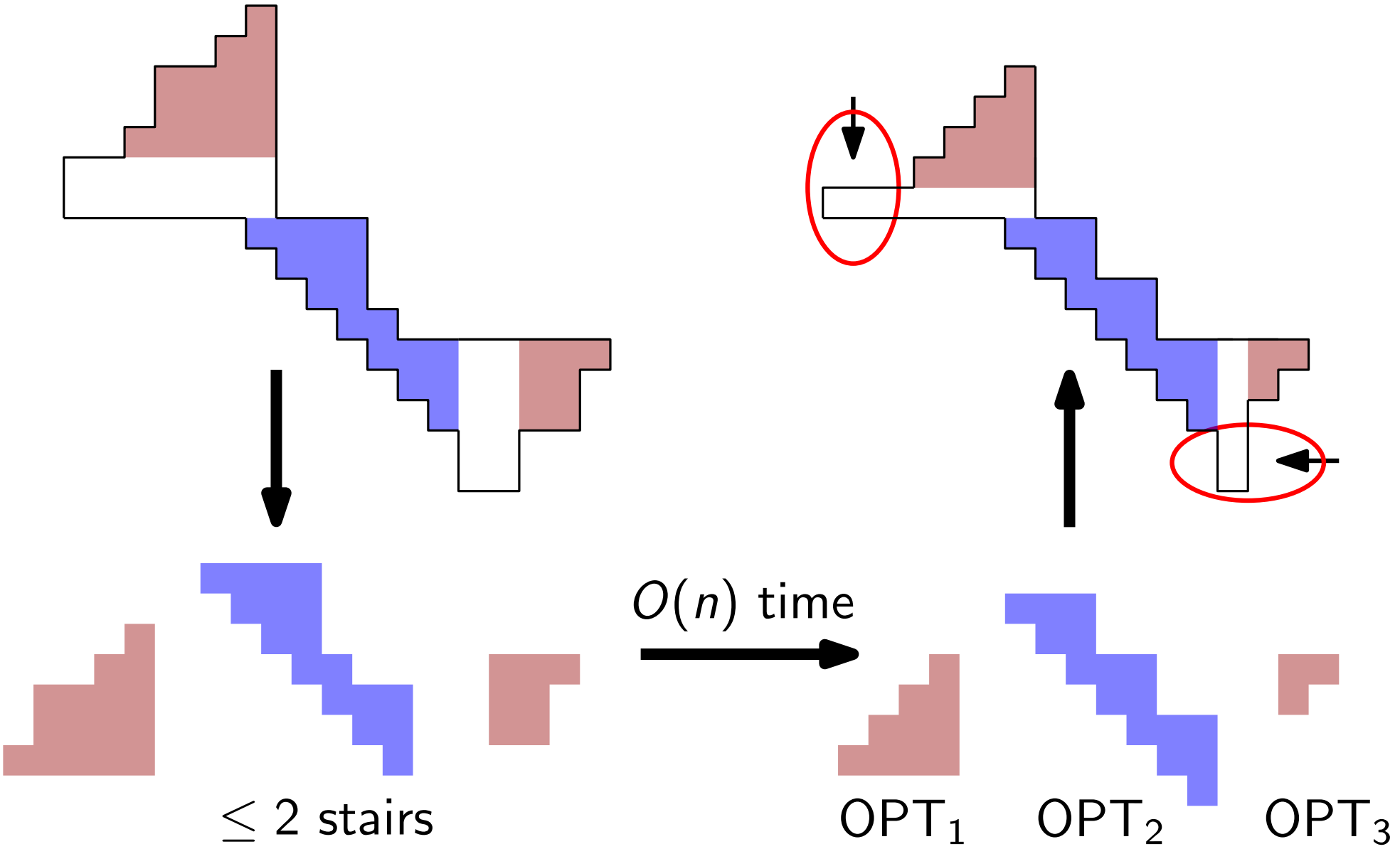
Computing Area of a Candidate

Given a candidate . . .



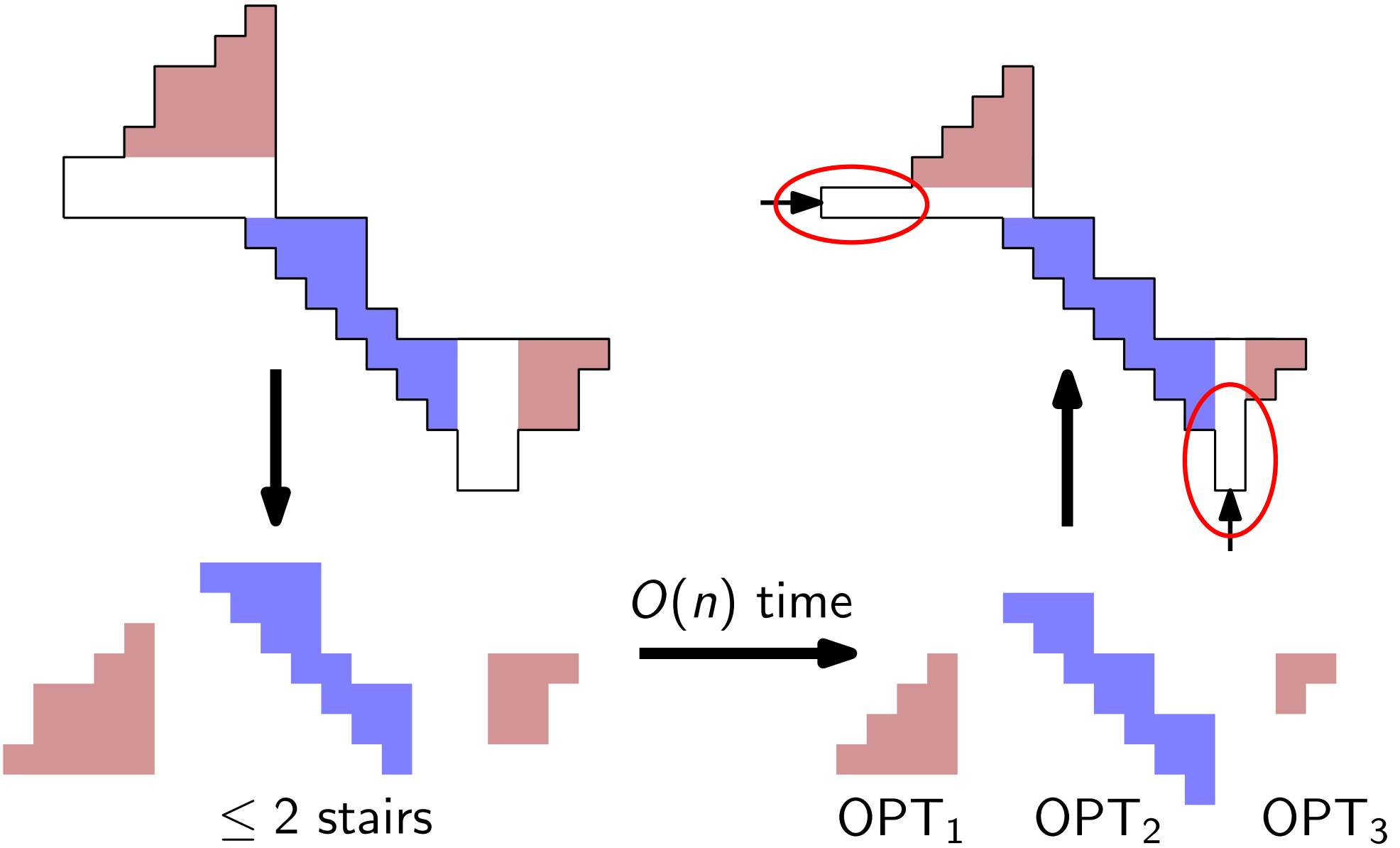
Computing Area of a Candidate

Given a candidate . . .



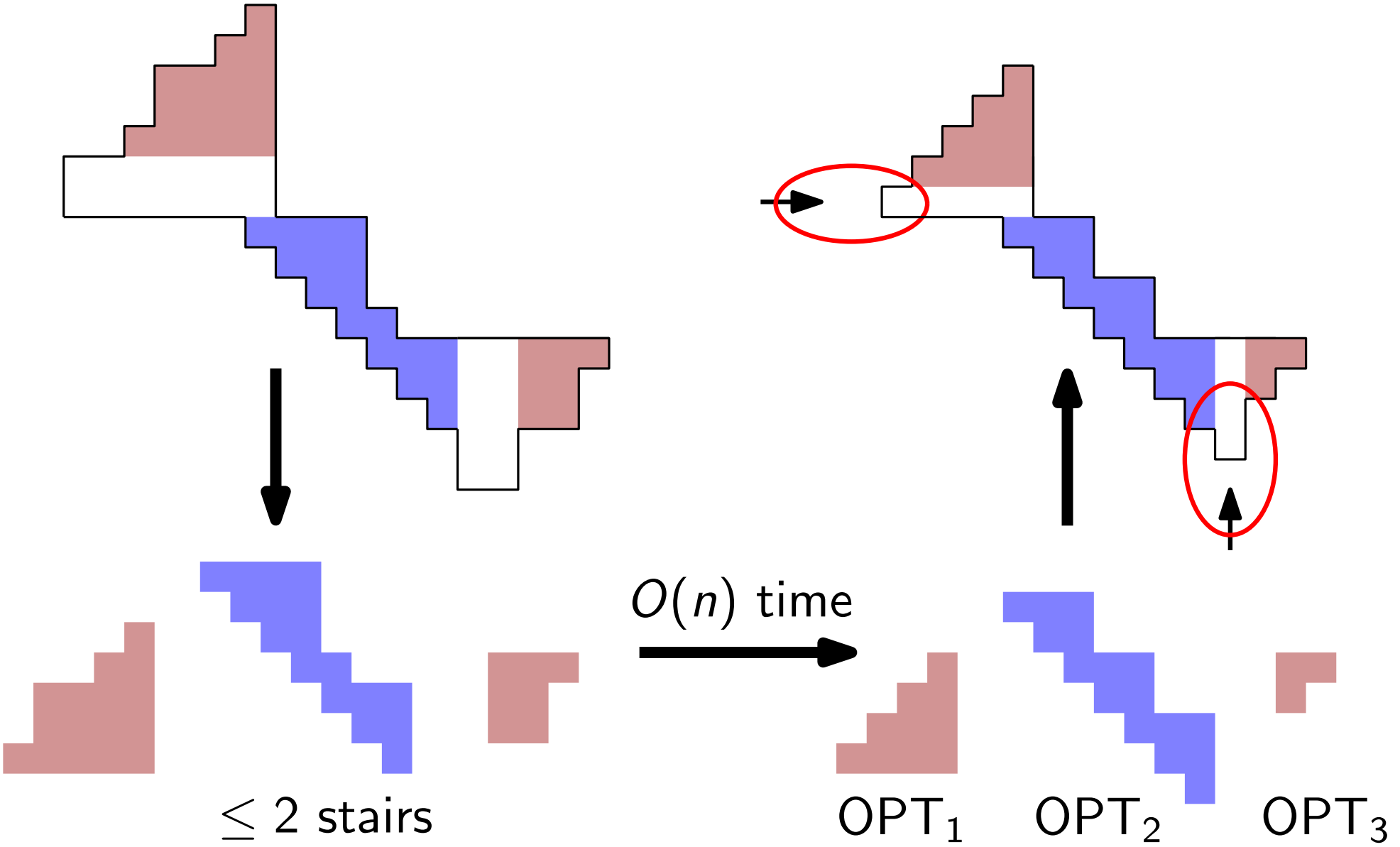
Computing Area of a Candidate

Given a candidate . . .



Computing Area of a Candidate

Given a candidate . . .



Algorithm

Algorithm:

For every candidate $\times O(1)$

compute $\mathrm{OPT}_{\mathrm{candidate}}$ $O(n)$

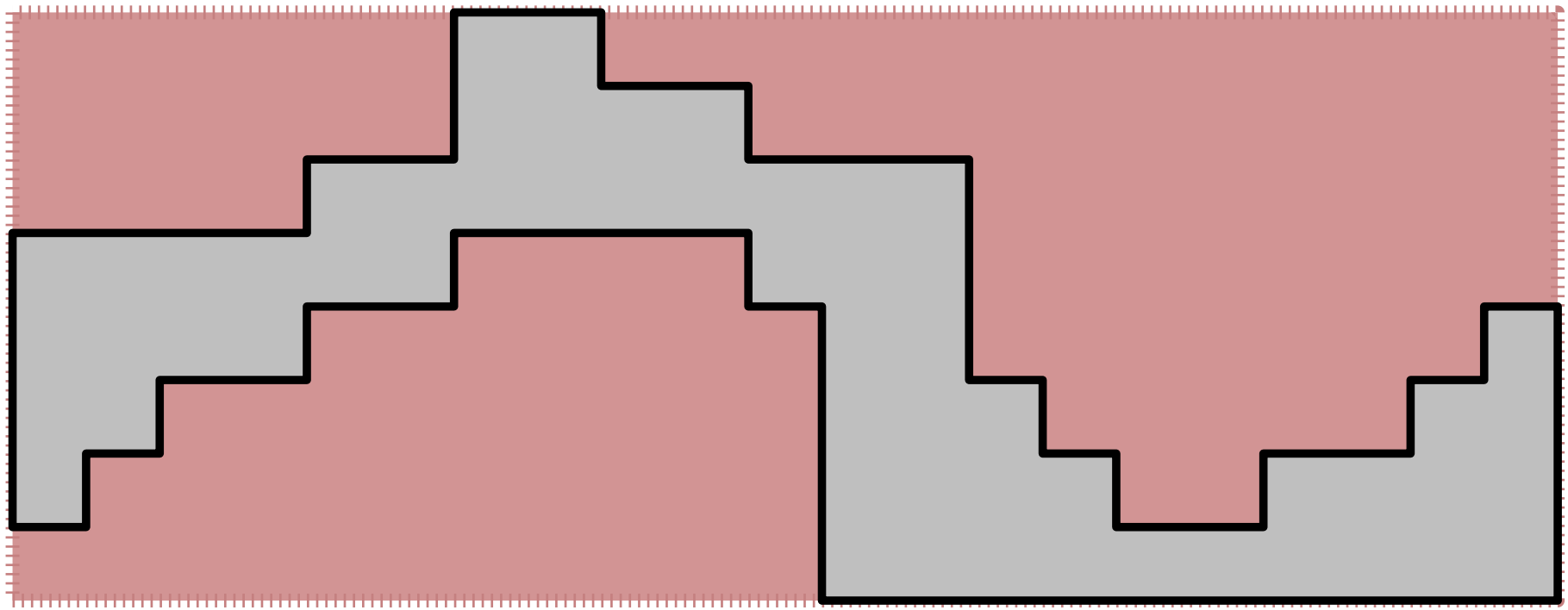
$$\mathrm{OPT} = \min \mathrm{OPT}_{\mathrm{candidate}}$$

$O(n)$

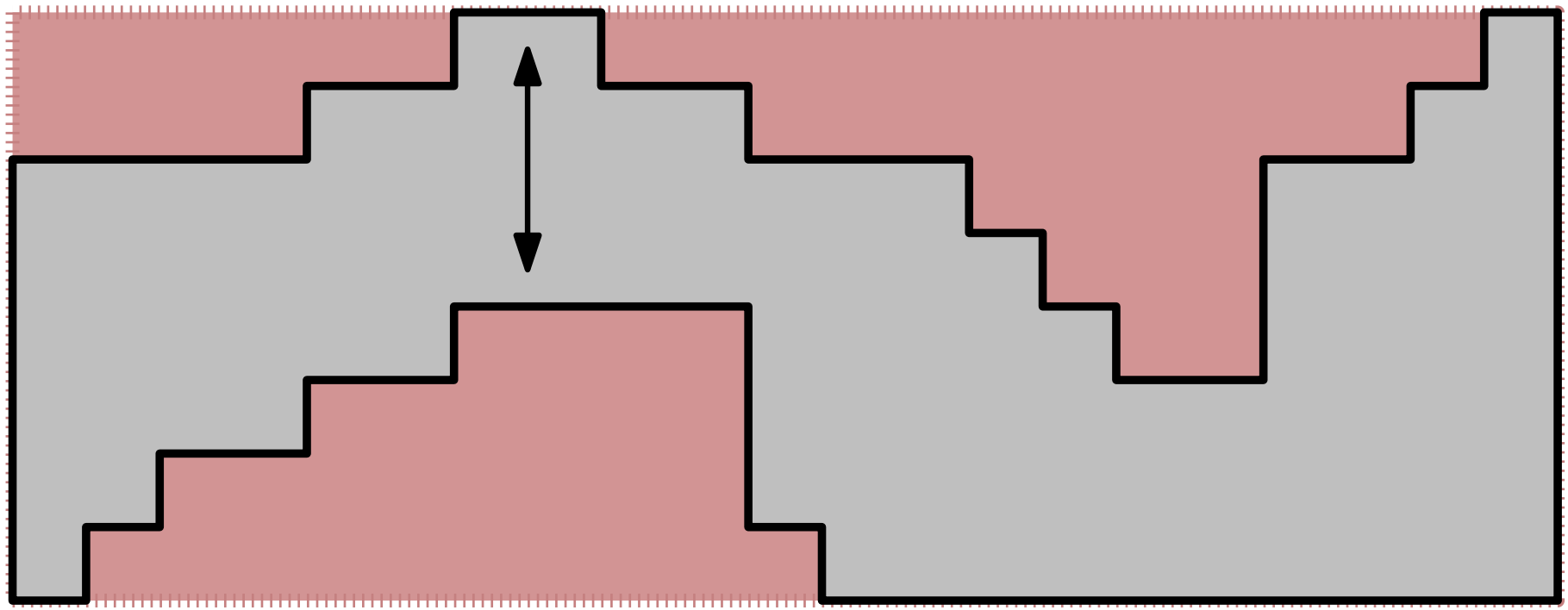
Minimizing Bounding Box of x-monotone Polygons

Canonical x-monotone Polygons

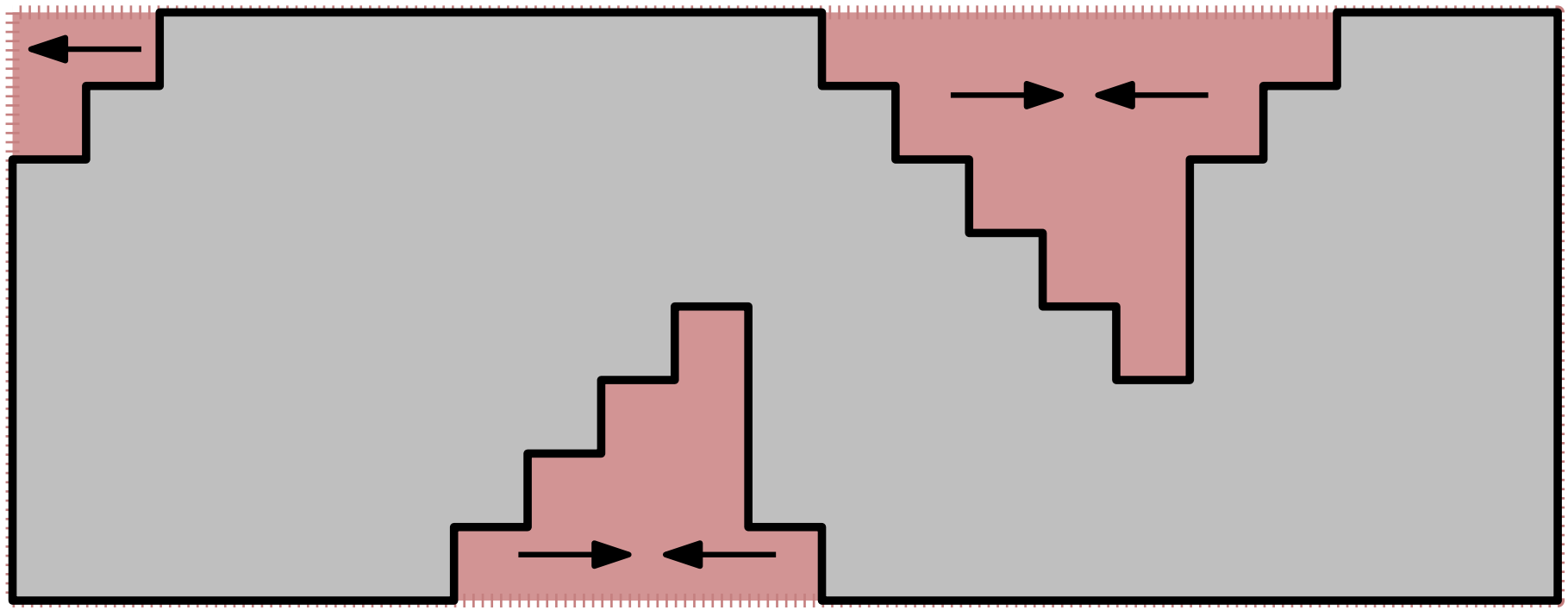
Bounding Box (BB)



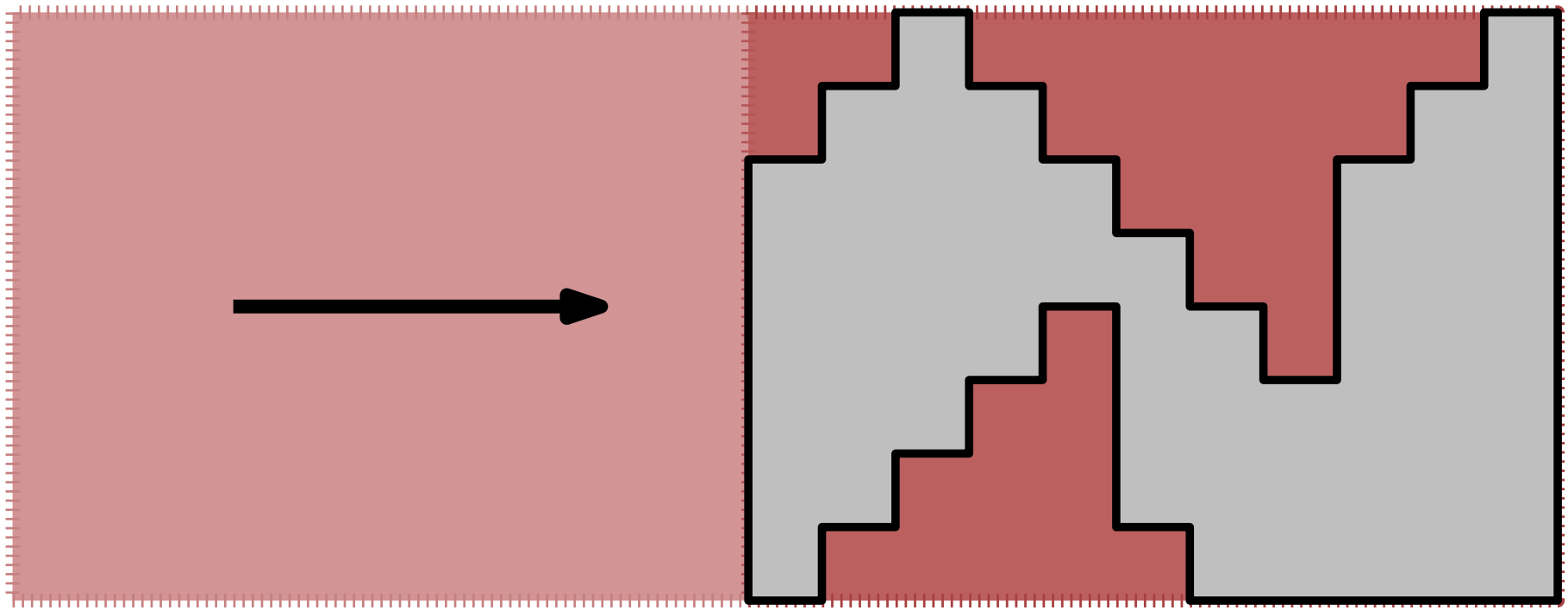
Canonical x-monotone Polygons



Canonical x-monotone Polygons

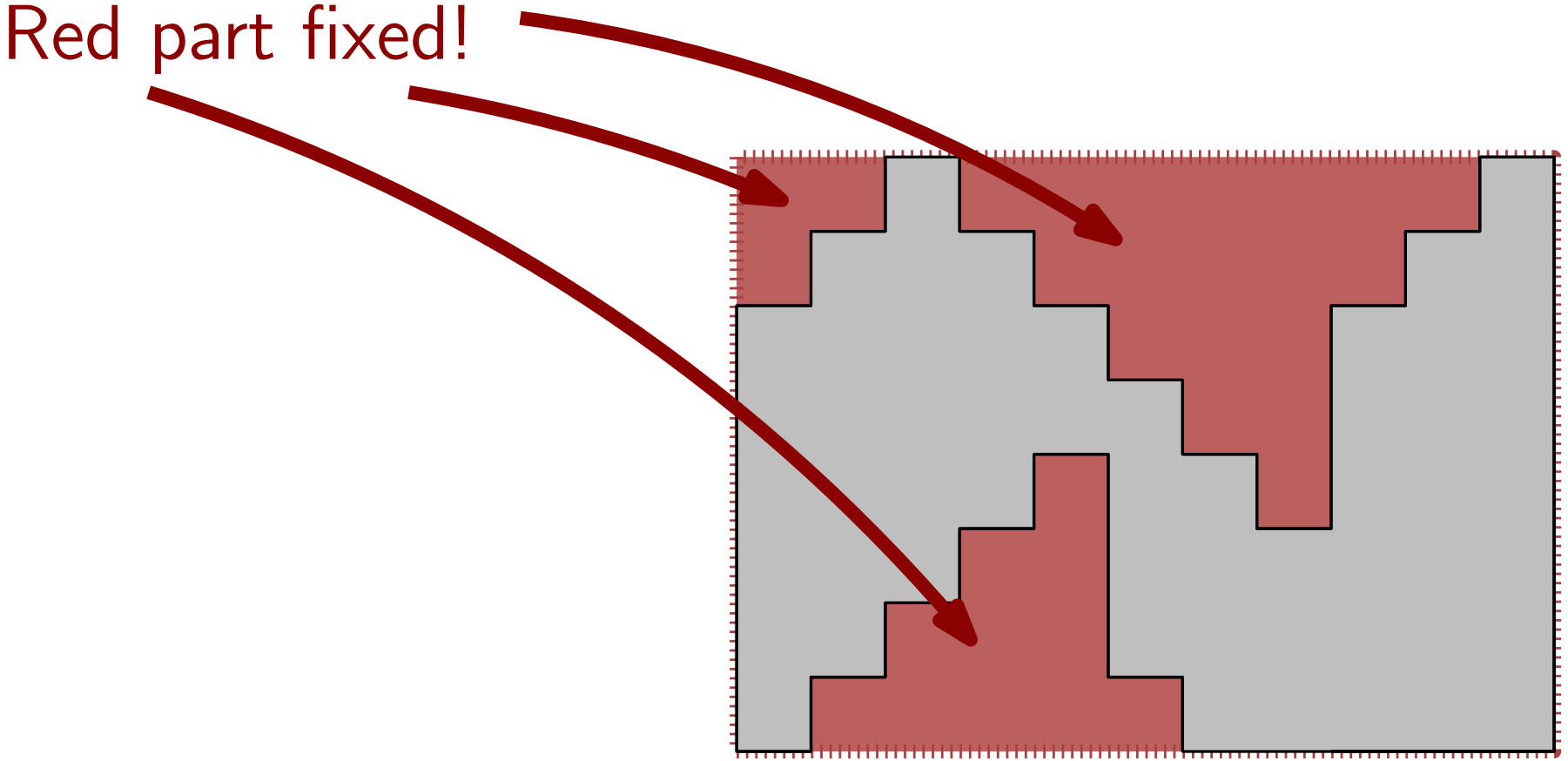


Canonical x-monotone Polygons



Canonical

Canonical x-monotone Polygons

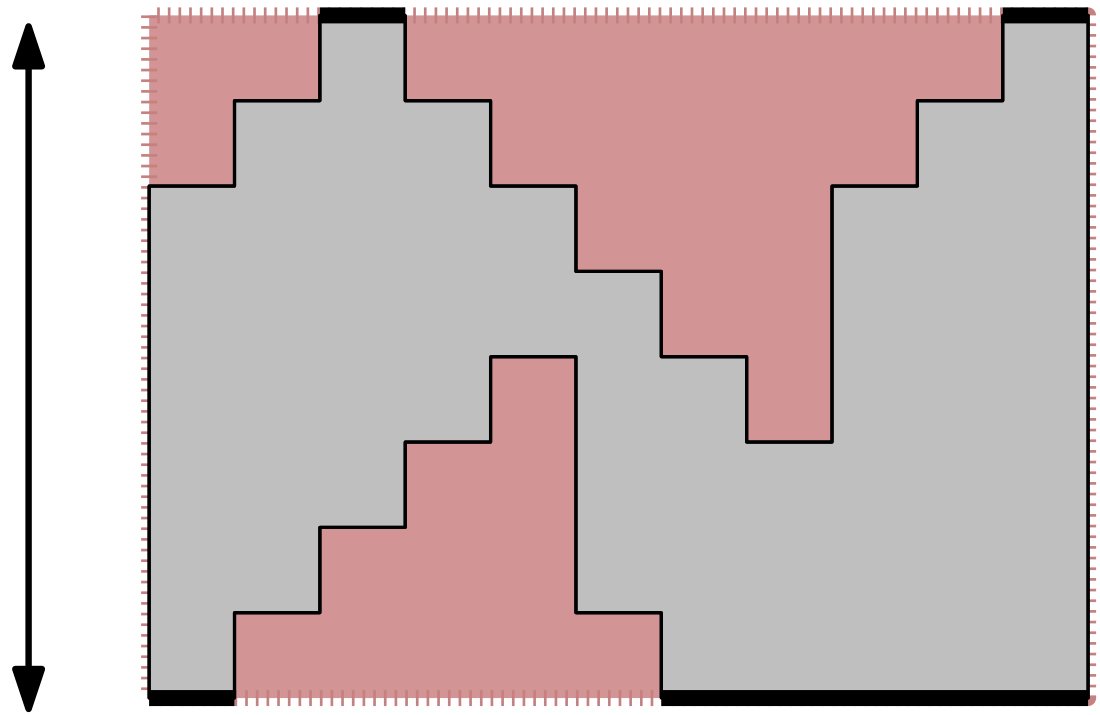


Canonical x-monotone Polygons

Red part fixed!

Only unknowns:

a) Height of BB



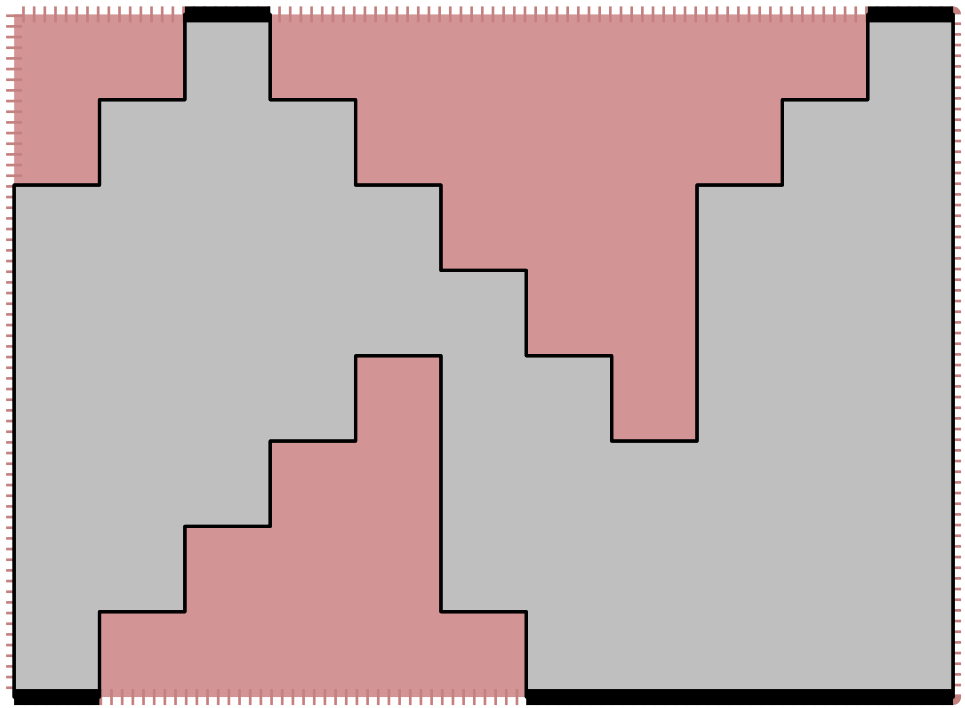
b) Width of extreme edges

Canonical x-monotone Polygons

Red part fixed!

Only unknowns:

a) Height of BB



b) Width of extreme edges

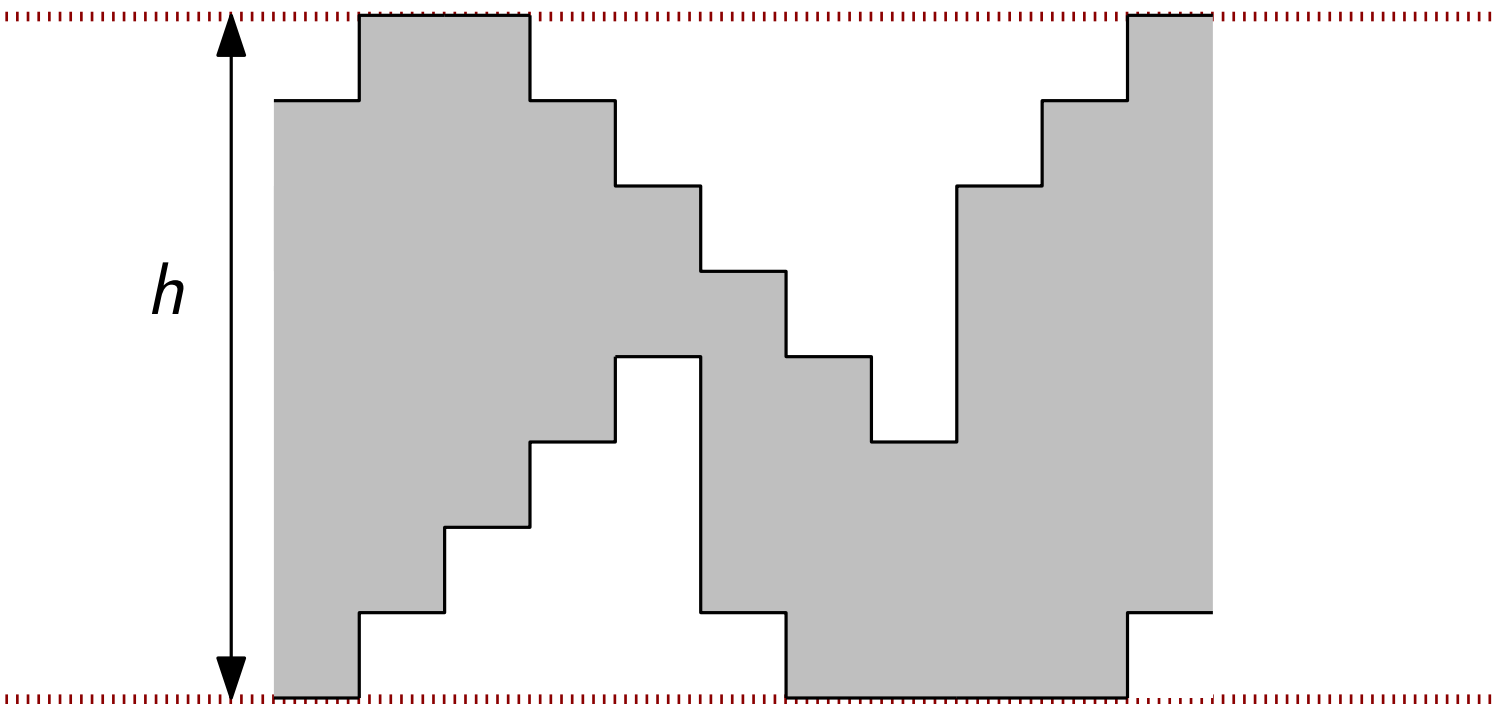
Algorithm:

Guess height of BB

Use DP to find widths

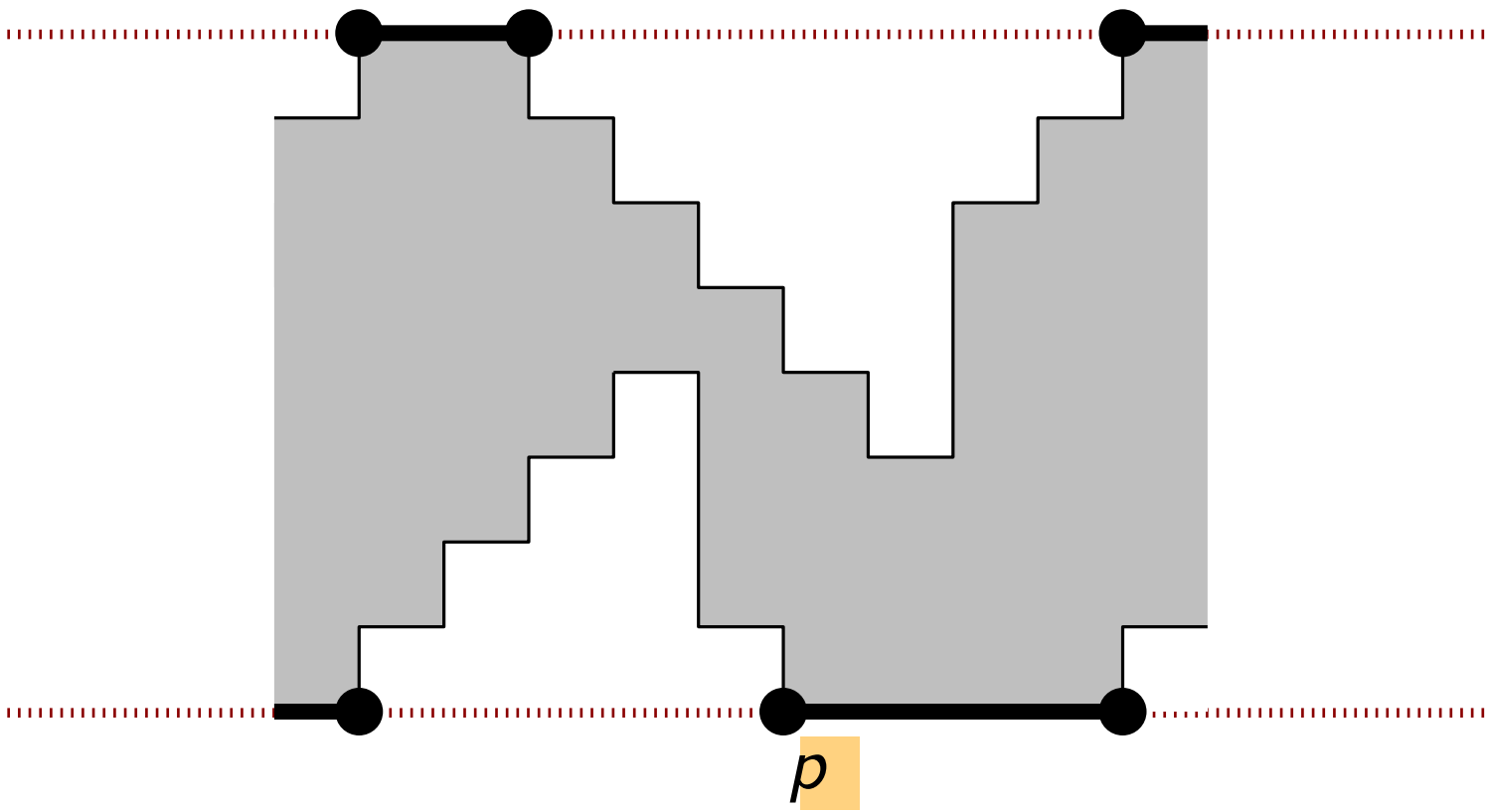
DP Algorithm

$T[\ ,\ ]$



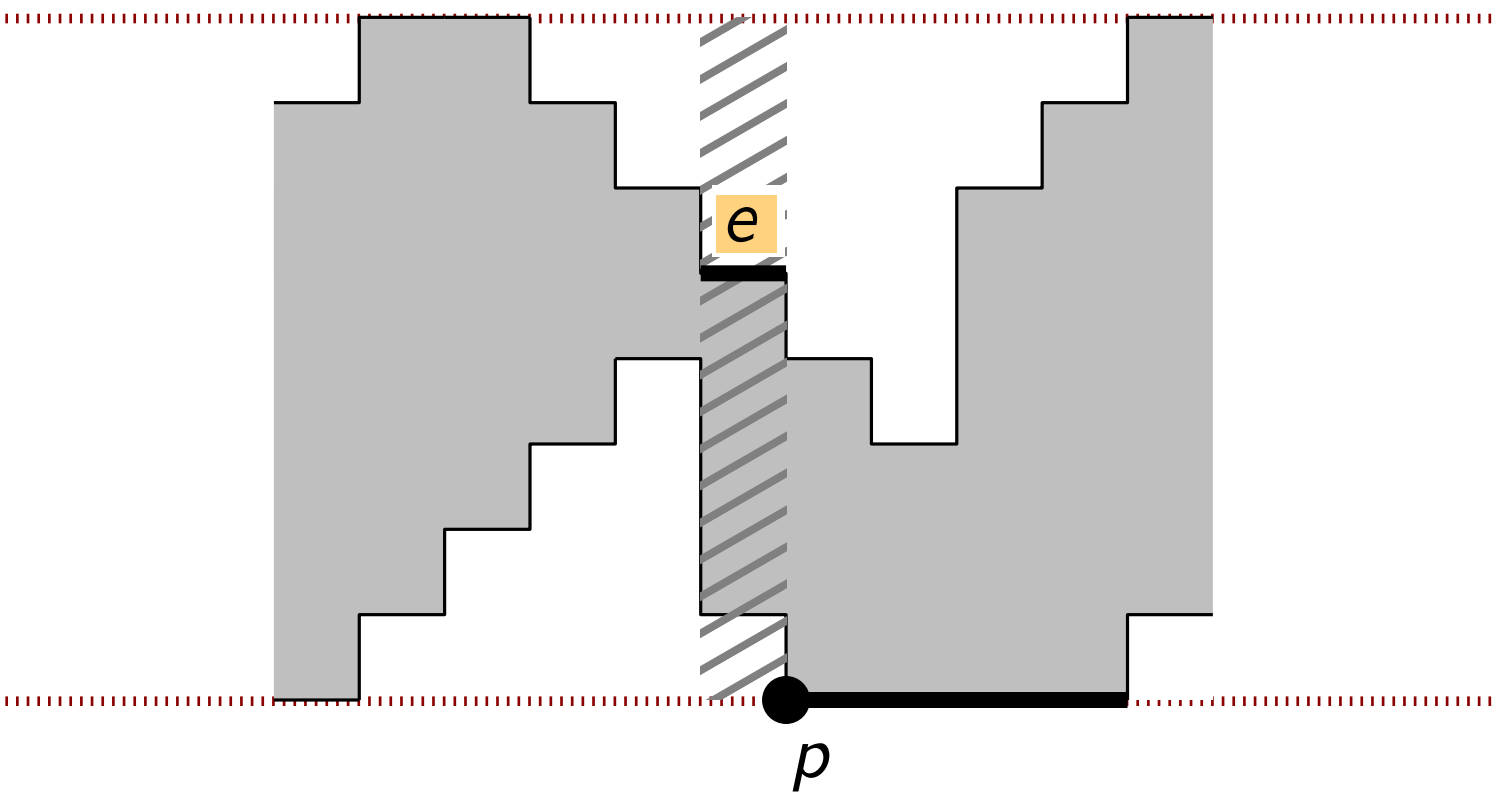
DP Algorithm

$T[p, \ ]$



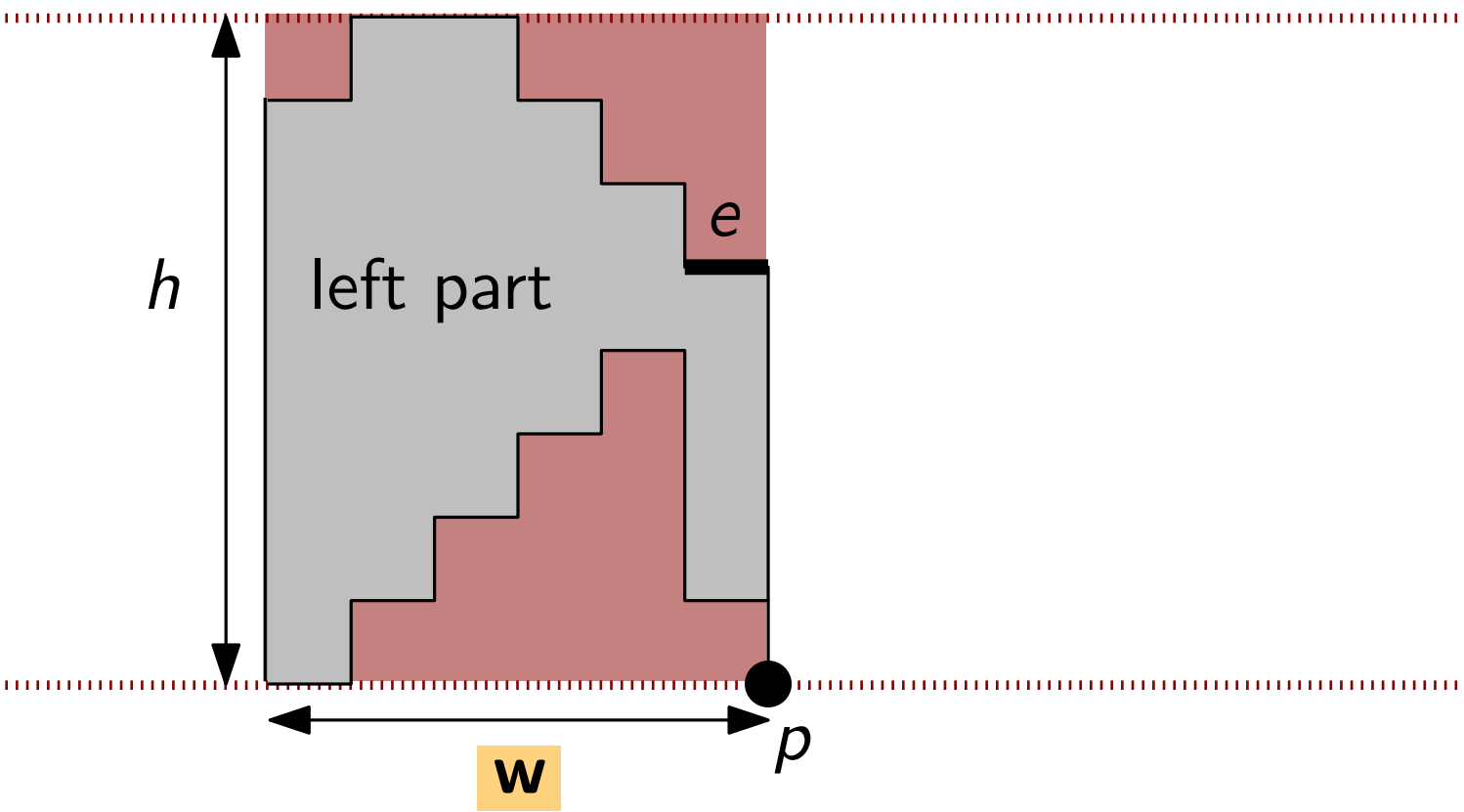
DP Algorithm

$T[p, e]$



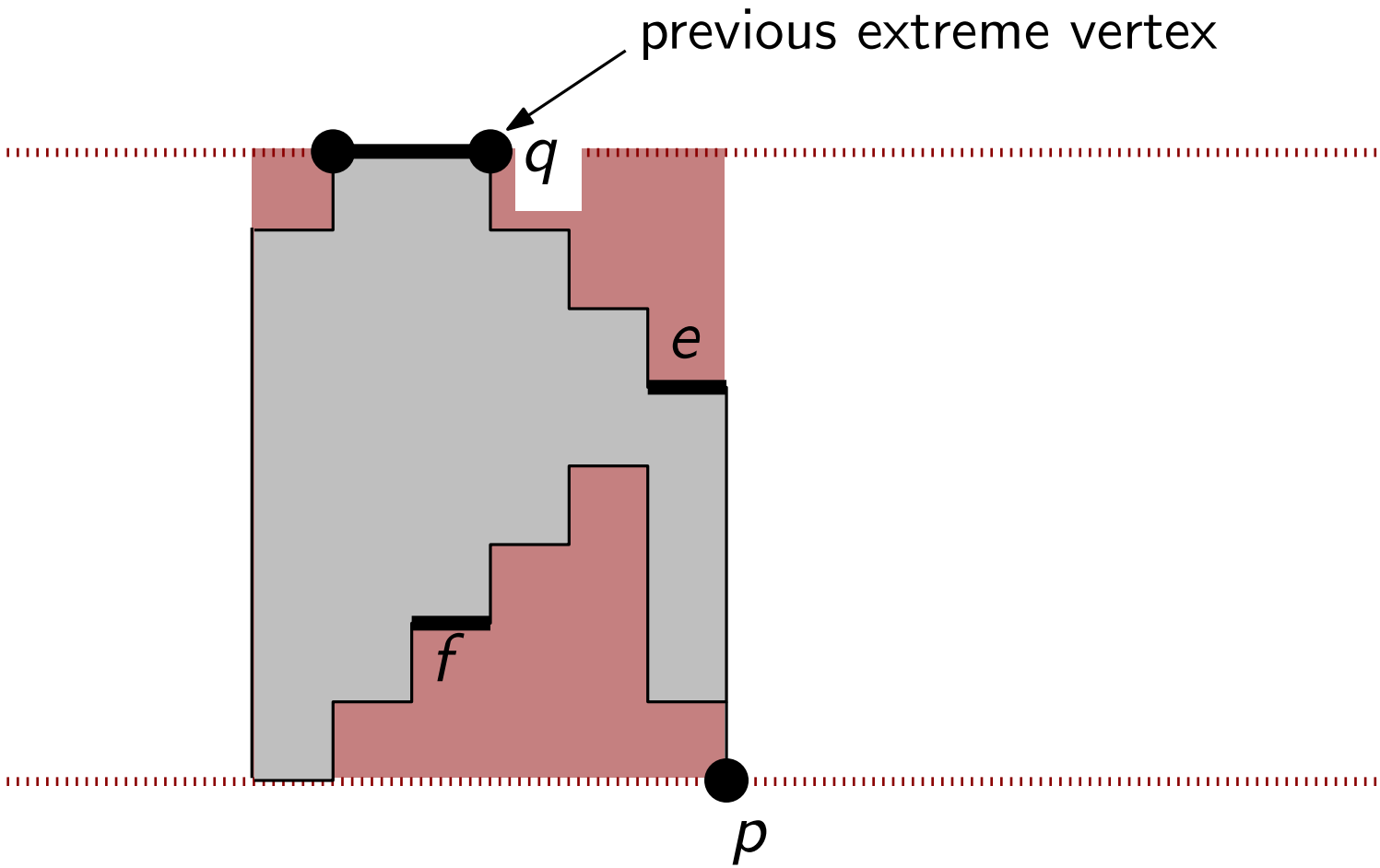
DP Algorithm

$$T[p, e] = \text{minimum } \mathbf{w} \text{ s.t. left part} \in \text{BB}(\mathbf{w} \times h)$$



DP Algorithm

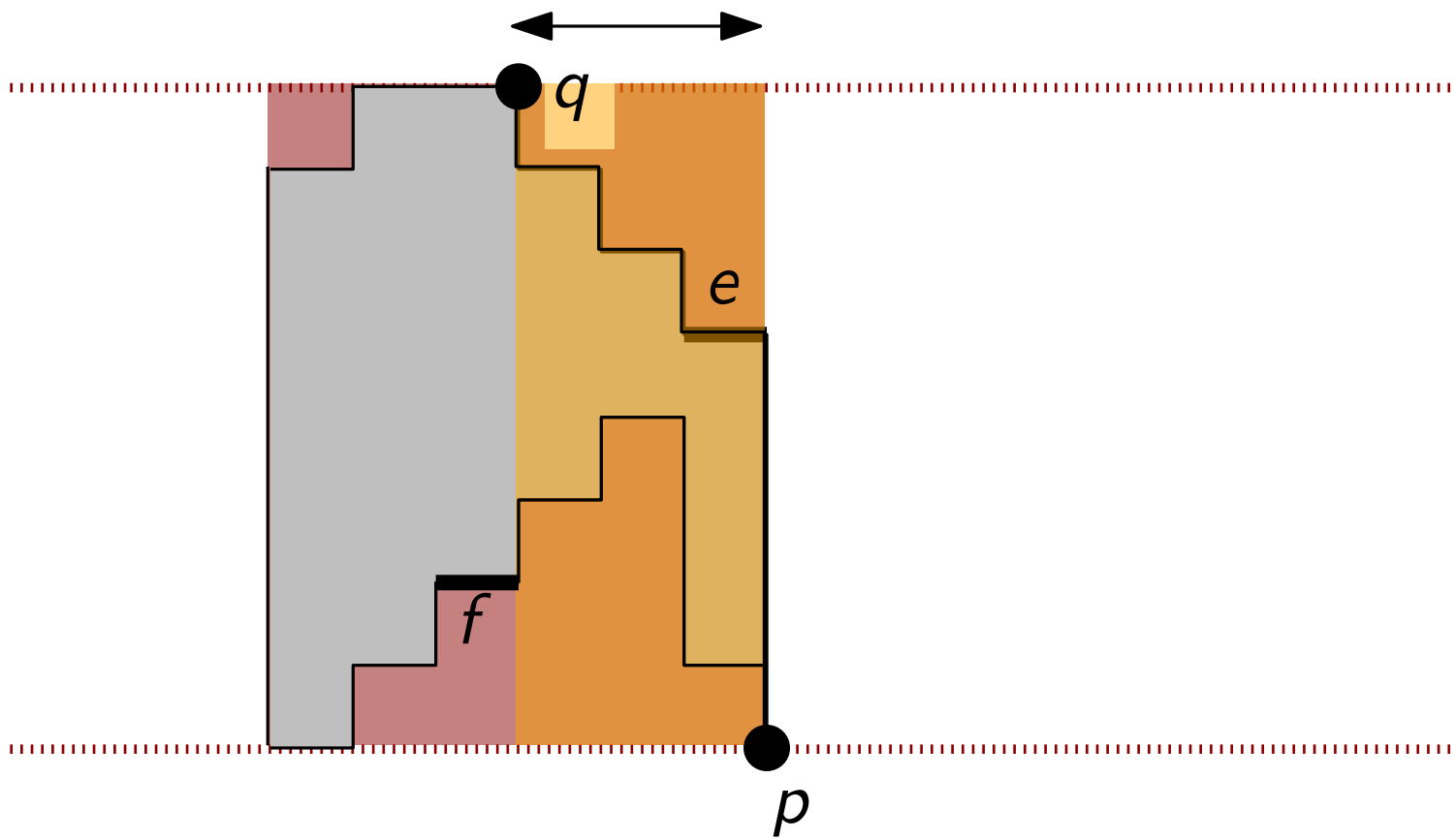
$$T[p, e] = \text{minimum } \mathbf{w} \text{ s.t. left part} \in \text{BB}(\mathbf{w} \times h)$$



DP Algorithm

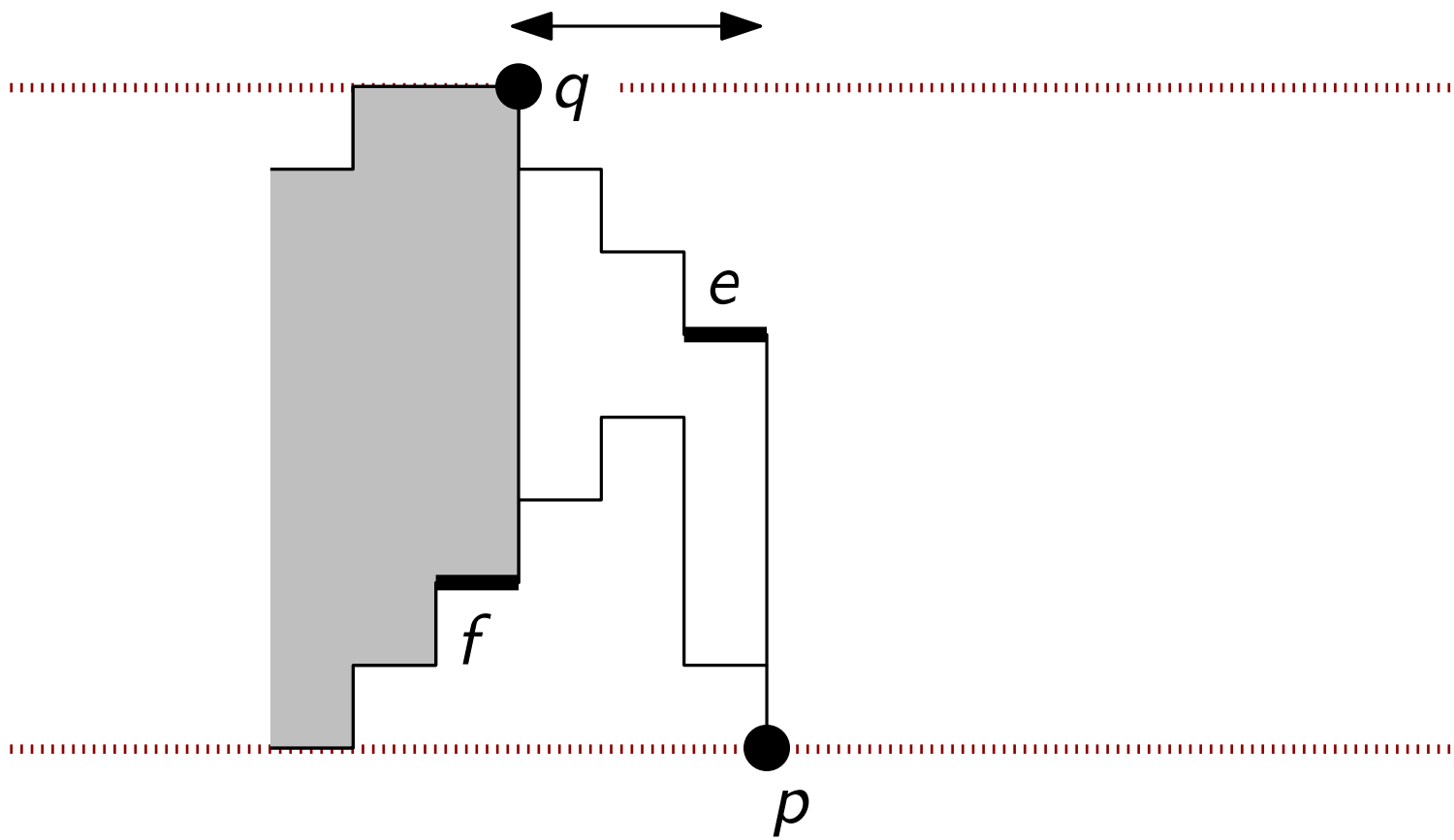
$T[p, e]$ = minimum $\mathbf{w}$ s.t. left part $\in$ BB($\mathbf{w} \times h$)

w' is fixed



DP Algorithm

$$T[p, e] = \text{minimum } \mathbf{w} \text{ s.t. left part} \in \text{BB}(\mathbf{w} \times h)$$
$$= \quad T[q, f] \quad + \quad w'$$



DP Algorithm

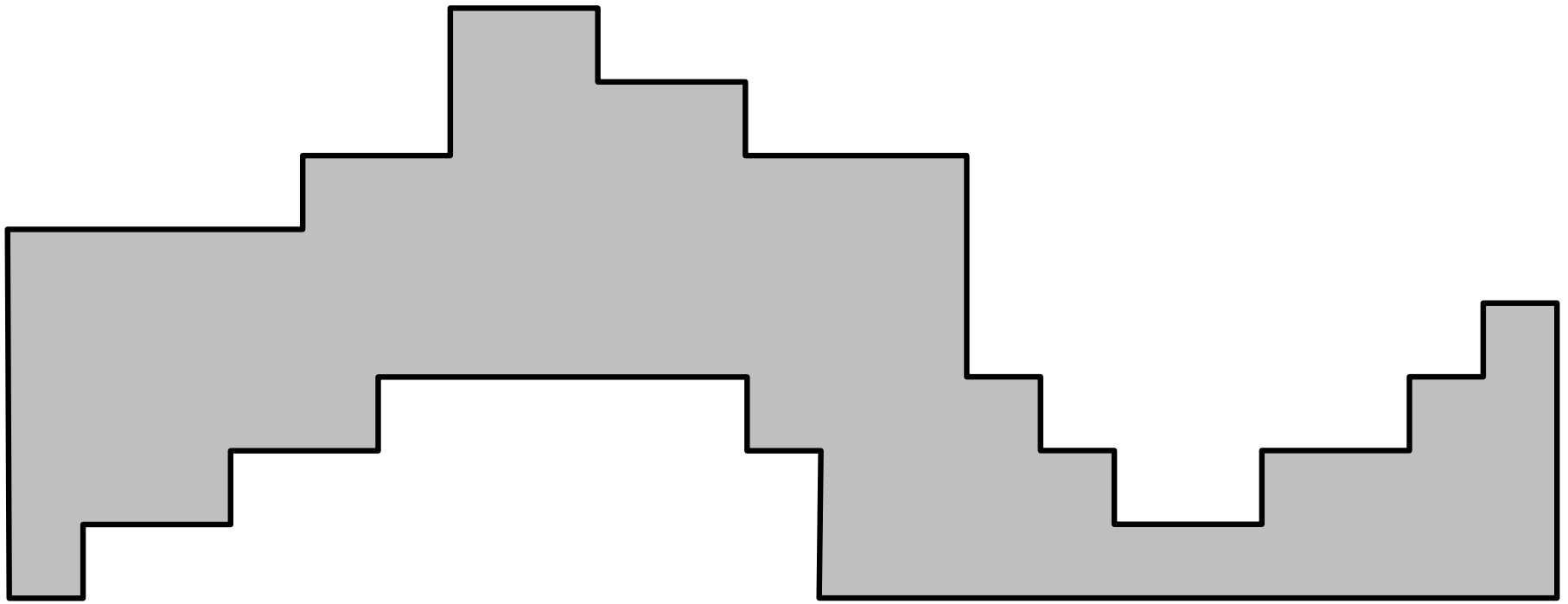
$$T[p, e] = \min \mathbf{w} \text{ s.t. left part} \in \mathrm{BB}(\mathbf{w} \times h)$$
$$= \quad T[q, f] \quad + \quad w'$$

Algorithm:

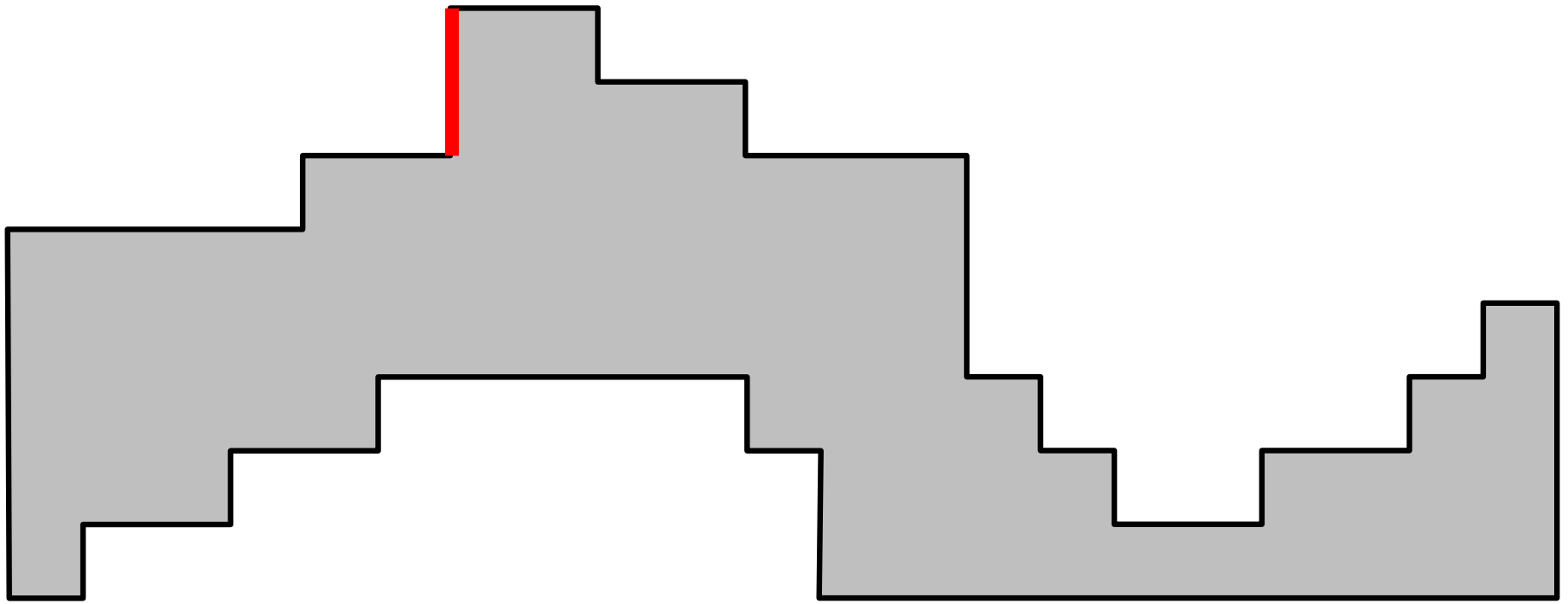
Guess height of BB	$O(n)$
Use DP to find widths	$O(n^2)$
	$O(n^3)$

Minimizing the Perimeter of x-monotone Polygons

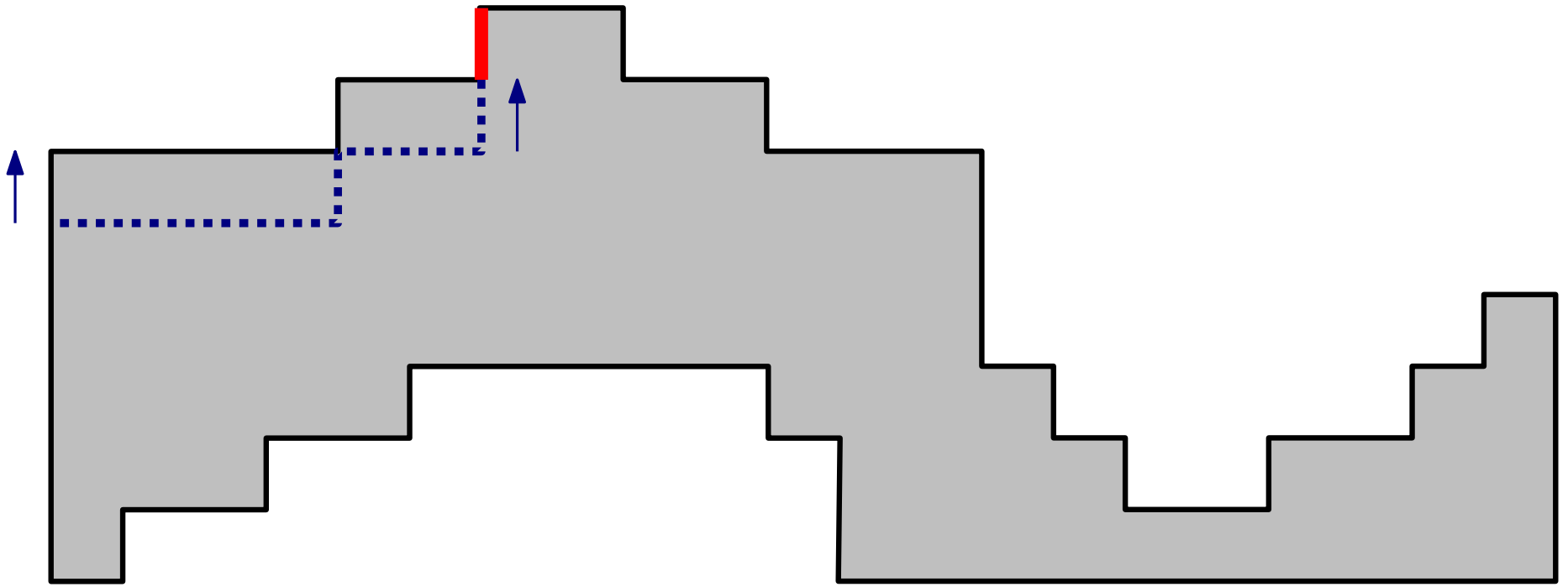
Canonical x-monotone Polygons



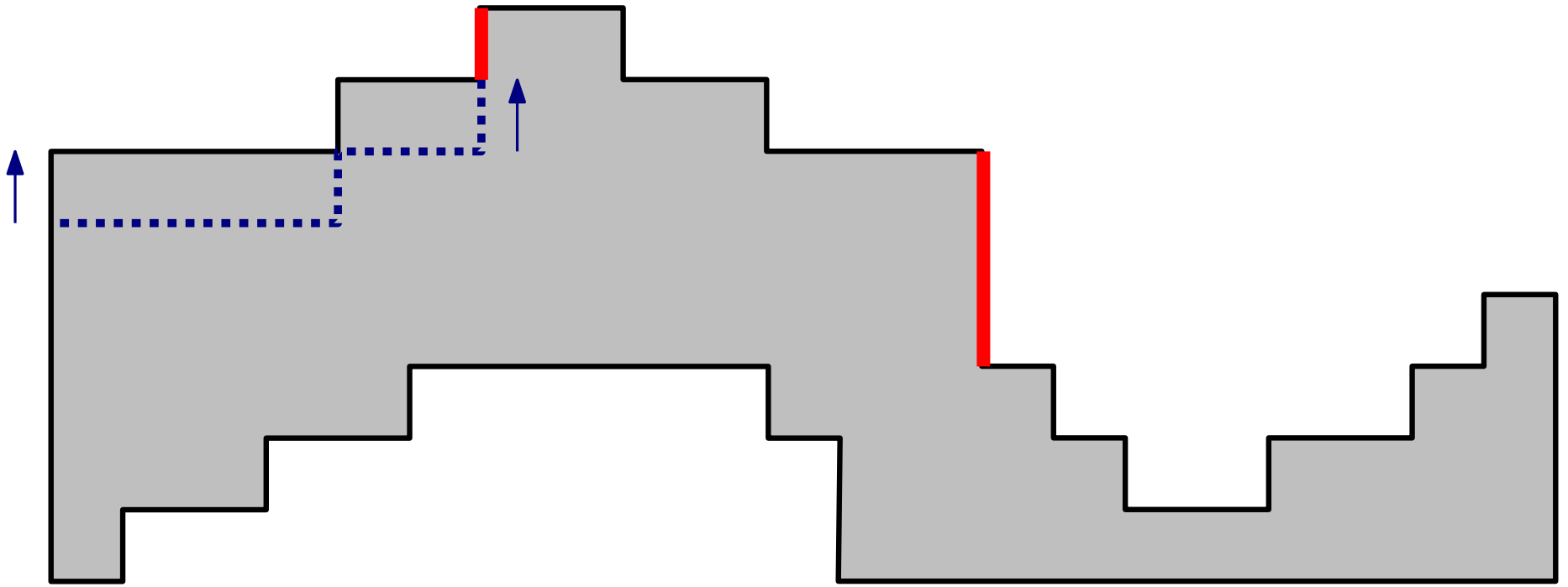
Canonical x-monotone Polygons



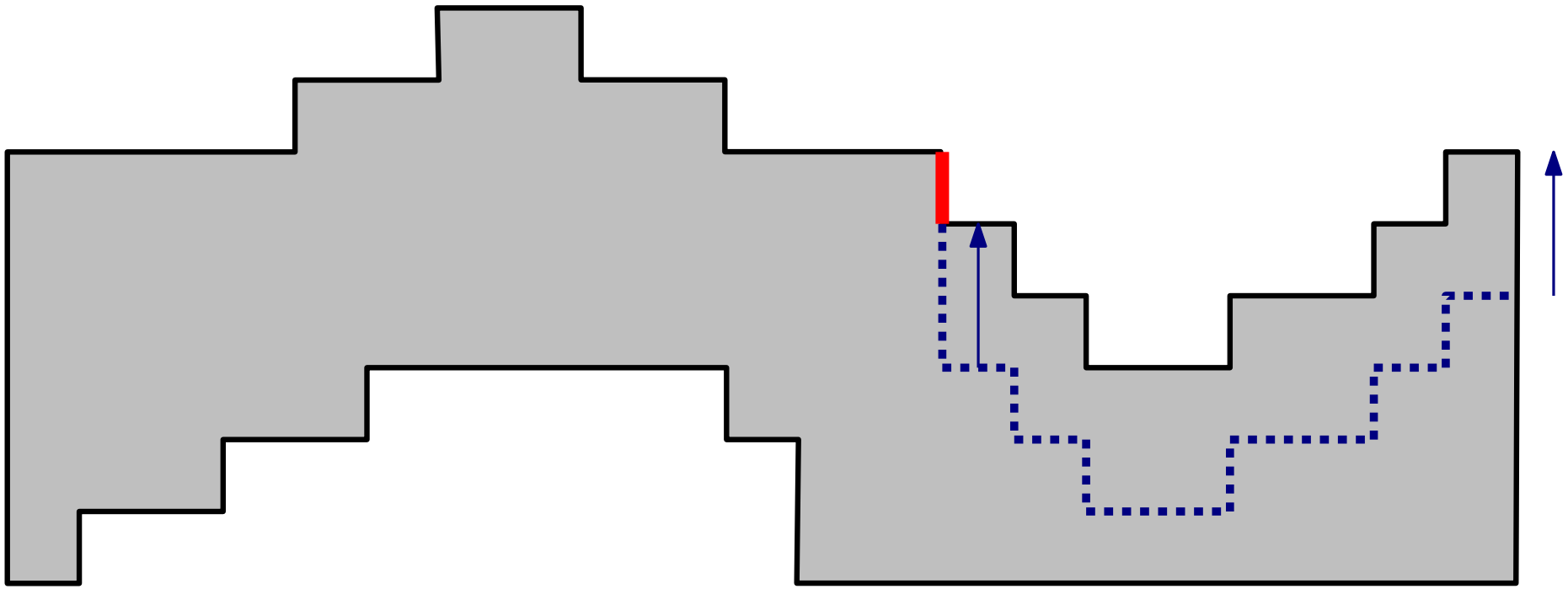
Canonical x-monotone Polygons



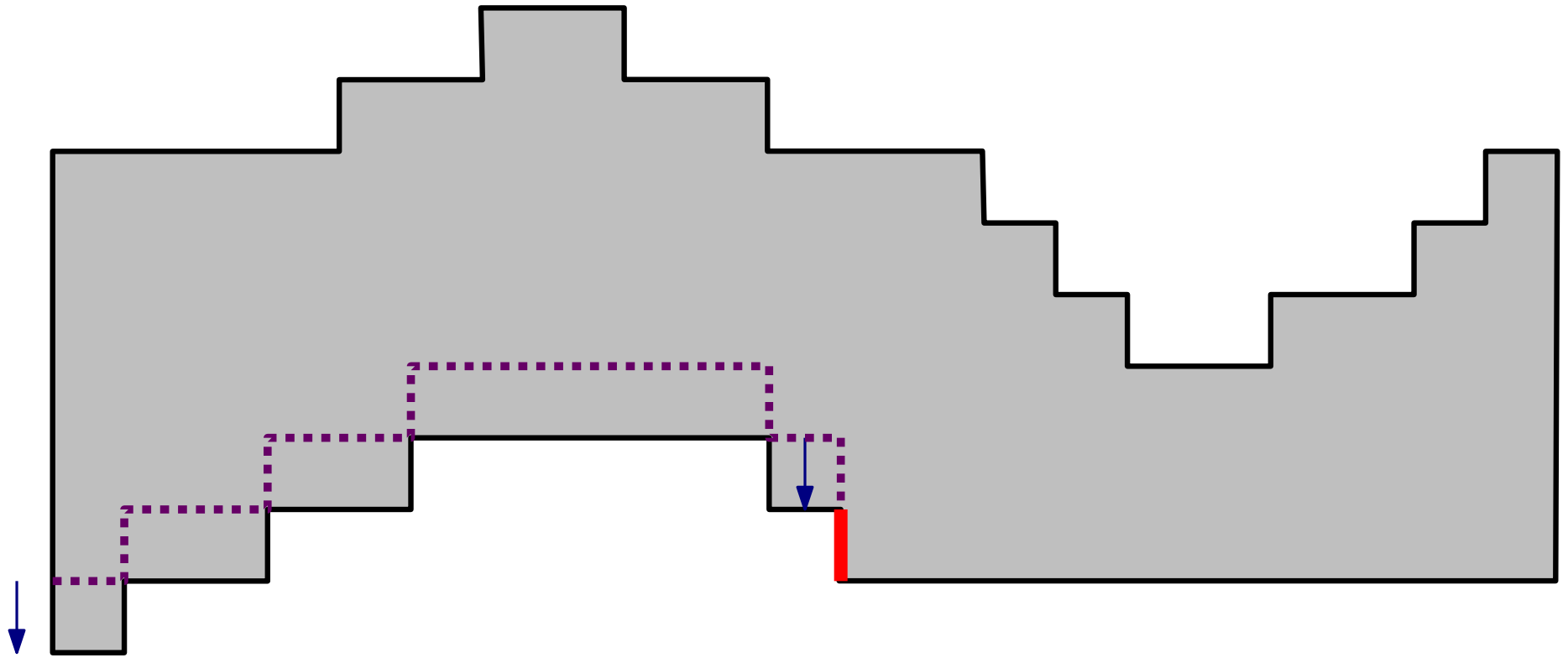
Canonical x-monotone Polygons



Canonical x-monotone Polygons

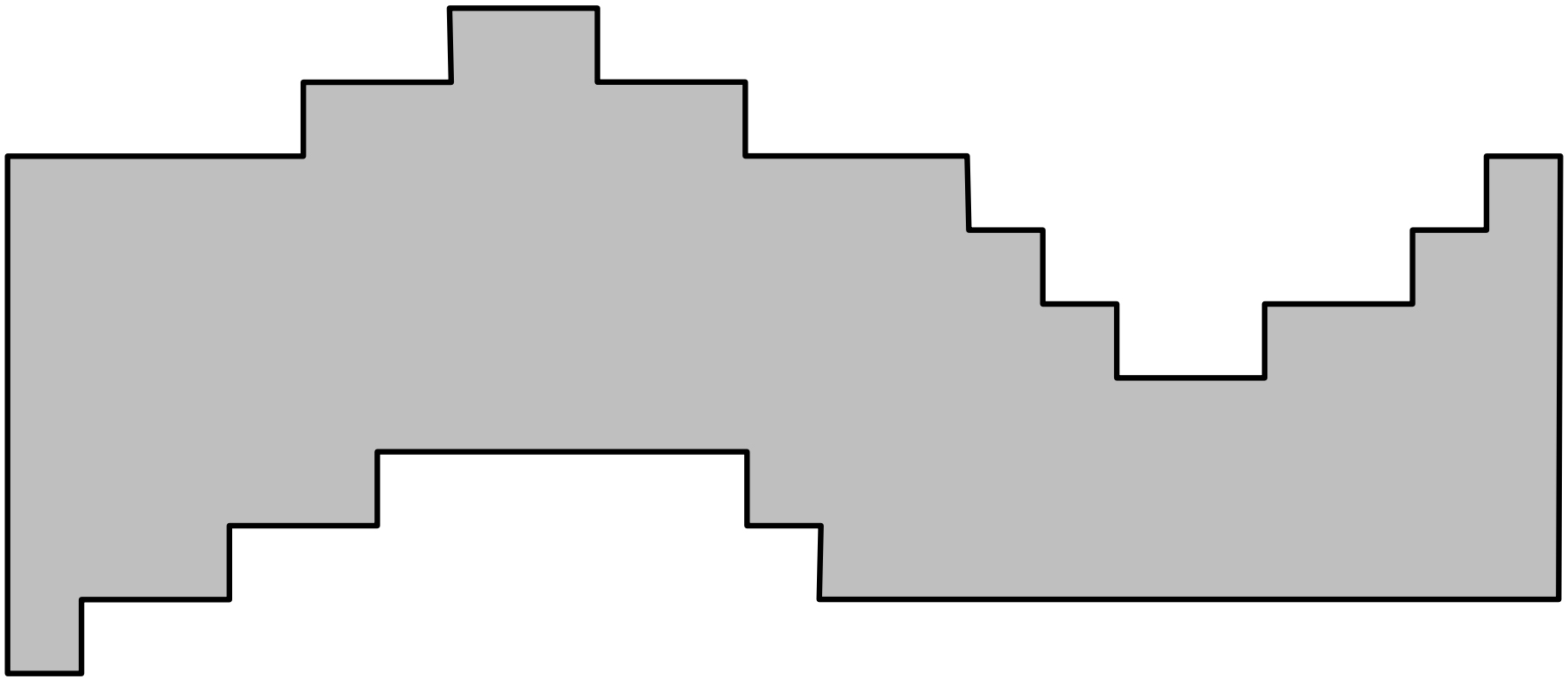


Canonical x-monotone Polygons



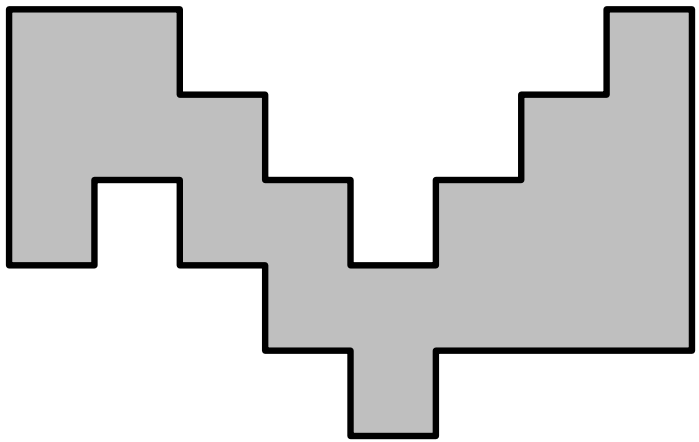
Canonical x-monotone Polygons

All vertical edges (except two extremes) are of unit-length



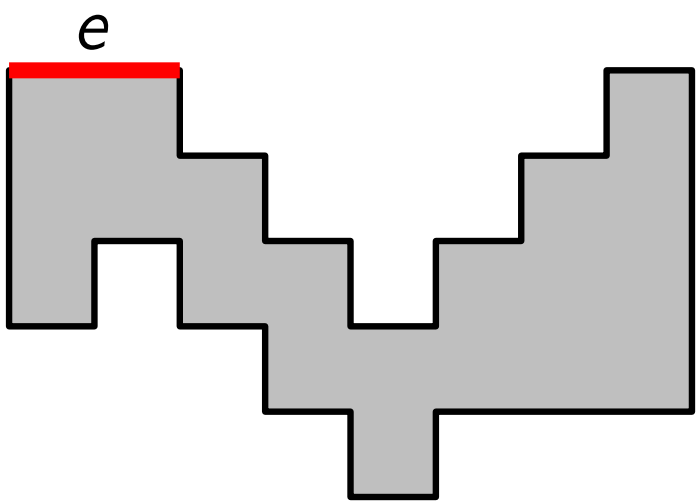
Canonical x-monotone Polygons

Assume that upper chain is longer (with more reflex vertices).



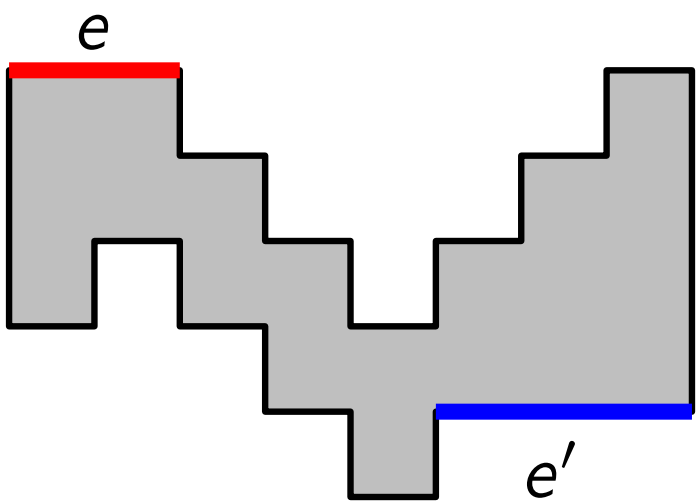
Canonical x-monotone Polygons

Assume that upper chain is longer (with more reflex vertices).



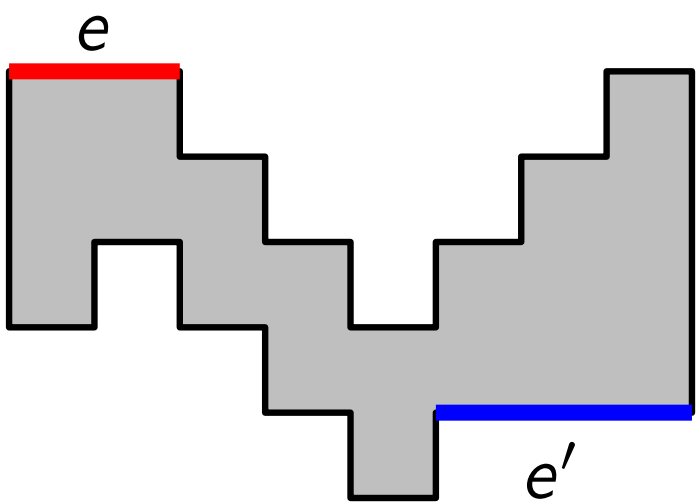
Canonical x-monotone Polygons

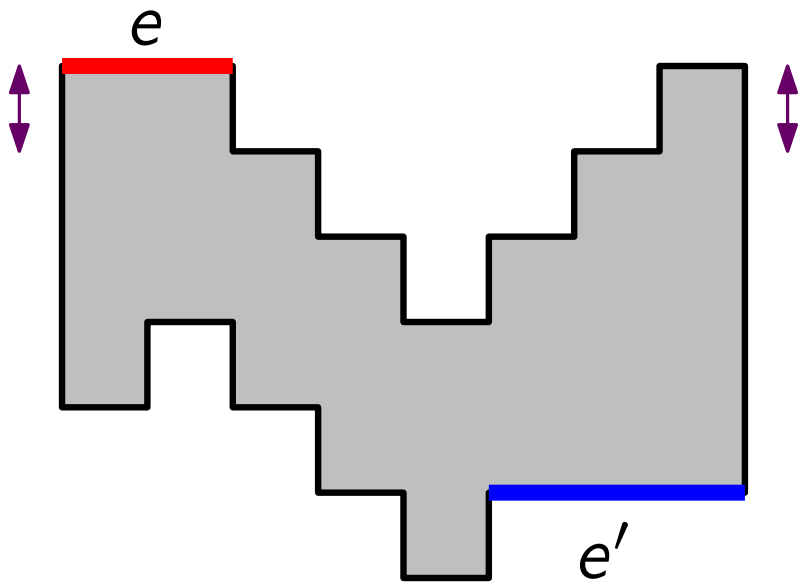
Assume that upper chain is longer (with more reflex vertices).



Canonical x-monotone Polygons

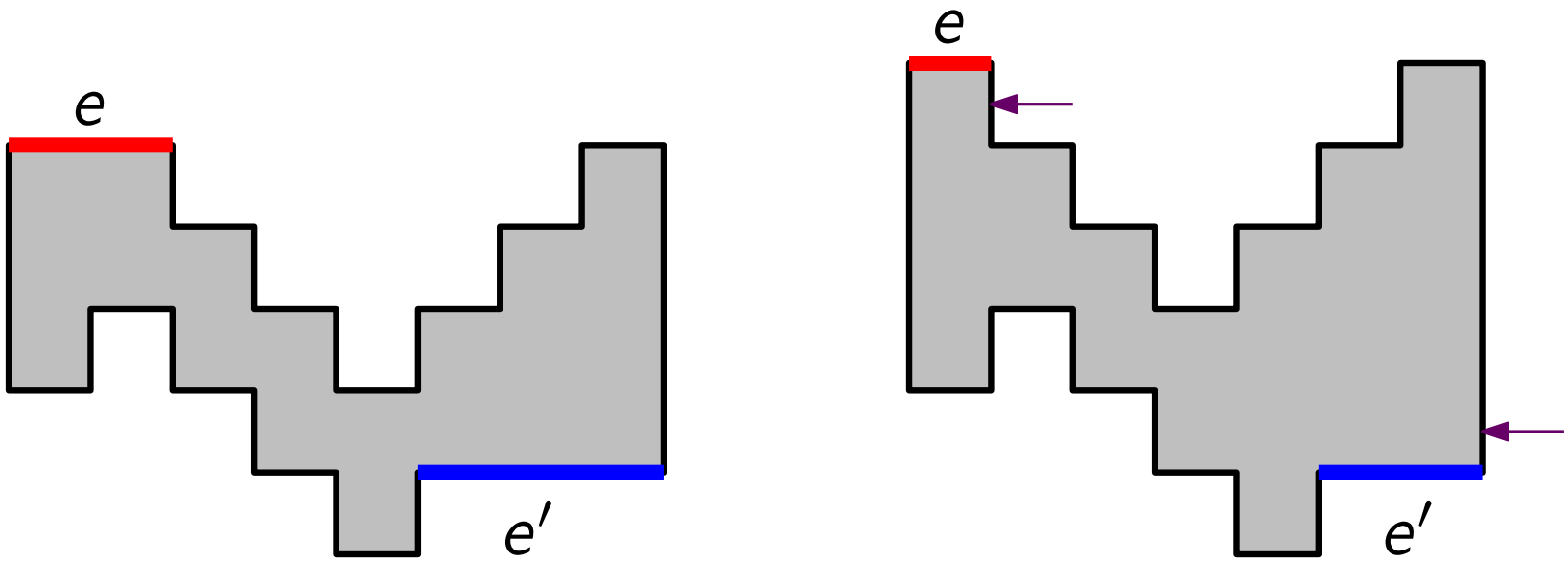
Assume that upper chain is longer (with more reflex vertices).





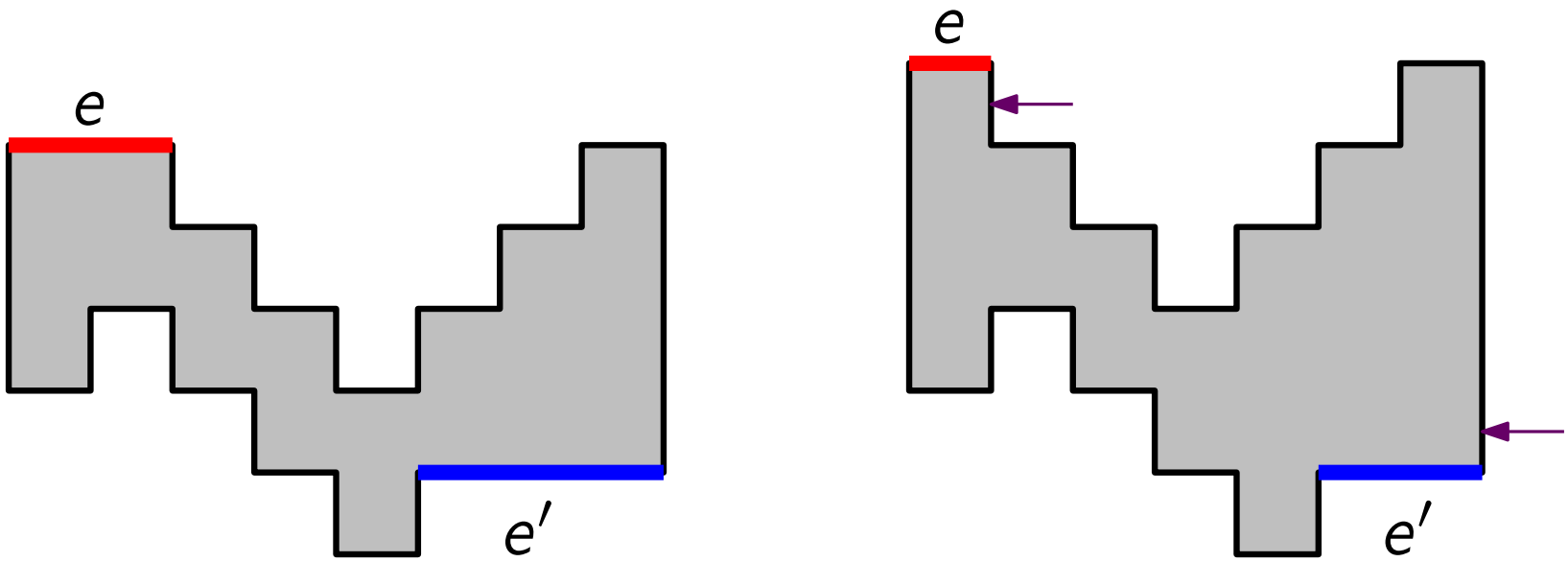
Canonical x-monotone Polygons

Assume that upper chain is longer (with more reflex vertices).



Canonical x-monotone Polygons

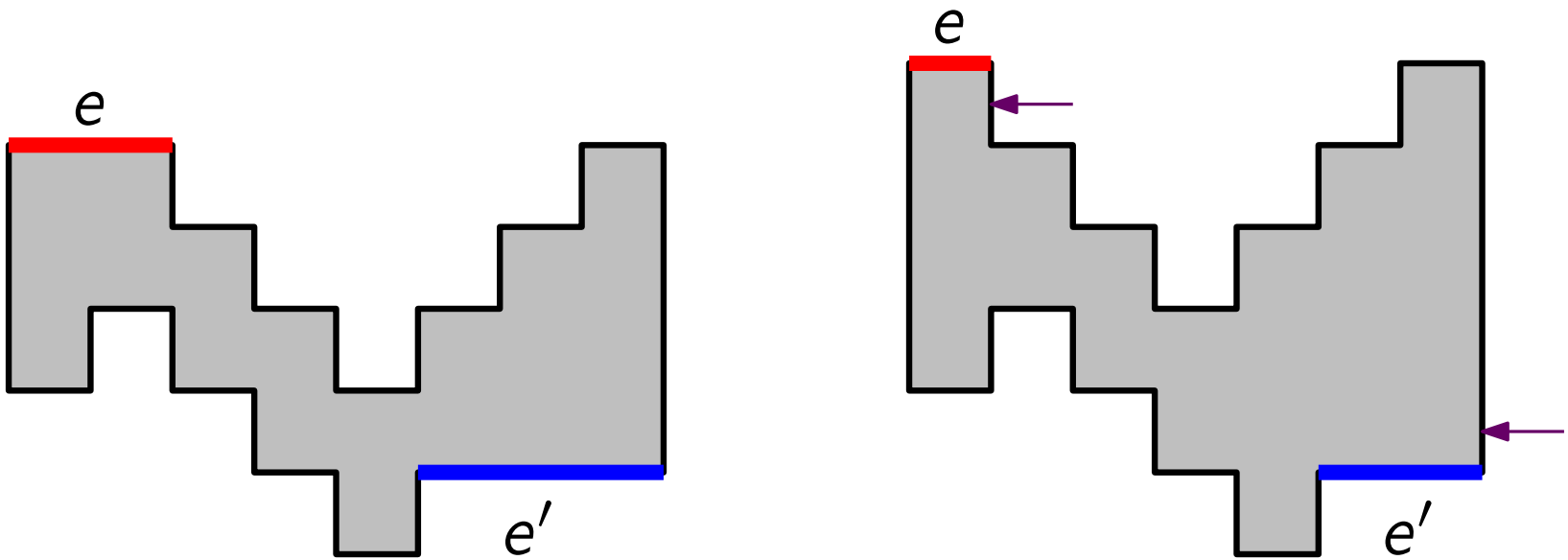
Assume that upper chain is longer (with more reflex vertices).



All horizontal edges in the longer chain are of unit-length

Canonical x-monotone Polygons

Assume that upper chain is longer (with more reflex vertices).

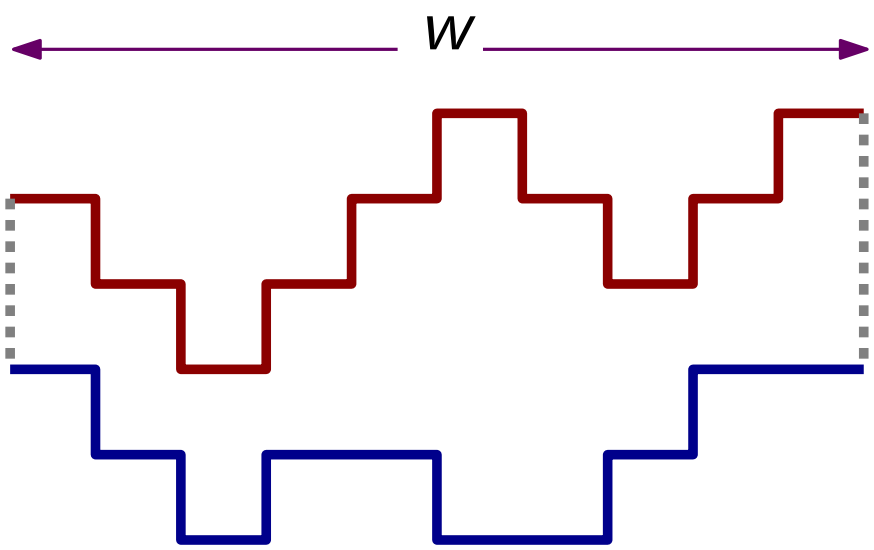


Canonical x-monotone polygon if

1. all vertical edges (except two extremes) are of unit-length, and
2. all horizontal edges in the longer chain are of unit-length

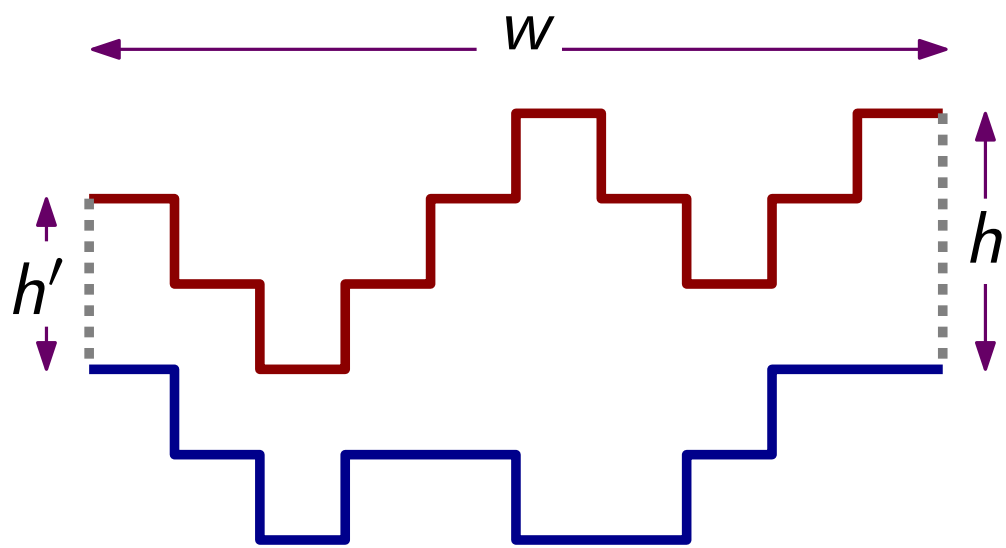
DP Algorithm

The upper (longer) chain is fixed, i.e., width w is fixed



DP Algorithm

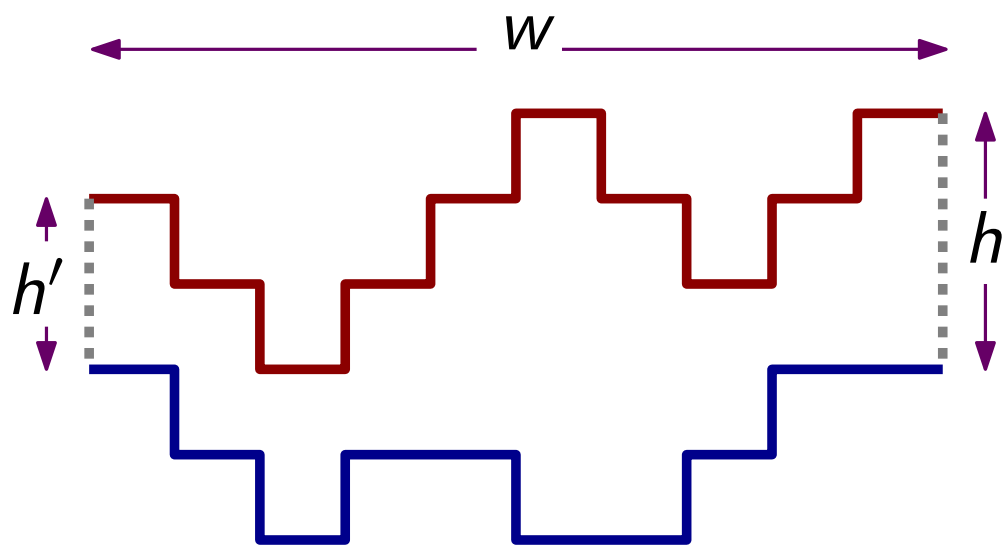
The upper (longer) chain is fixed, i.e., width w is fixed



perimeter $= 2w +$ (# of vertical edges) $+ h + h'$

DP Algorithm

The upper (longer) chain is fixed, i.e., width w is fixed

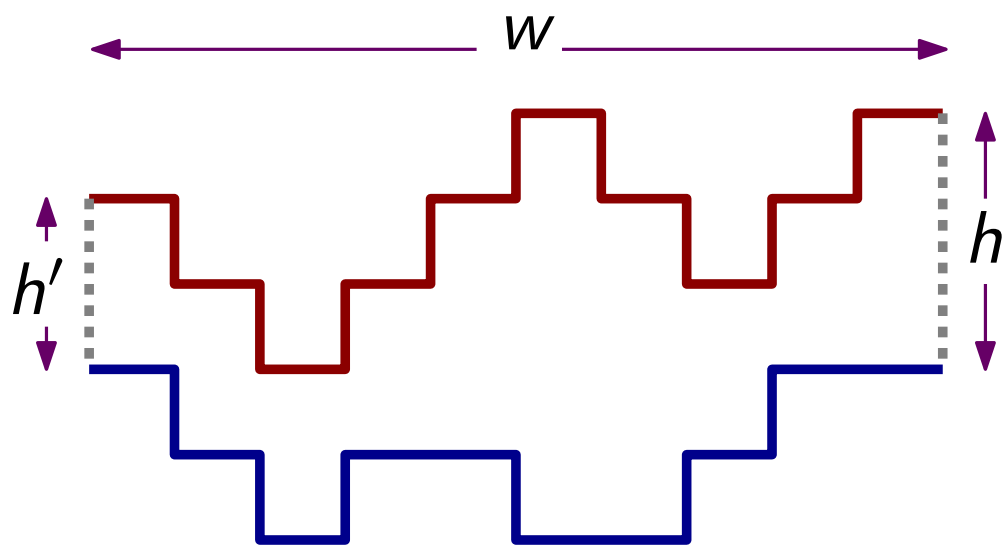


perimeter $= 2w +$ (# of vertical edges) $+ h + h'$

Unknowns: Height h (= length of the rightmost edge)

DP Algorithm

The upper (longer) chain is fixed, i.e., width w is fixed



perimeter $= 2w +$ (# of vertical edges) $+ h + h'$

Unknowns: Height h (= length of the rightmost edge)

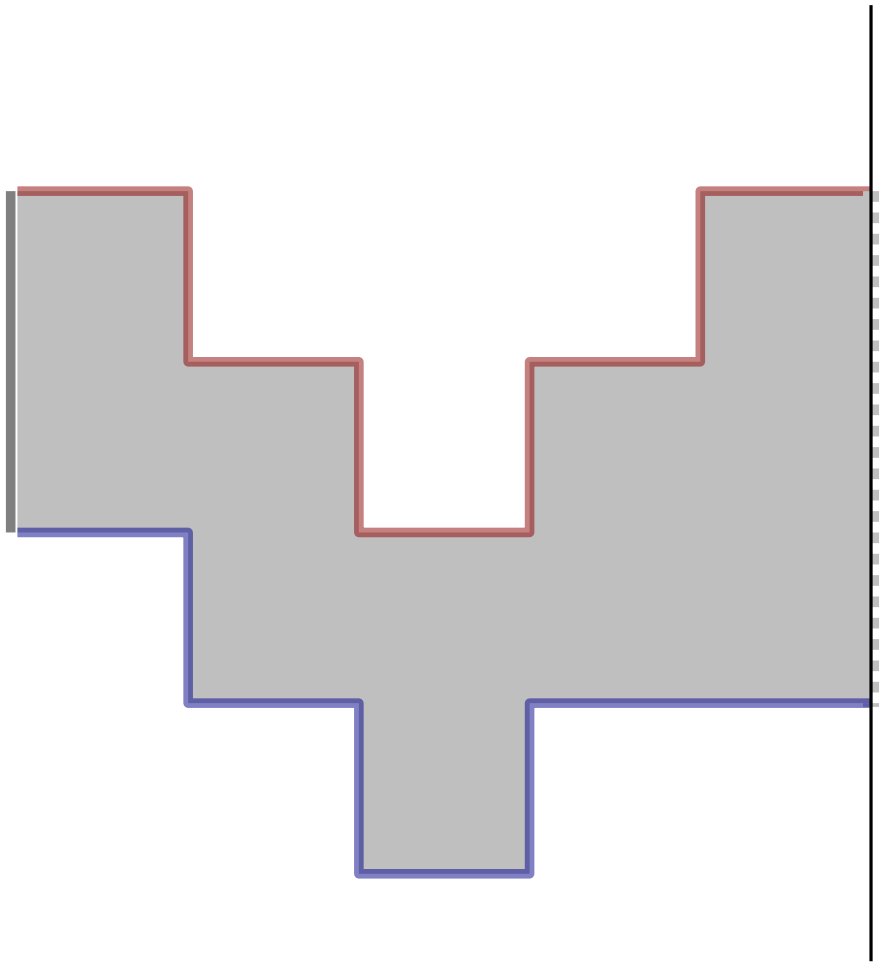
min perimeter = min h

DP Algorithm

$$\text{min perimeter} = \min h$$

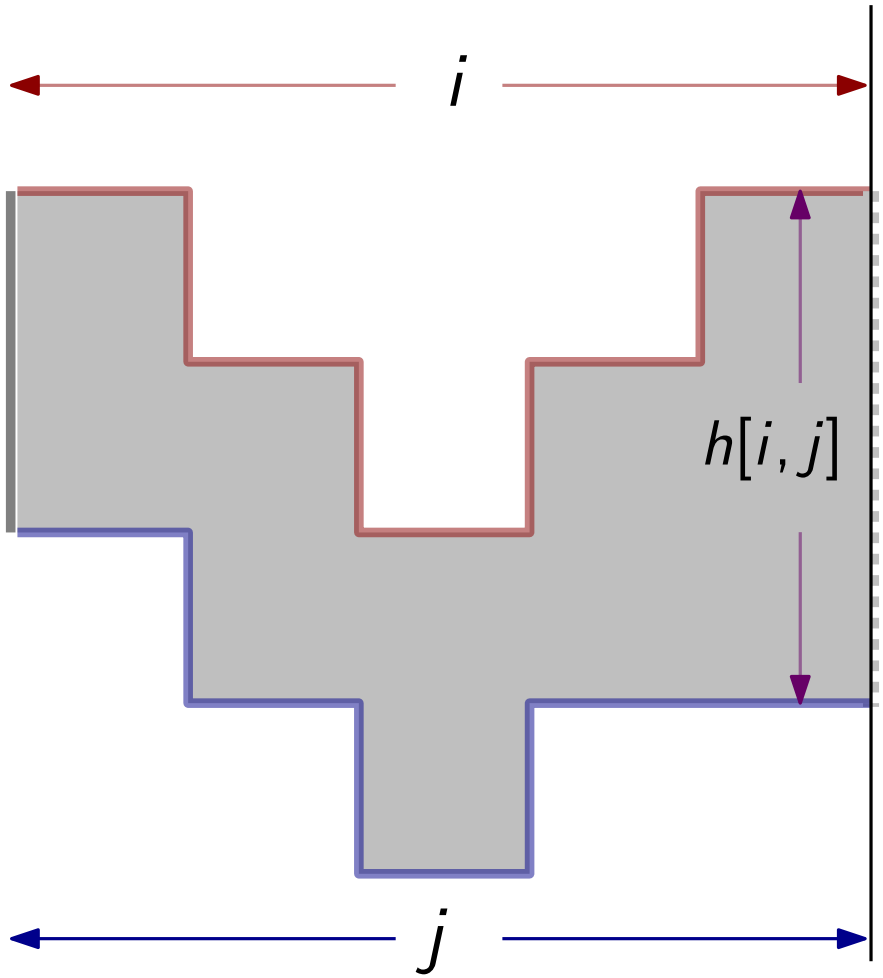
DP Algorithm

min perimeter $=$ min h



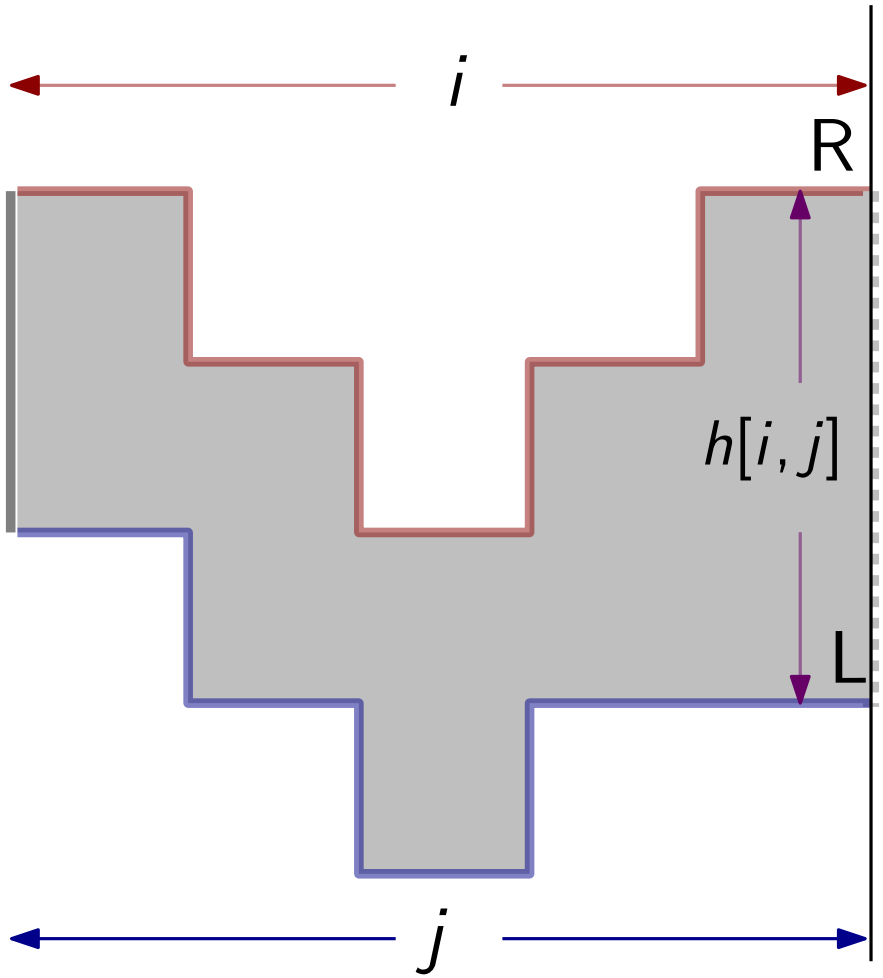
DP Algorithm

min perimeter = min h



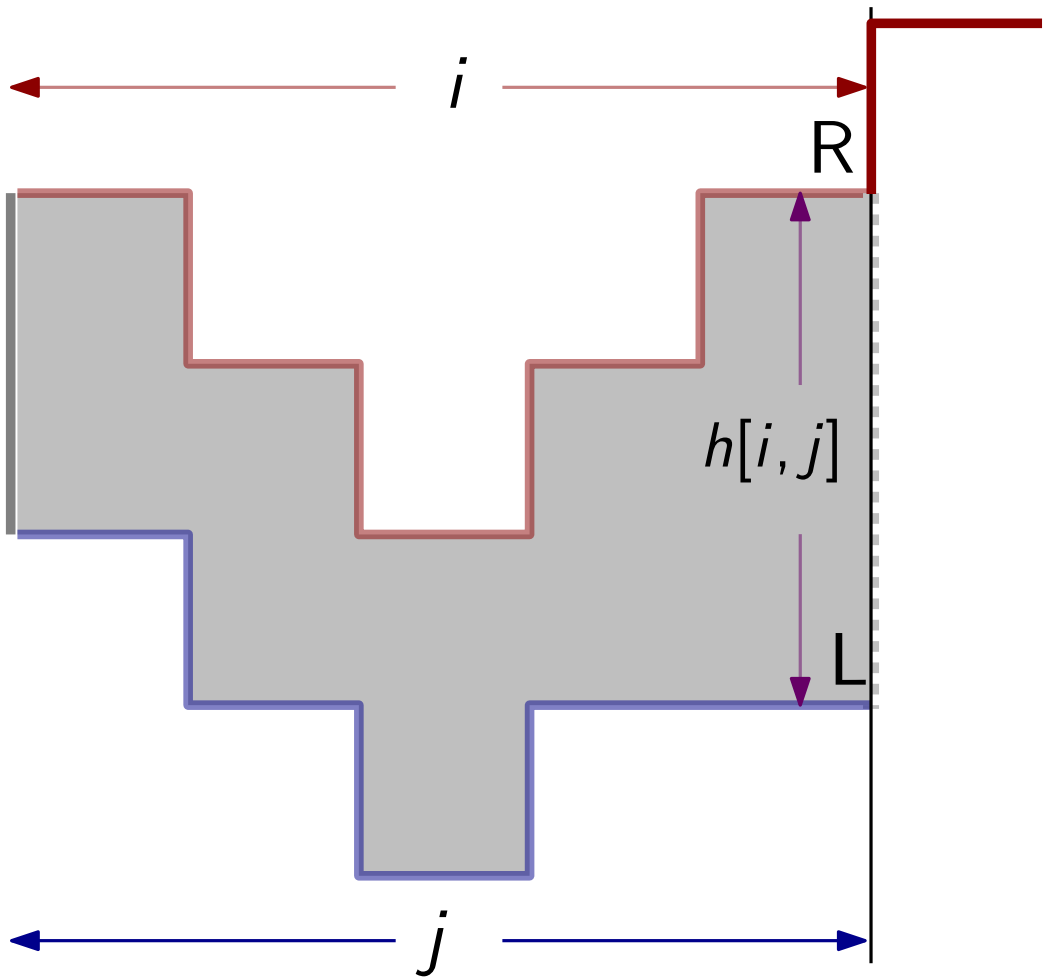
DP Algorithm

min perimeter = min h



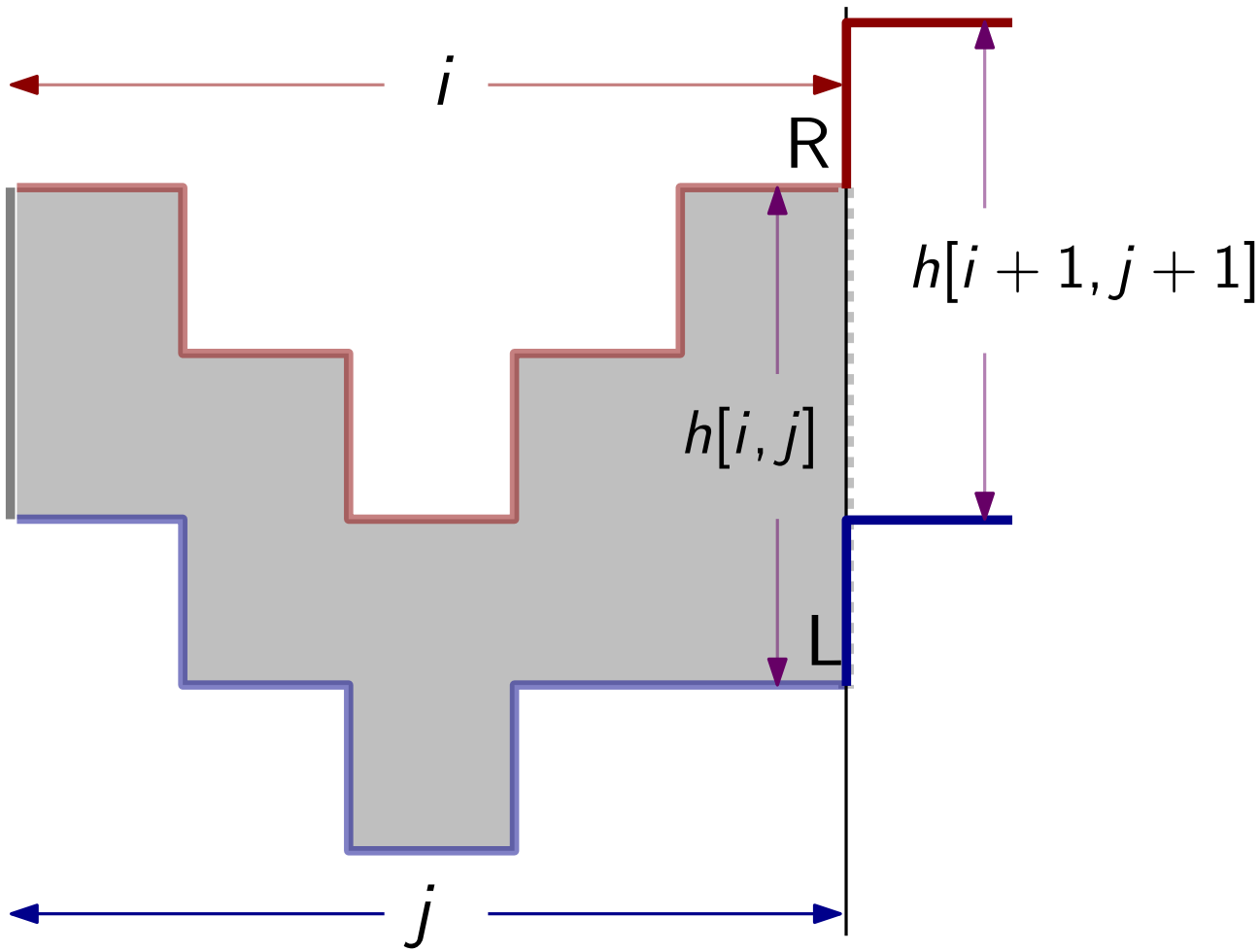
DP Algorithm

min perimeter = min h



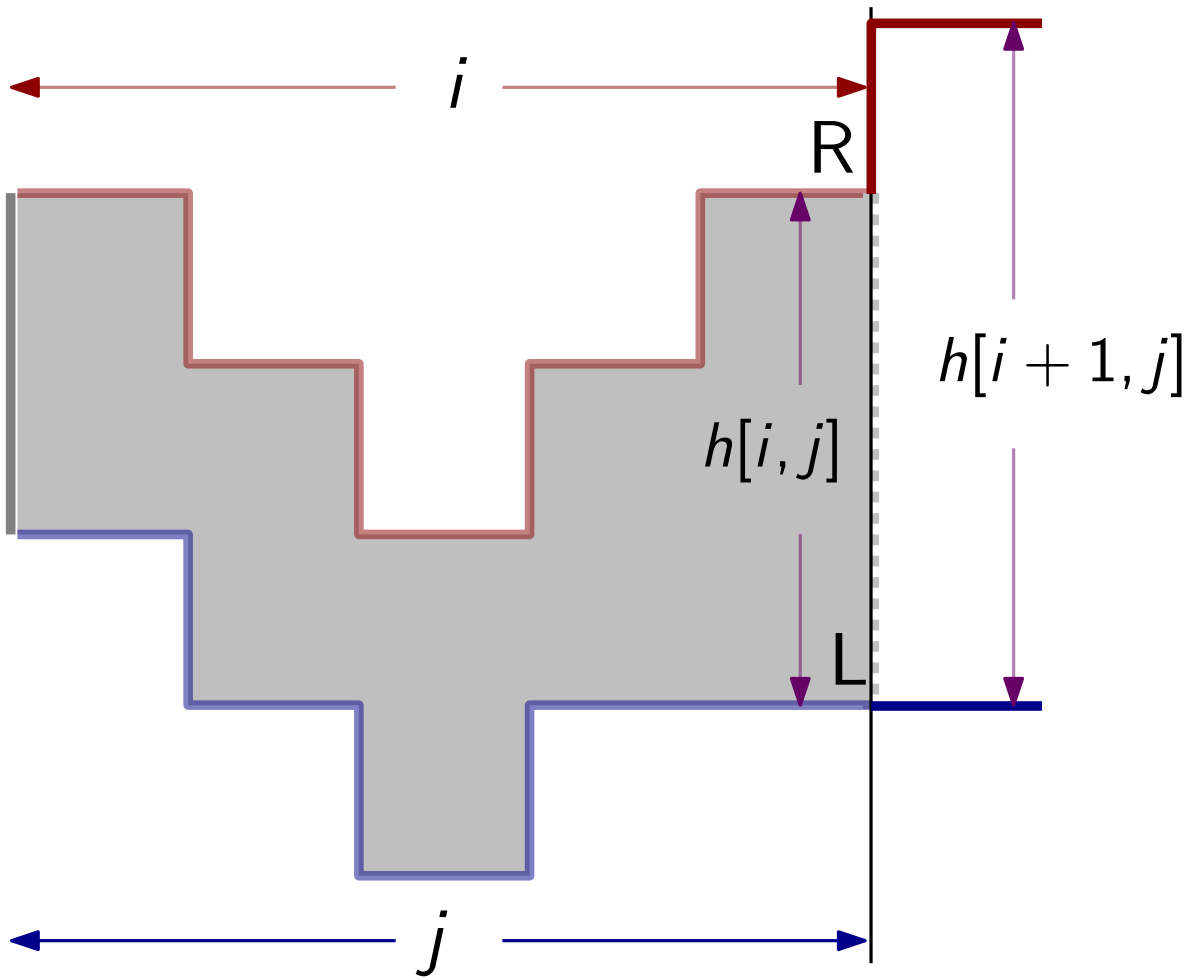
DP Algorithm

min perimeter = min h



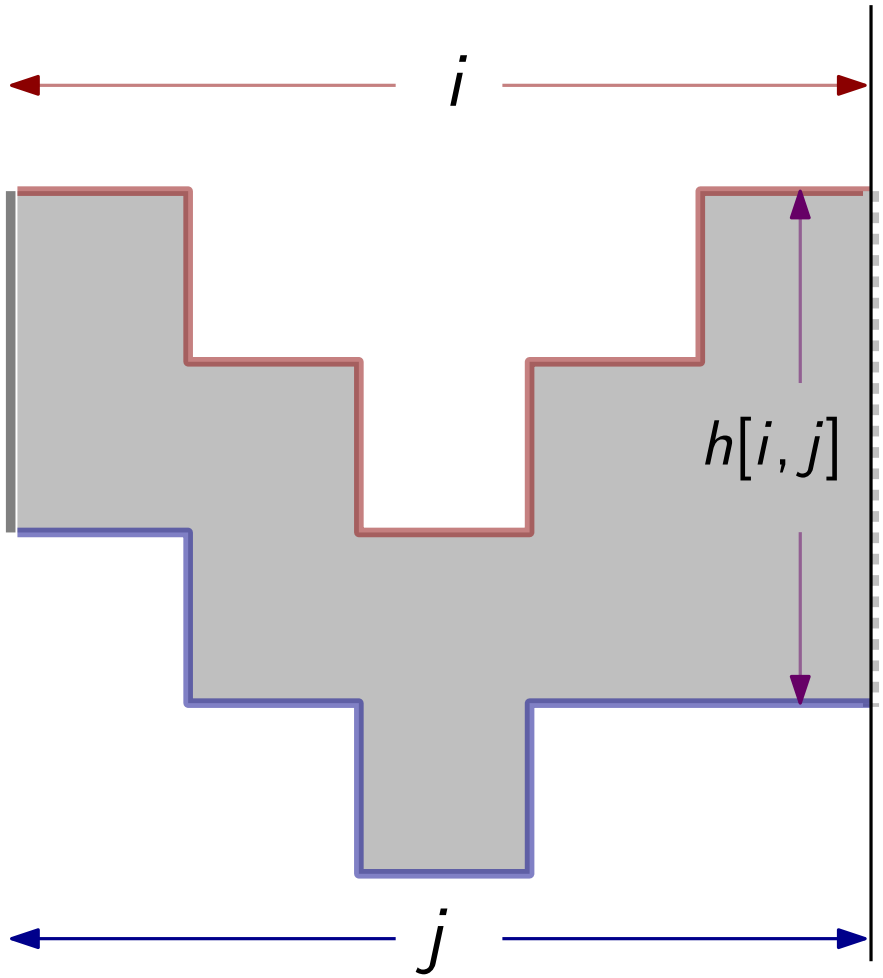
DP Algorithm

min perimeter = min h



DP Algorithm

min perimeter = min h

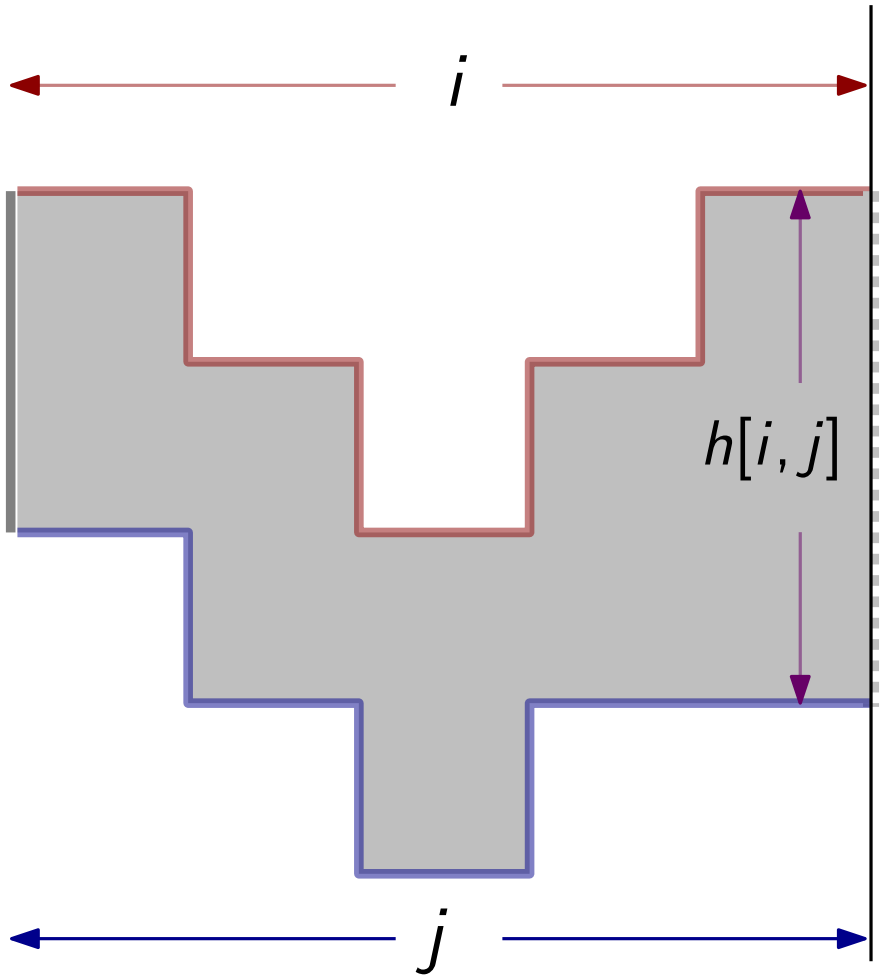


For (i,j) pairs:

Update $h[i,j]$

DP Algorithm

min perimeter = min h



For (i,j) pairs:	$O(n^2)$
Update $h[i,j]$	$O(1)$
	$O(n^2)$

Conclusions

1. Showed that realizing angles sequences optimally is NP-hard.
2. Designed algorithms for monotone sequences.

Conclusions

1. Showed that realizing angles sequences optimally is NP-hard.
2. Designed algorithms for monotone sequences.

Many questions...

1. Other sequences giving polynomial-time realization?

Conclusions

1. Showed that realizing angles sequences optimally is NP-hard.
2. Designed algorithms for monotone sequences.

Many questions...

1. Other sequences giving polynomial-time realization?
2. Approximation algorithms?

Conclusions

1. Showed that realizing angles sequences optimally is NP-hard.
2. Designed algorithms for monotone sequences.

Many questions...

1. Other sequences giving polynomial-time realization?
2. Approximation algorithms?
3. Angle sequences for the rectilinear chains?

Conclusions

1. Showed that realizing angles sequences optimally is NP-hard.
2. Designed algorithms for monotone sequences.

Many questions...

1. Other sequences giving polynomial-time realization?
2. Approximation algorithms?
3. Angle sequences for the rectilinear chains?

Thank you