

Integer Programs meet Fixed-Parameter Tractability

Alexandra Lassota (TU/e)

Integer Programming

$$Ax \leq b, x \in \mathbb{Z}^n$$

Integer Programming

Introduction and Relevance

$$Ax \leq b, x \in \mathbb{Z}^n$$

Integer Programming

~~Introduction and Relevance~~

$$Ax \leq b, x \in \mathbb{Z}^n$$

Integer Programming

A hand-drawn diagram illustrating a linear constraint in integer programming. It consists of three main components: a coefficient matrix, a variable vector, and a right-hand side vector, connected by a dot and a less-than-or-equal-to symbol.

- Coefficient Matrix (Blue Box):** A rectangular box containing the coefficients a_{11}, \dots, a_{1n} in the top row, a_{m1}, \dots, a_{mn} in the bottom row, and vertical ellipses in the middle to indicate intermediate rows and columns.
- Variable Vector (Yellow Box):** A vertical rectangular box containing the variables x_1 at the top and x_n at the bottom, with vertical ellipses in the middle.
- Right-Hand Side Vector (Green Box):** A vertical rectangular box containing the values b_1 at the top and b_m at the bottom, with vertical ellipses in the middle.
- Operators:** A dot (\cdot) is placed between the coefficient matrix and the variable vector, and a less-than-or-equal-to symbol (\leq) is placed between the variable vector and the right-hand side vector.

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Integer Programming

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

NP-hard -> exponential blow-up in runtime

How do we deal the NP-hardness

How do we deal the NP-hardness

- work on solvers

How do we deal the NP-hardness

- work on solvers
- heuristics

How do we deal the NP-hardness

- work on solvers
- heuristics
- approximation algorithms

How do we deal the NP-hardness

- work on solvers
- heuristics
- approximation algorithms
- focussing on sub-classes of the problem

How do we deal the NP-hardness

- work on solvers
- heuristics
- approximation algorithms
- focussing on sub-classes of the problem
- fixed-parameter tractability

How do we deal the NP-hardness

- work on solvers
- heuristics
- approximation algorithms
- focussing on sub-classes of the problem
- fixed-parameter tractability

Fixed-Parameter Tractability

Given: Problem P with parameter k

Fixed-Parameter Tractability

Given: Problem P with parameter k

If we can solve any instance of P in time

$$f(k) \cdot \text{poly}(|I|)$$

then P is in FPT/ P is fixed-parameter tractable w.r.t. k

Fixed-Parameter Tractability - Example

Given:

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}, x \in \{0,1\}^n \text{ with parameter } n$$

Fixed-Parameter Tractability - Example

Given:

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}, x \in \{0,1\}^n \text{ with parameter } n$$

Instance size: $O(nm \log(m\Delta))$

$\uparrow \|A\|_\infty$

Fixed-Parameter Tractability - Example

Given:

$$\begin{array}{|c|} \hline a_{11} \quad \dots \quad a_{1n} \\ \hline \vdots \quad \ddots \quad \vdots \\ \hline a_{m1} \quad \dots \quad a_{mn} \\ \hline \end{array} \cdot \begin{array}{|c|} \hline x_1 \\ \hline \vdots \\ \hline x_n \\ \hline \end{array} \leq \begin{array}{|c|} \hline b_1 \\ \hline \vdots \\ \hline b_m \\ \hline \end{array}, x \in \{0,1\}^n \text{ with parameter } n$$

Instance size: $O(nm \log(m\Delta))$

$\uparrow \|A\|_\infty$

Algorithm: Guess and test solution; $\underbrace{2^n}_{f(n)} \cdot \underbrace{O(nm)}_{\text{poly}(|I|)}$

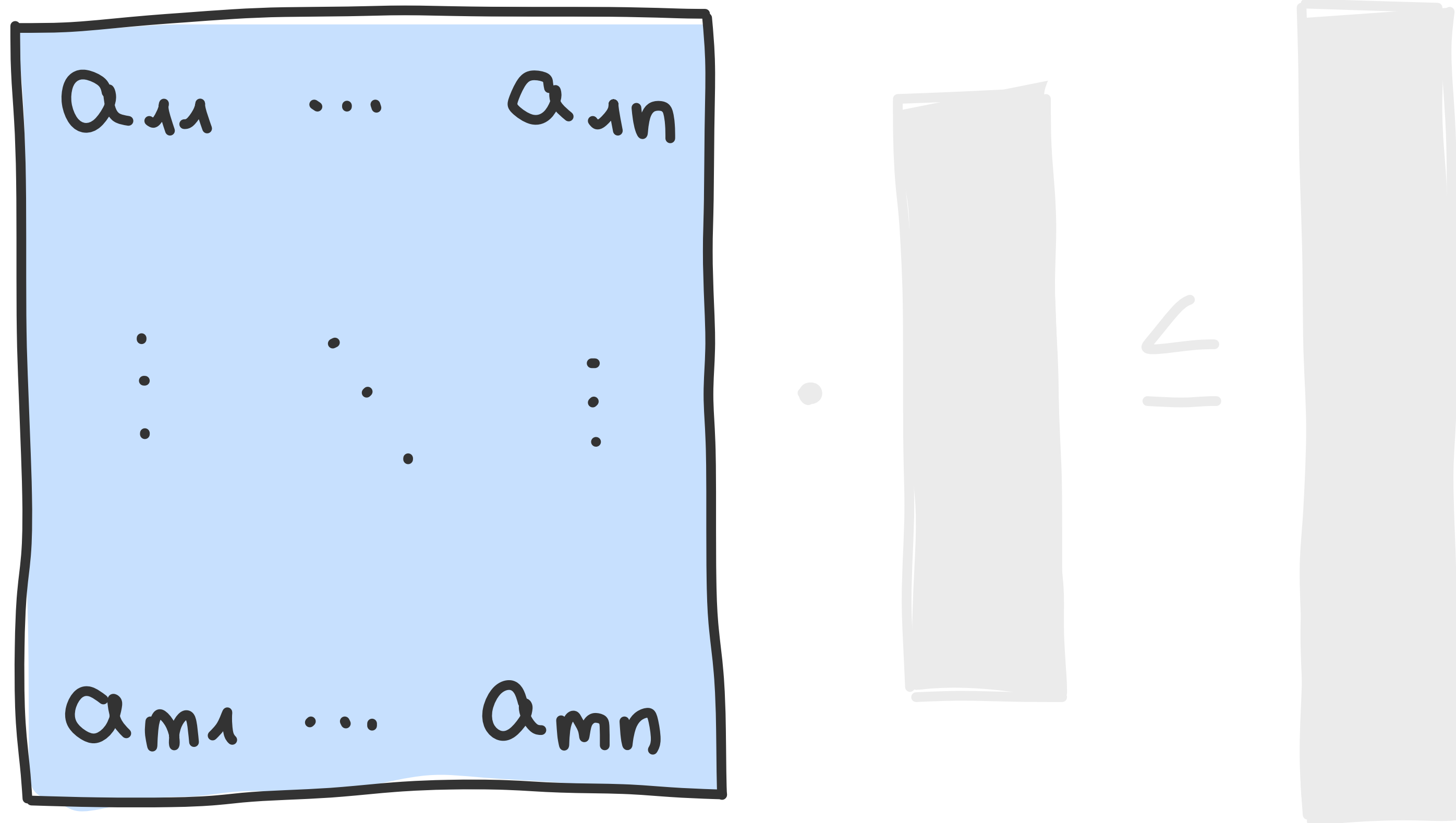
Integer Programming meets FPT

A hand-drawn diagram illustrating a linear constraint in integer programming. It consists of three main components: a coefficient matrix, a variable vector, and a right-hand side vector, connected by a dot product and an inequality symbol.

- Coefficient Matrix (Blue Box):** A rectangular box containing the coefficients a_{11}, \dots, a_{1n} in the top row, a_{m1}, \dots, a_{mn} in the bottom row, and vertical ellipses in the middle to indicate intermediate rows and columns.
- Variable Vector (Yellow Box):** A vertical rectangular box containing the variables x_1 at the top and x_n at the bottom, with vertical ellipses in the middle.
- Right-Hand Side Vector (Green Box):** A vertical rectangular box containing the values b_1 at the top and b_m at the bottom, with vertical ellipses in the middle.
- Operators:** A dot (\cdot) is placed between the coefficient matrix and the variable vector, and a less-than-or-equal-to symbol (\leq) is placed between the variable vector and the right-hand side vector.

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

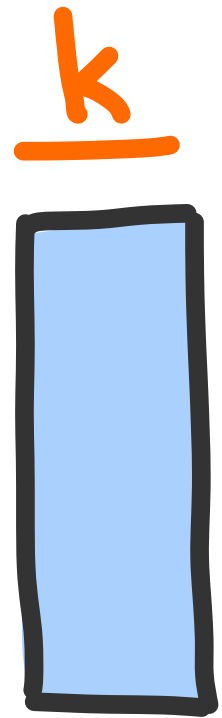
Integer Programming meets FPT



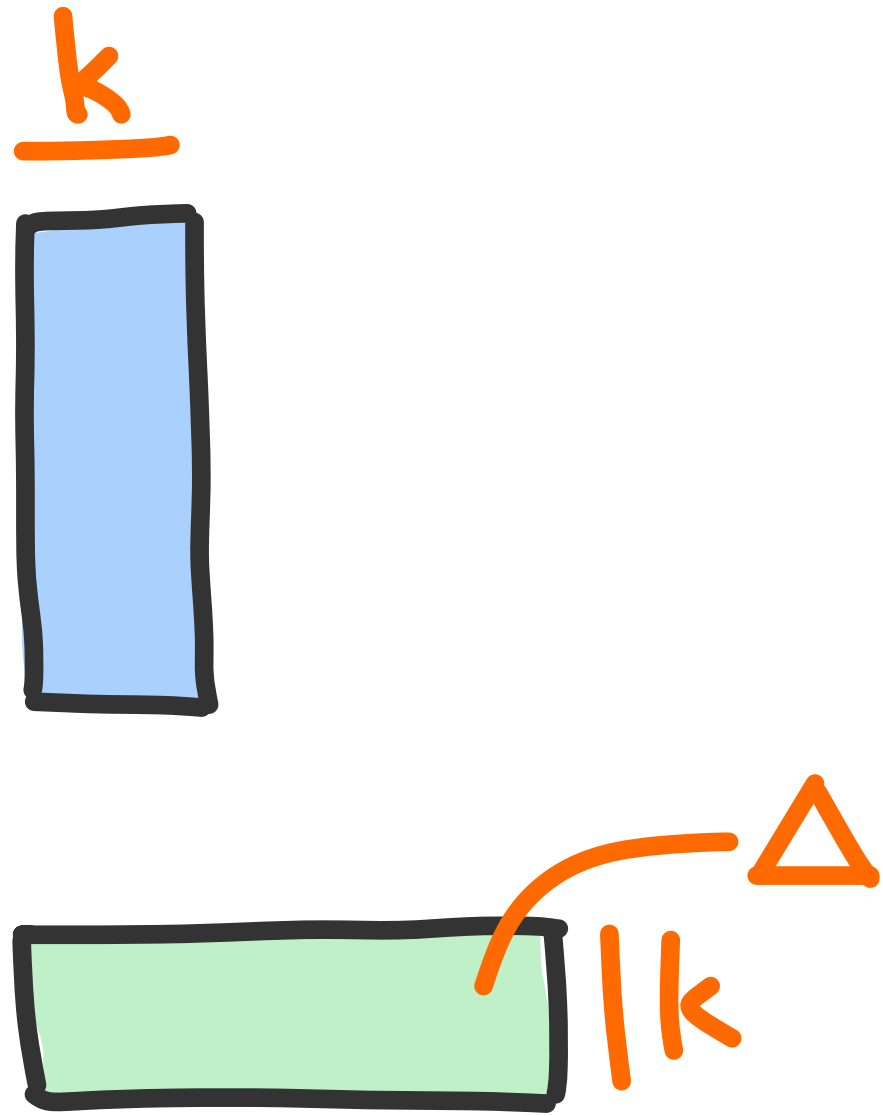
A hand-drawn diagram illustrating the multiplication of two matrices. On the left, a light blue square represents matrix A , with its top row labeled $a_{11} \dots a_{1n}$ and its bottom row labeled $a_{m1} \dots a_{mn}$. In the center, a small dot represents the multiplication operation. To the right of the dot is a light gray rectangle representing matrix B , followed by an equals sign, and then another light gray rectangle representing the resulting matrix C .

$$\begin{matrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{matrix} \cdot \text{[gray box]} = \text{[gray box]}$$

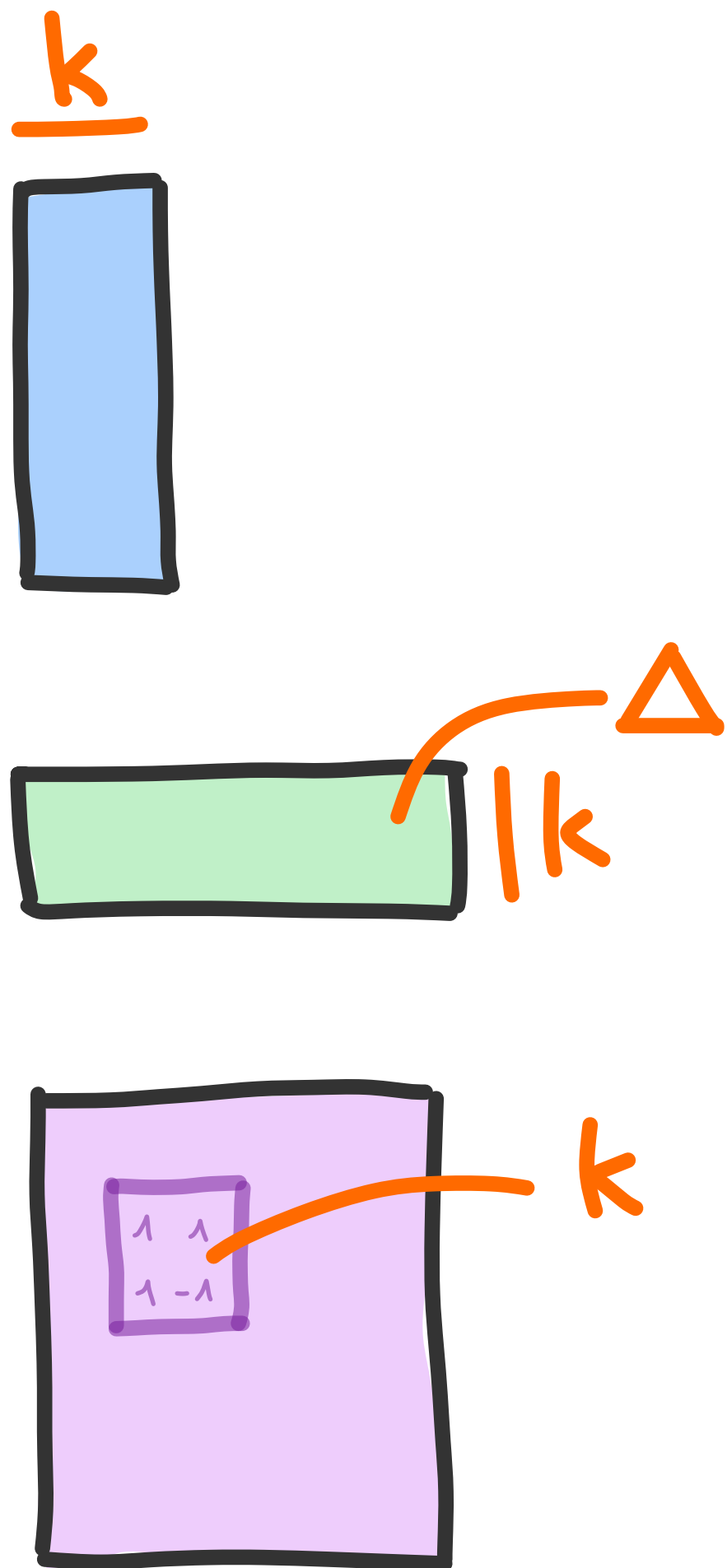
Integer Programming meets FPT



Integer Programming meets FPT

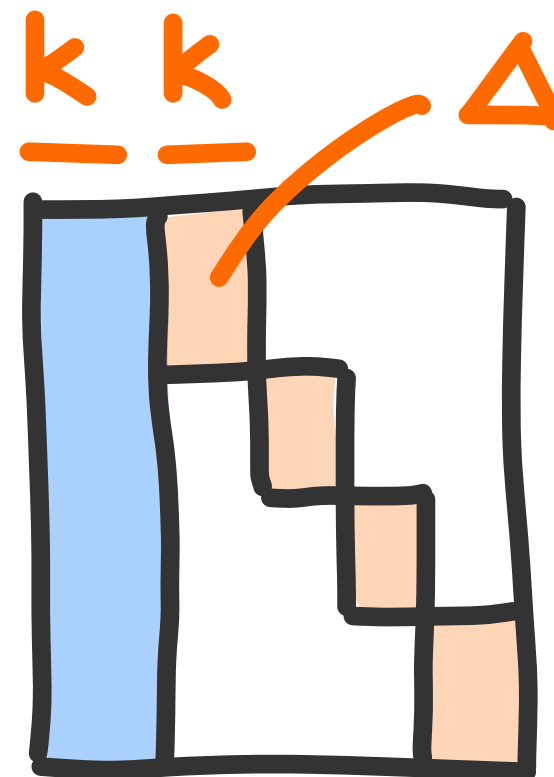
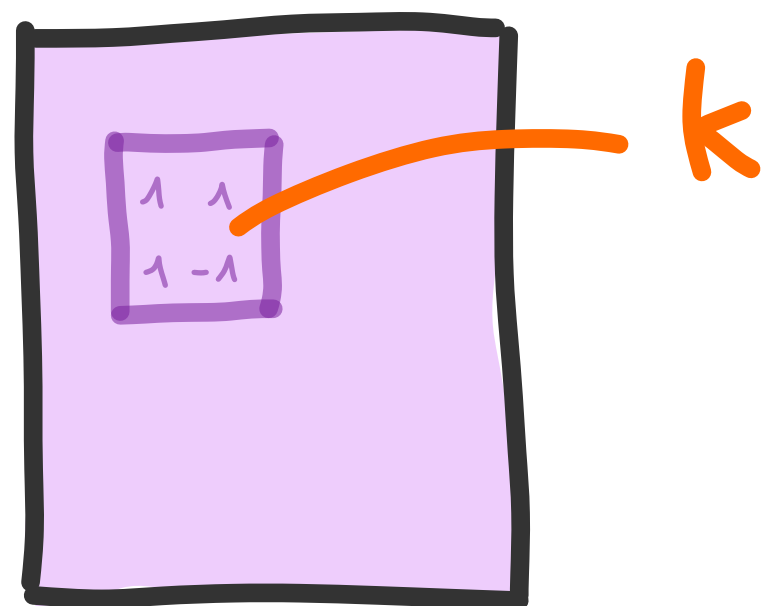
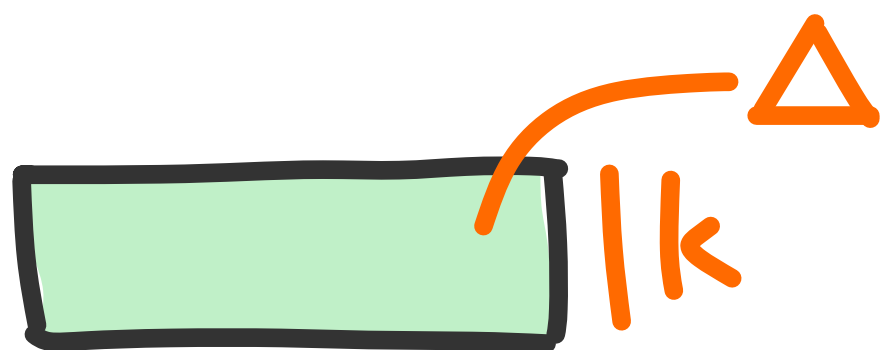
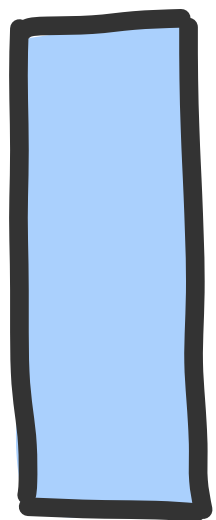


Integer Programming meets FPT

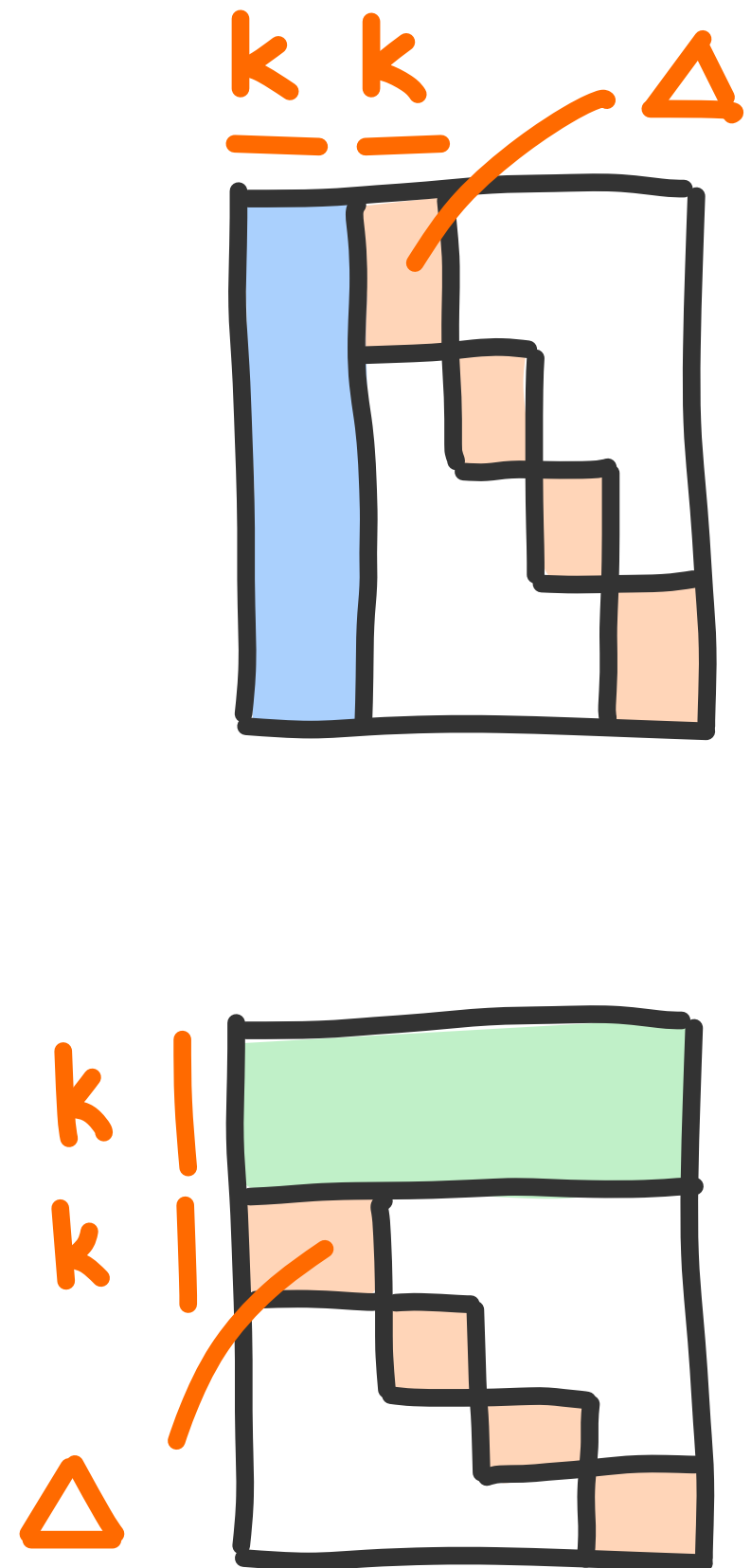
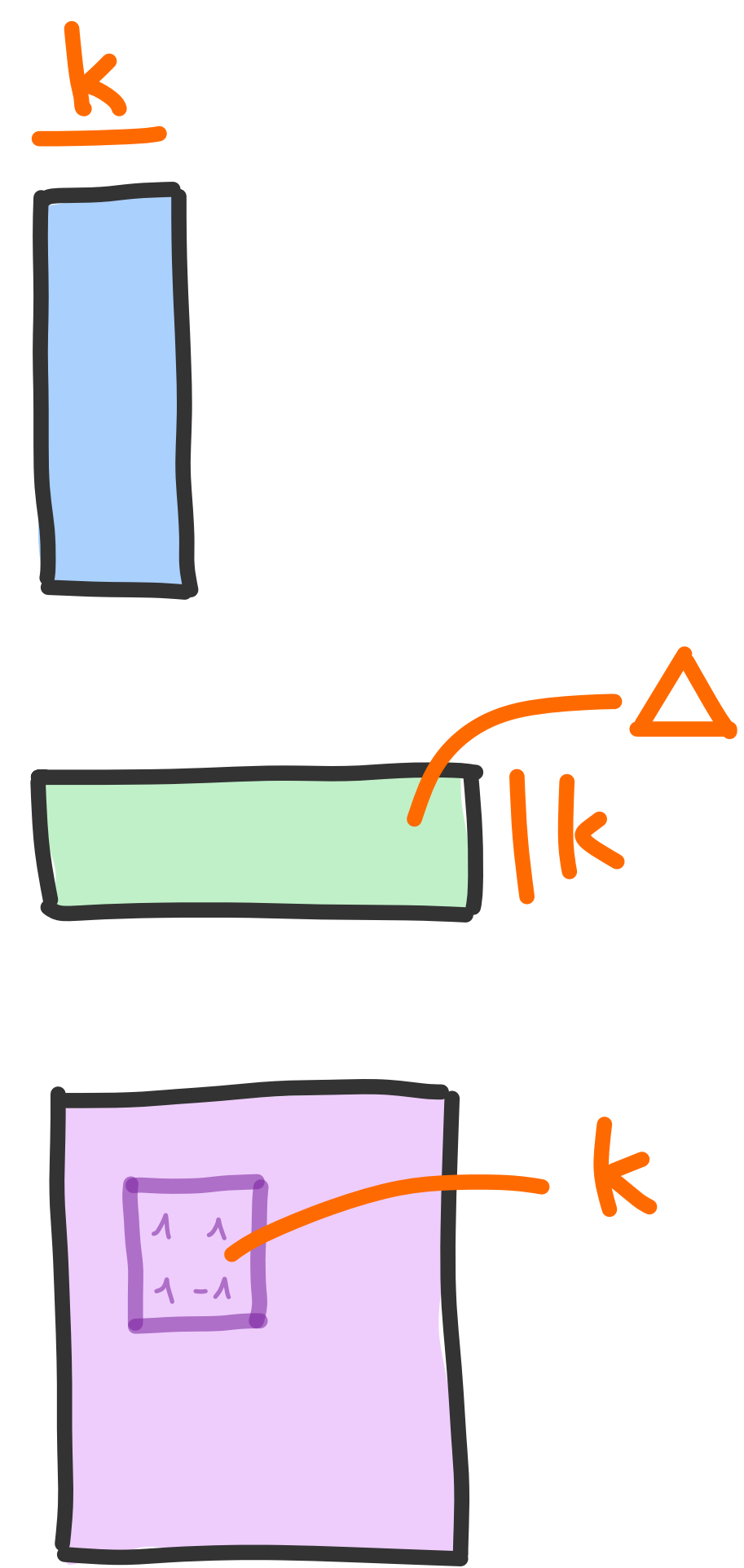


Integer Programming meets FPT

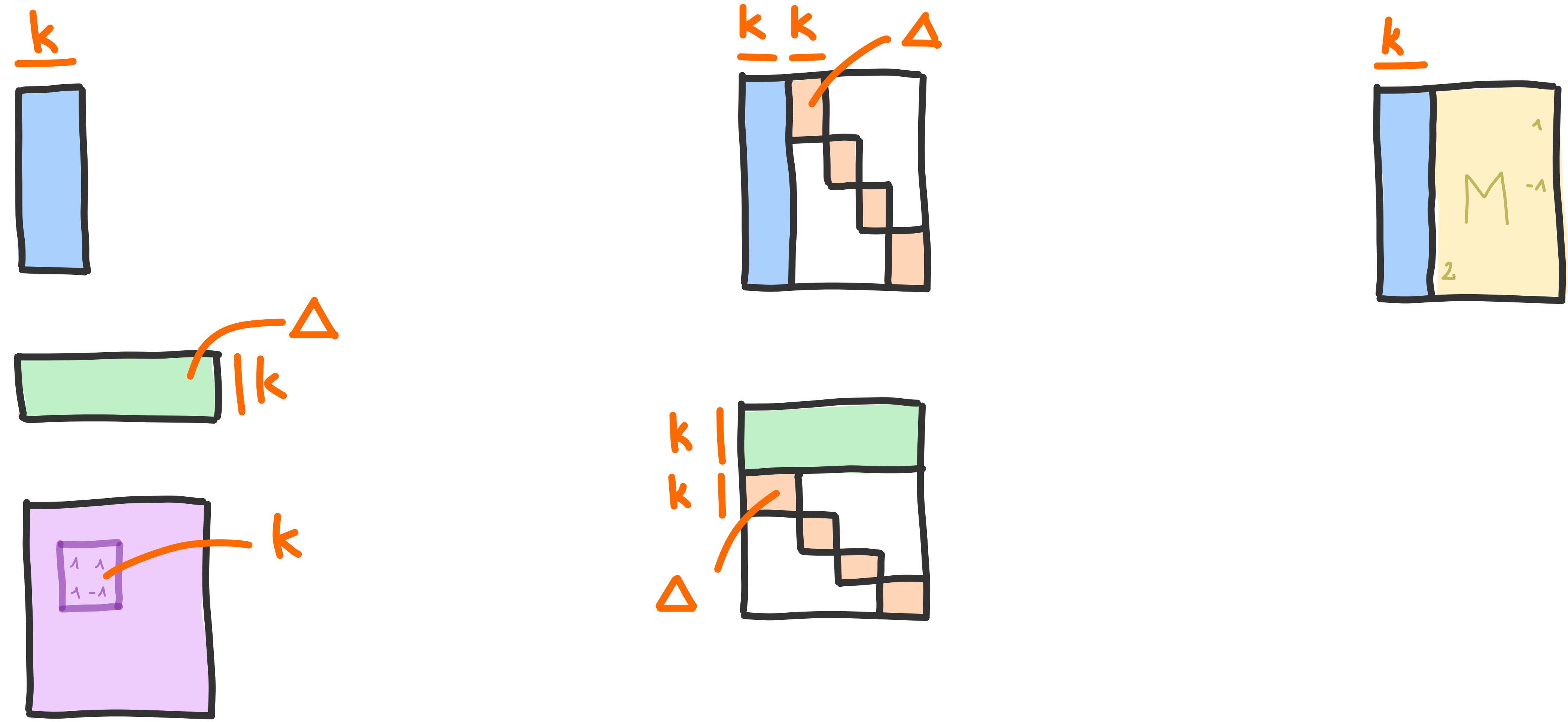
k



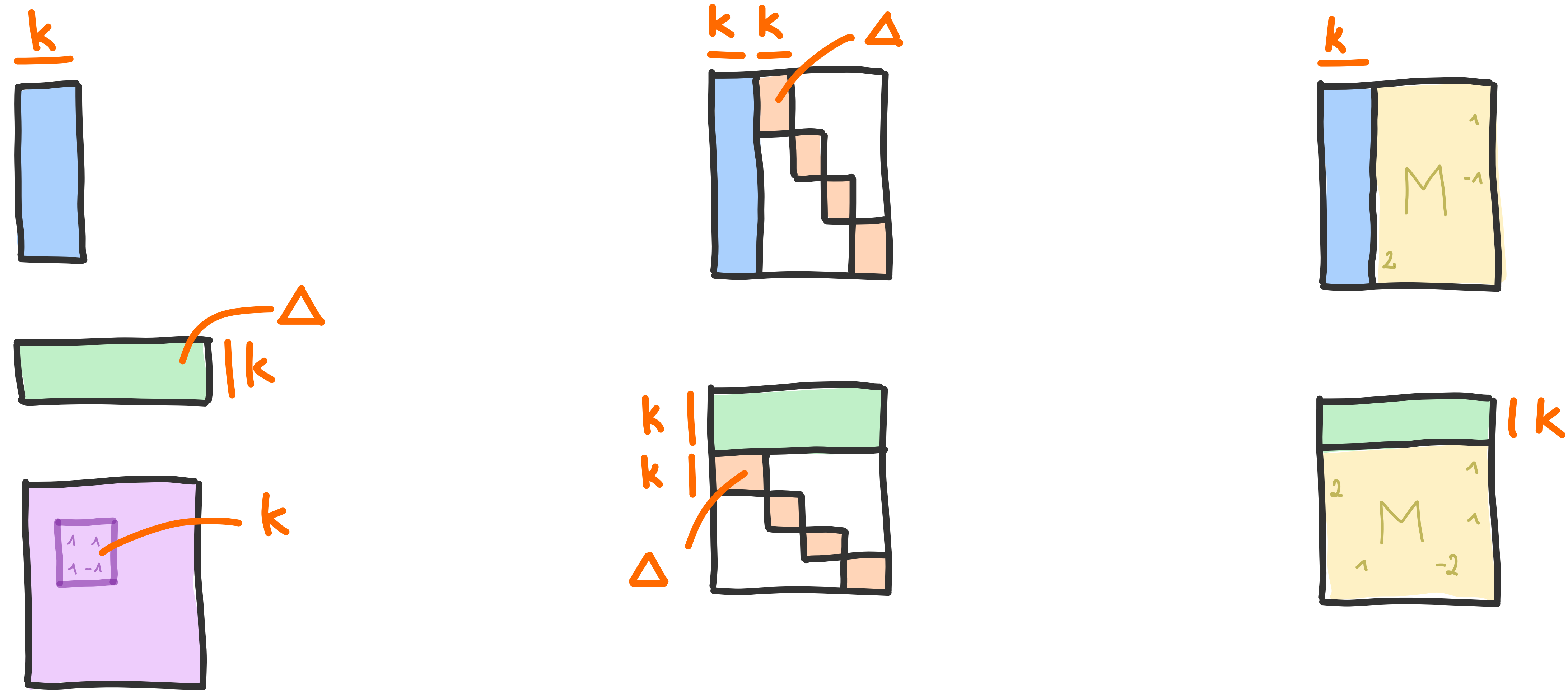
Integer Programming meets FPT



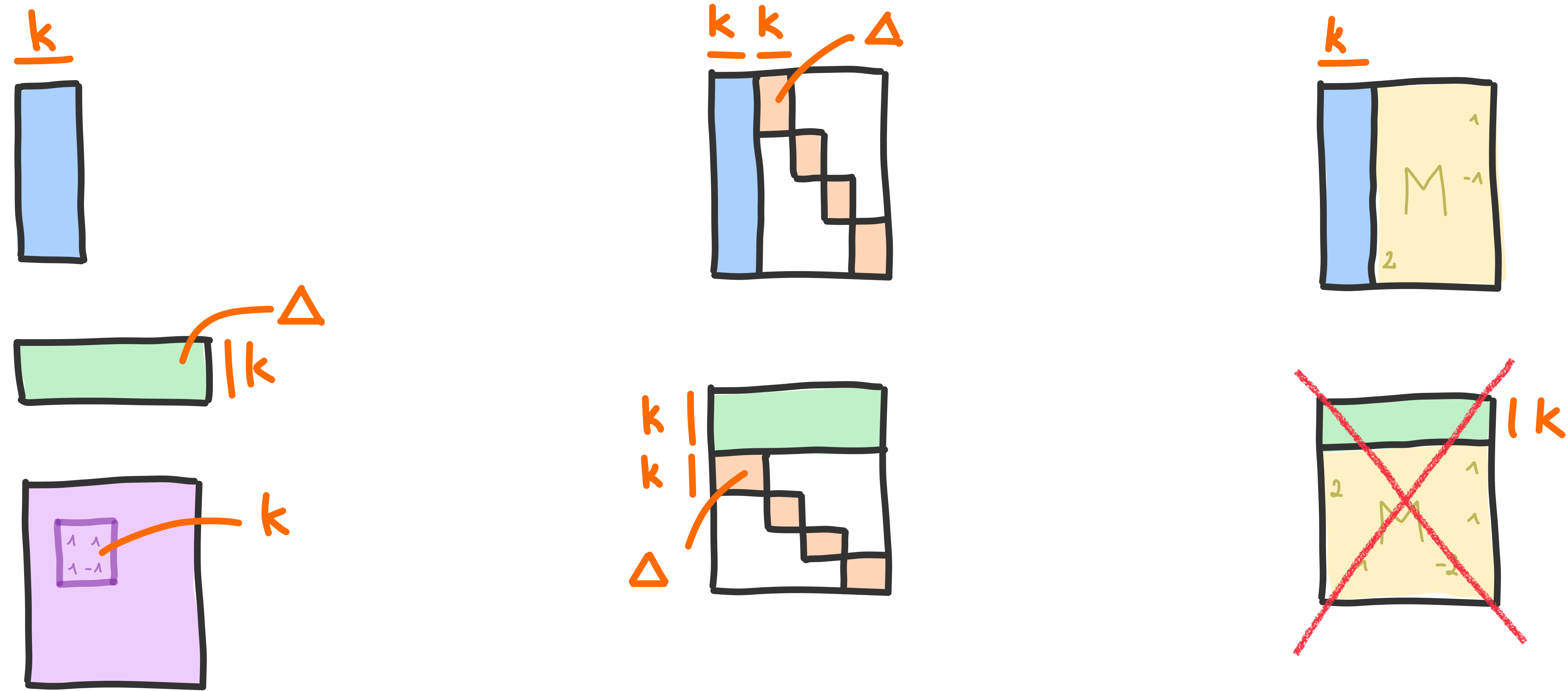
Integer Programming meets FPT



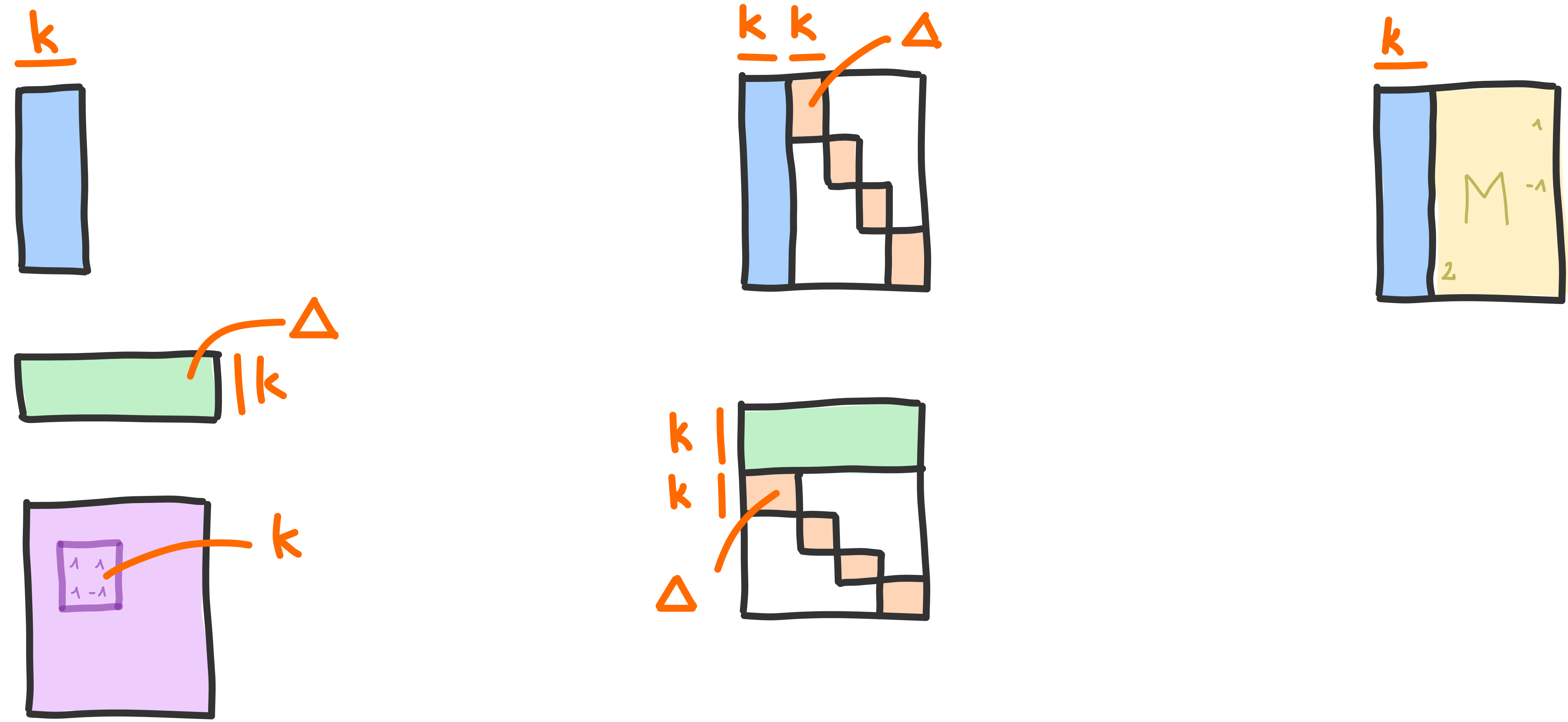
Integer Programming meets FPT



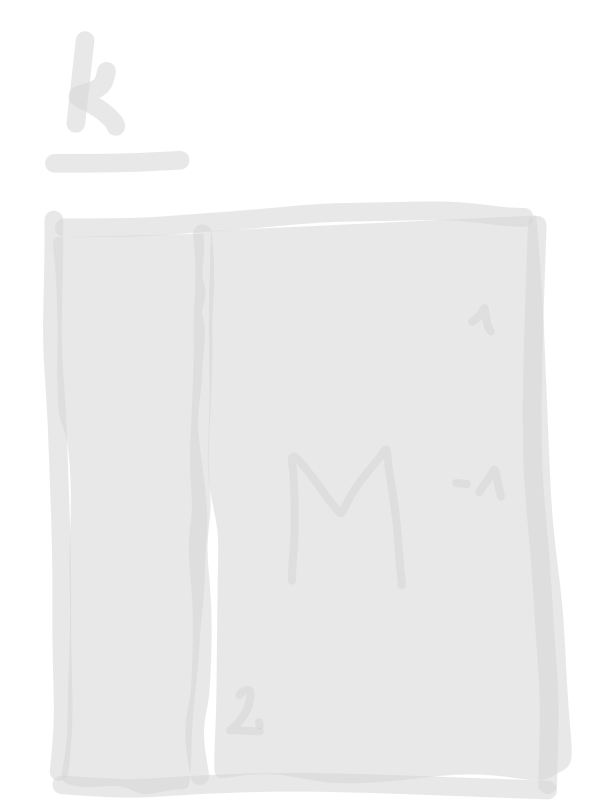
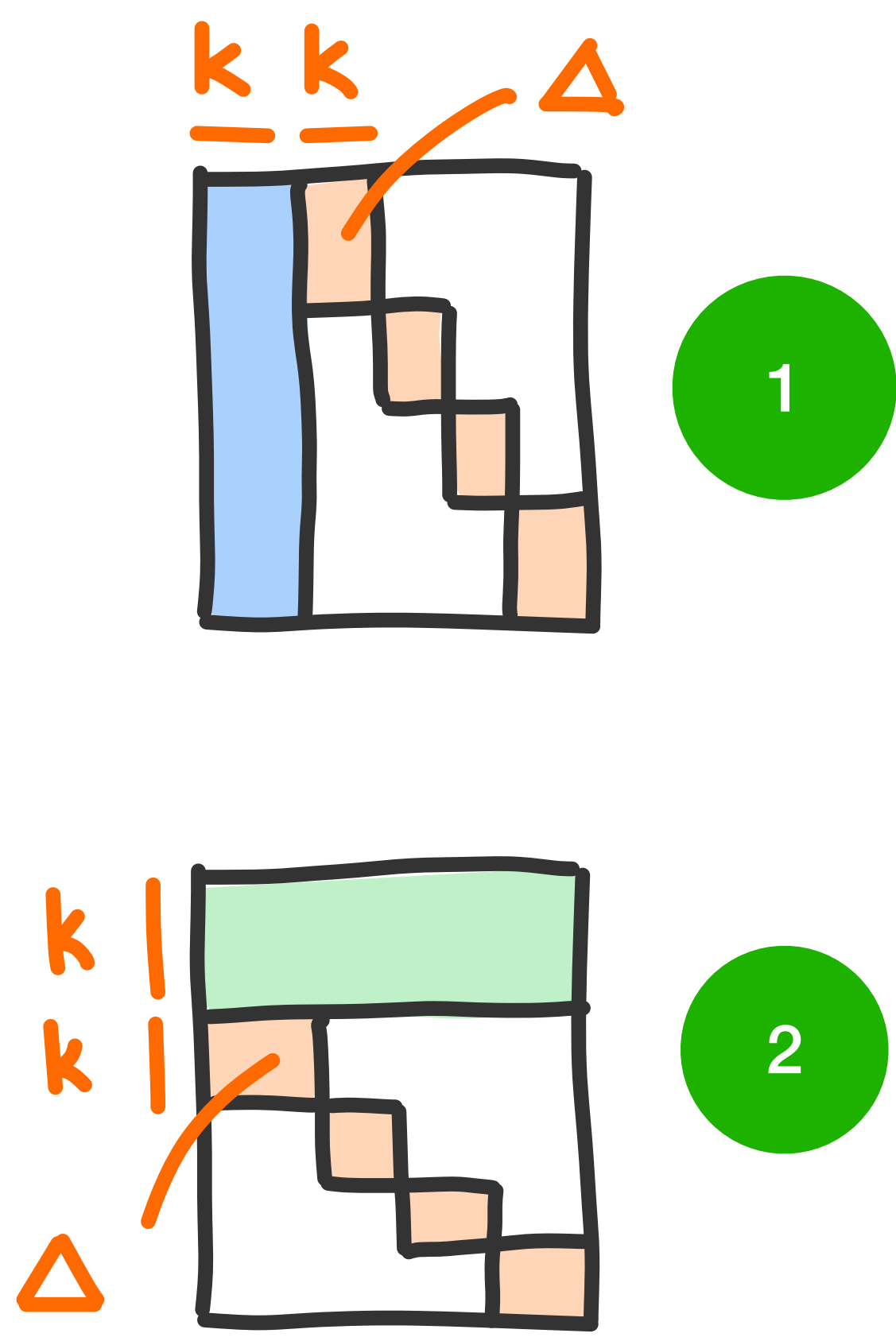
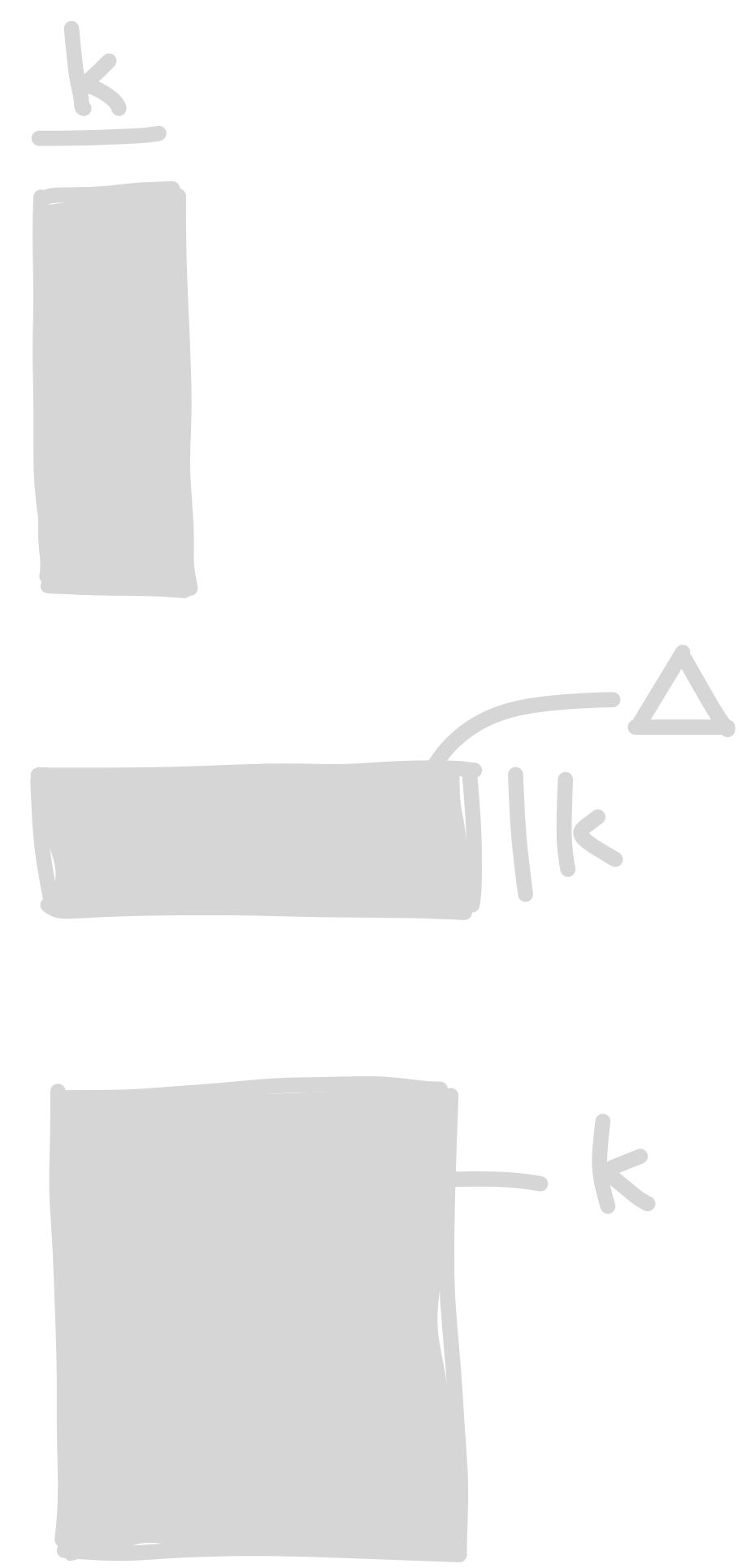
Integer Programming meets FPT

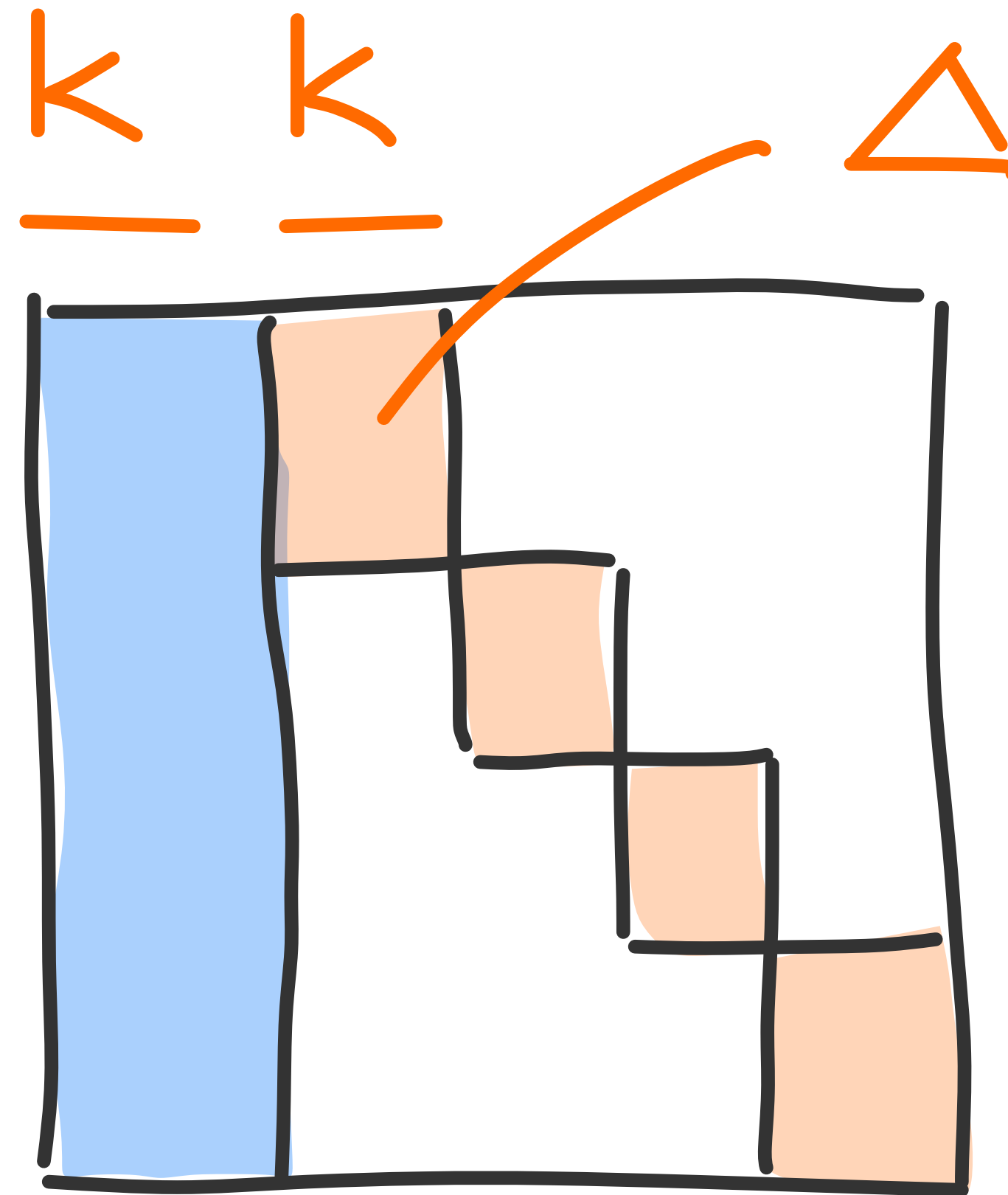
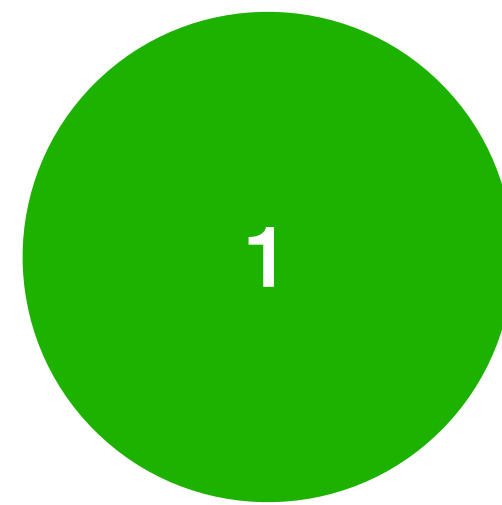


Integer Programming meets FPT



Integer Programming meets FPT - Agenda





2-stage stochastic IPs

2-stage stochastic IPs

The diagram illustrates a 2-stage stochastic linear programming problem. It consists of three main components: a coefficient matrix, a decision variable vector, and a right-hand side vector.

- Matrix:** A large rectangle is divided into two parts. The left part is a blue column containing blocks labeled A_1 , A_2 , \vdots , and A_n . The right part is an orange staircase-shaped block containing blocks labeled D_1 , D_2 , \vdots , and D_n .
- Dot Product:** A small black dot \cdot is placed between the matrix and the vector x .
- Decision Variable Vector:** A yellow vertical rectangle containing the labels x_1 , \vdots , and x_n .
- Inequality:** A less-than-or-equal-to symbol \leq is placed between the vector x and the vector b .
- Right-Hand Side Vector:** A green vertical rectangle containing the labels b_1 , \vdots , and b_m .

The overall equation represented is:

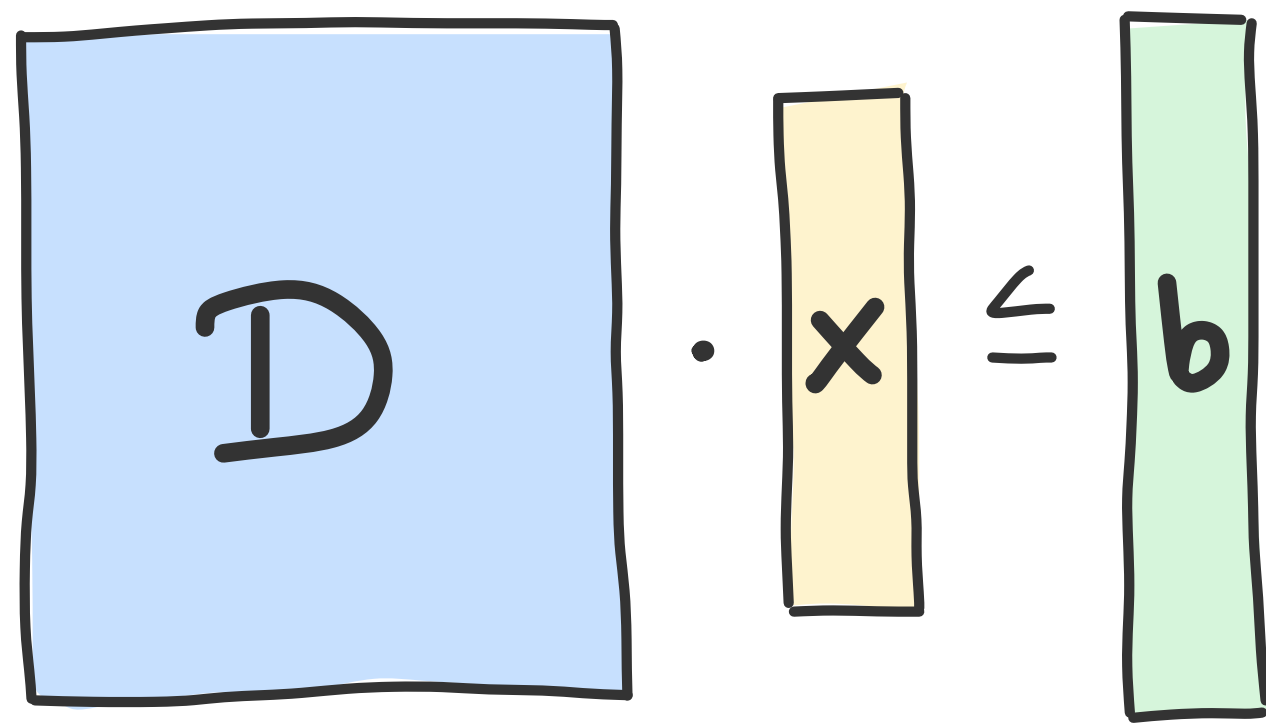
$$\begin{bmatrix} A_1 & D_1 \\ A_2 & \\ \vdots & \\ A_n & D_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

2-stage stochastic IPs

$$\begin{bmatrix} A_1 & D_1 \\ A_2 & & D_2 \\ \vdots & & & \ddots \\ A_n & & & & D_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

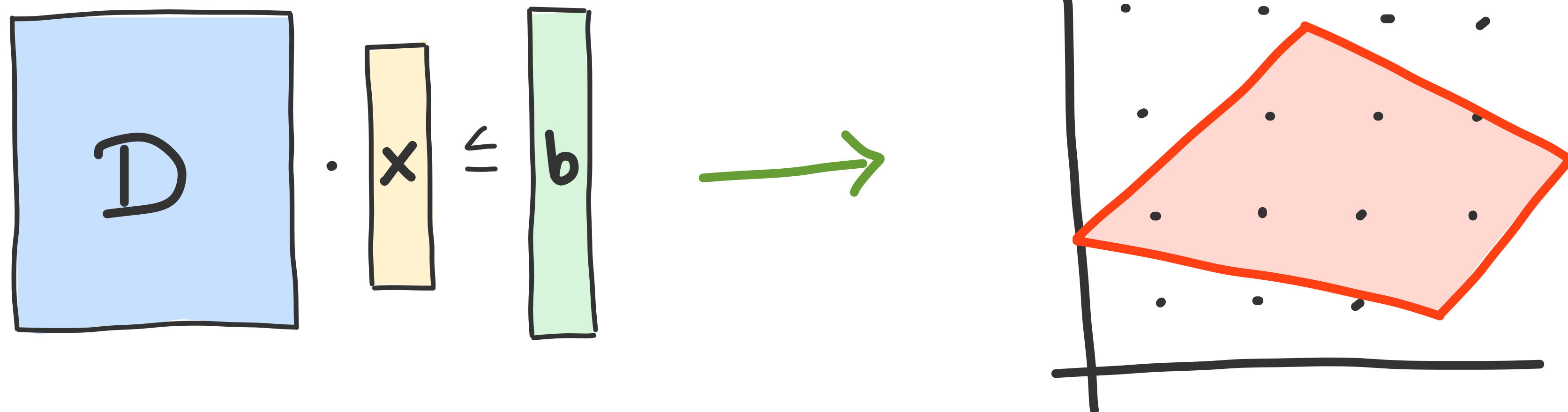
neural networks, worker scheduling, project planning, routing, facility location planning, ...

The Integer Hull

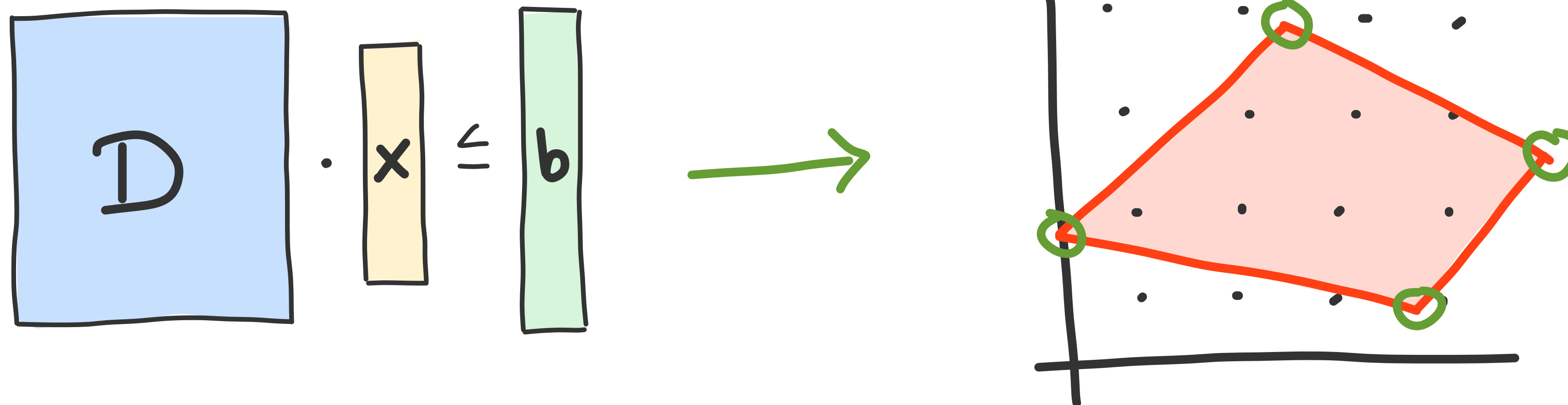


A diagram illustrating the equation $D \cdot x \leq b$. It features three hand-drawn elements: a light blue square on the left containing the letter 'D', a small yellow vertical rectangle in the middle containing the letter 'x', and a light green vertical rectangle on the right containing the letter 'b'. A dot is placed between the square and the yellow rectangle, and a less-than-or-equal-to symbol (\leq) is placed between the yellow rectangle and the green rectangle.

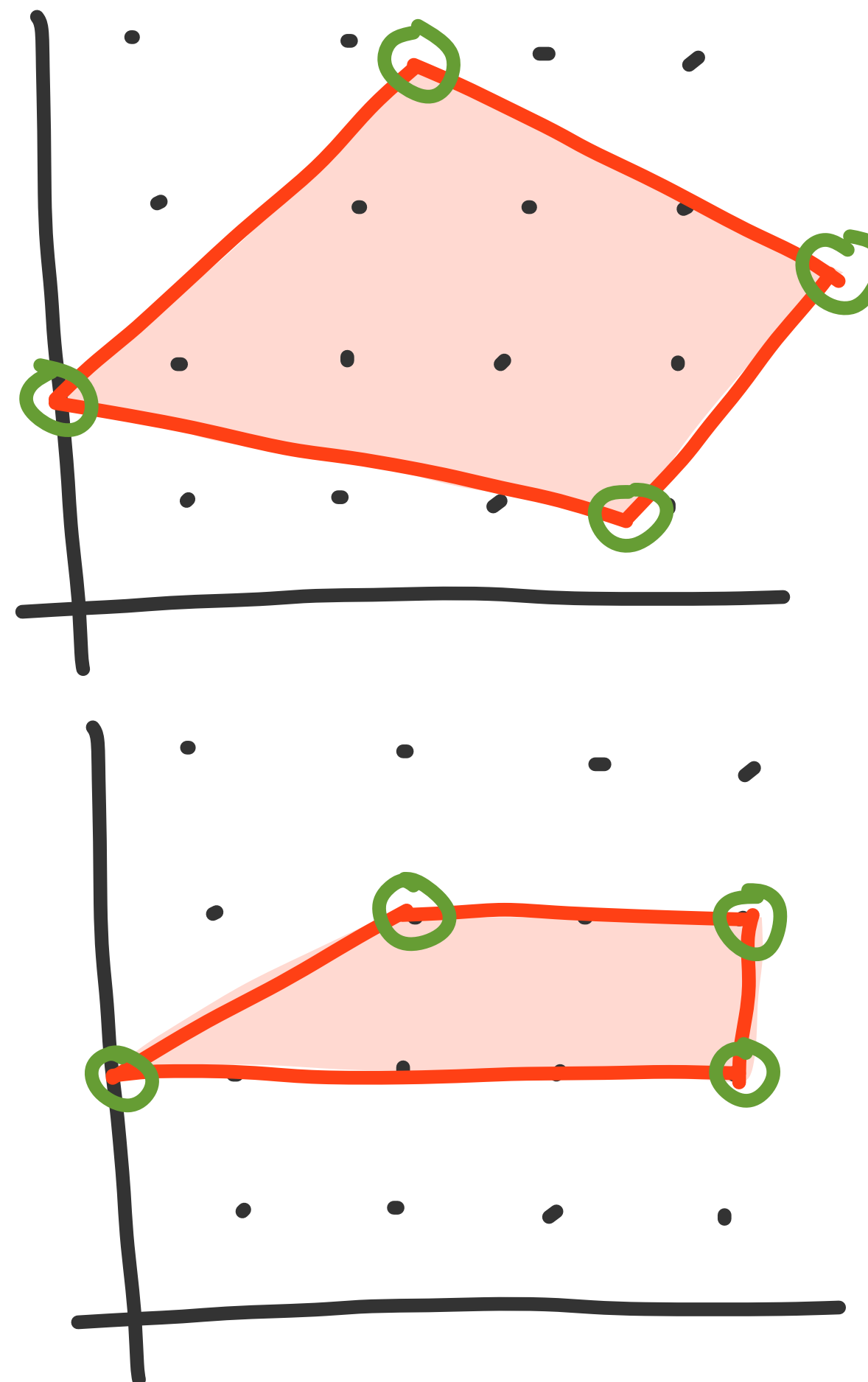
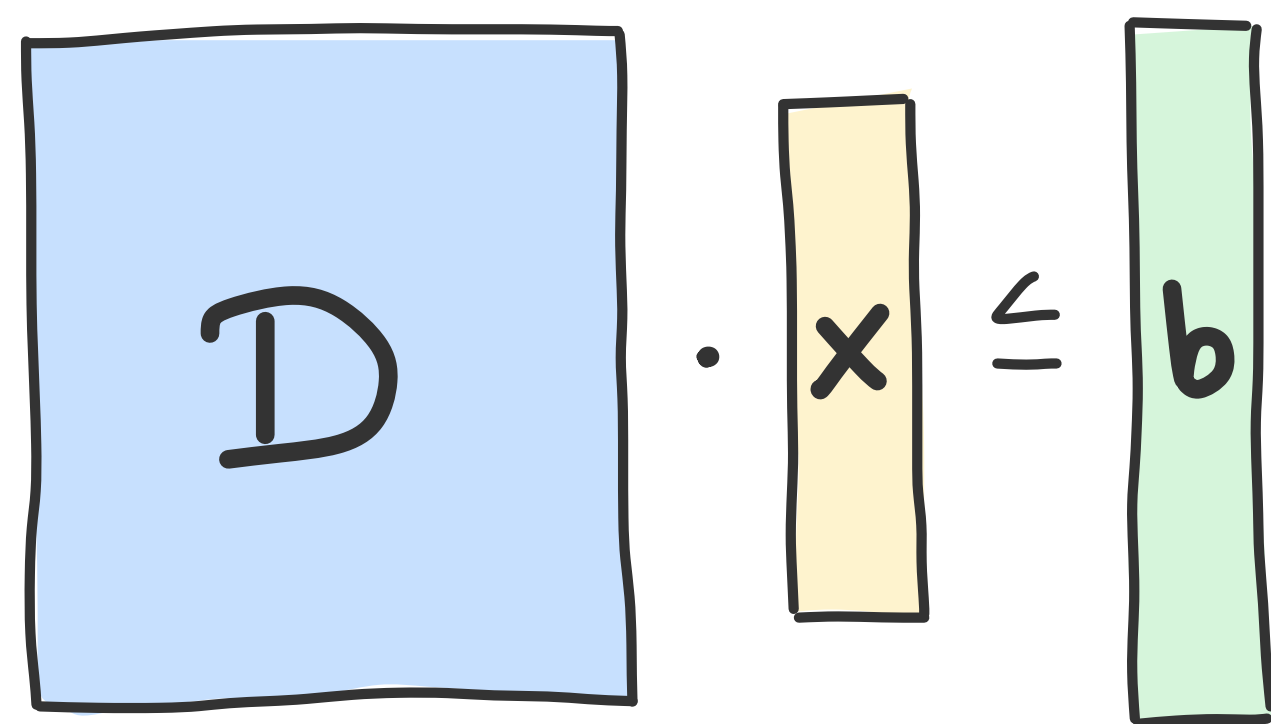
The Integer Hull



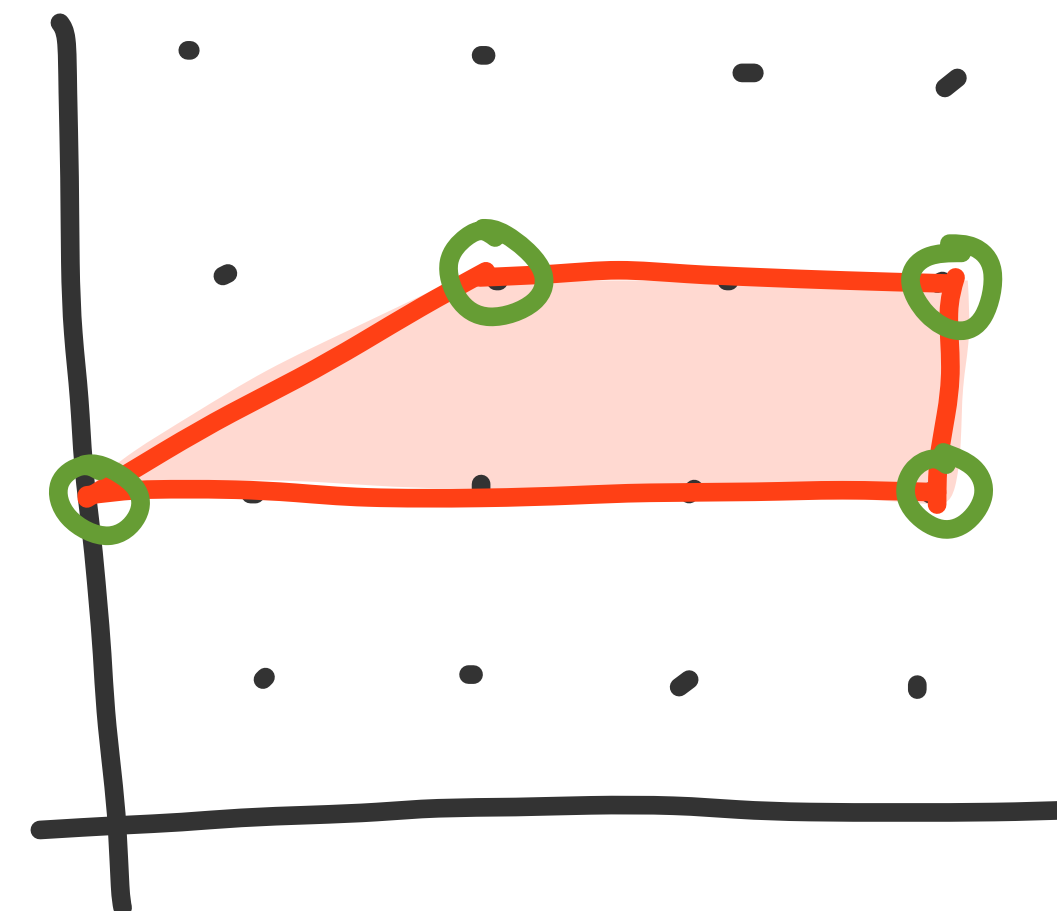
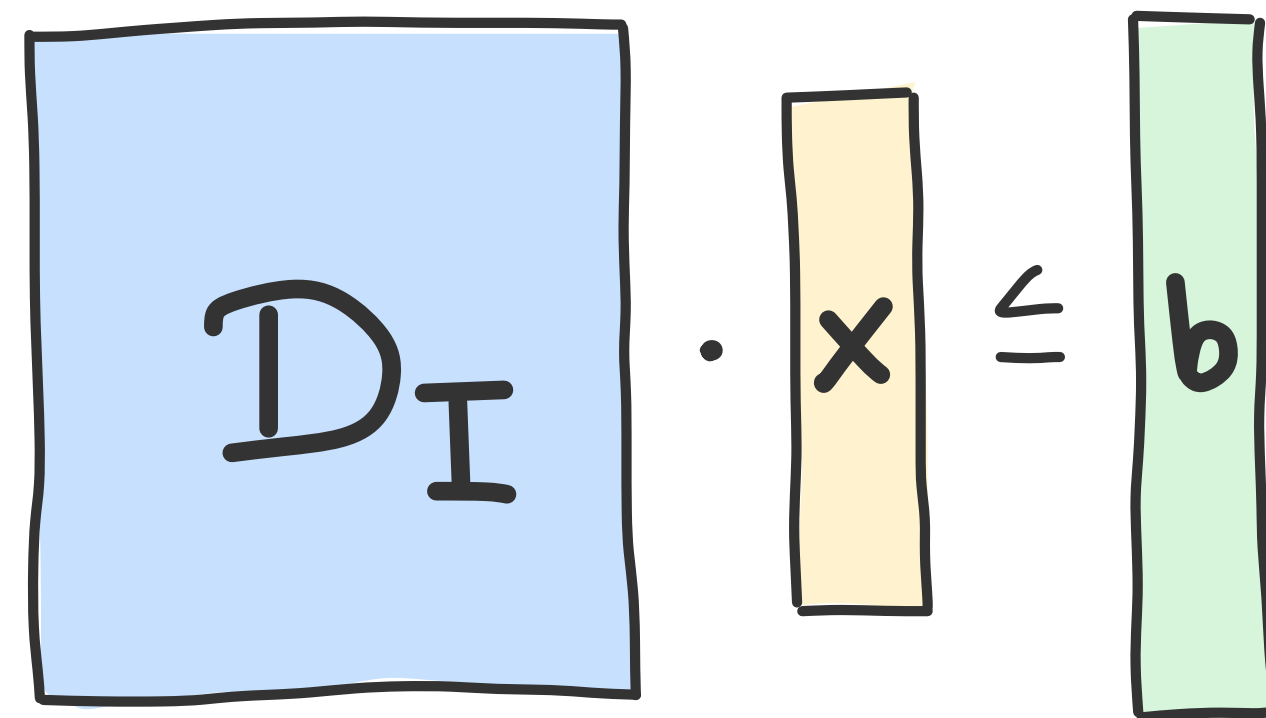
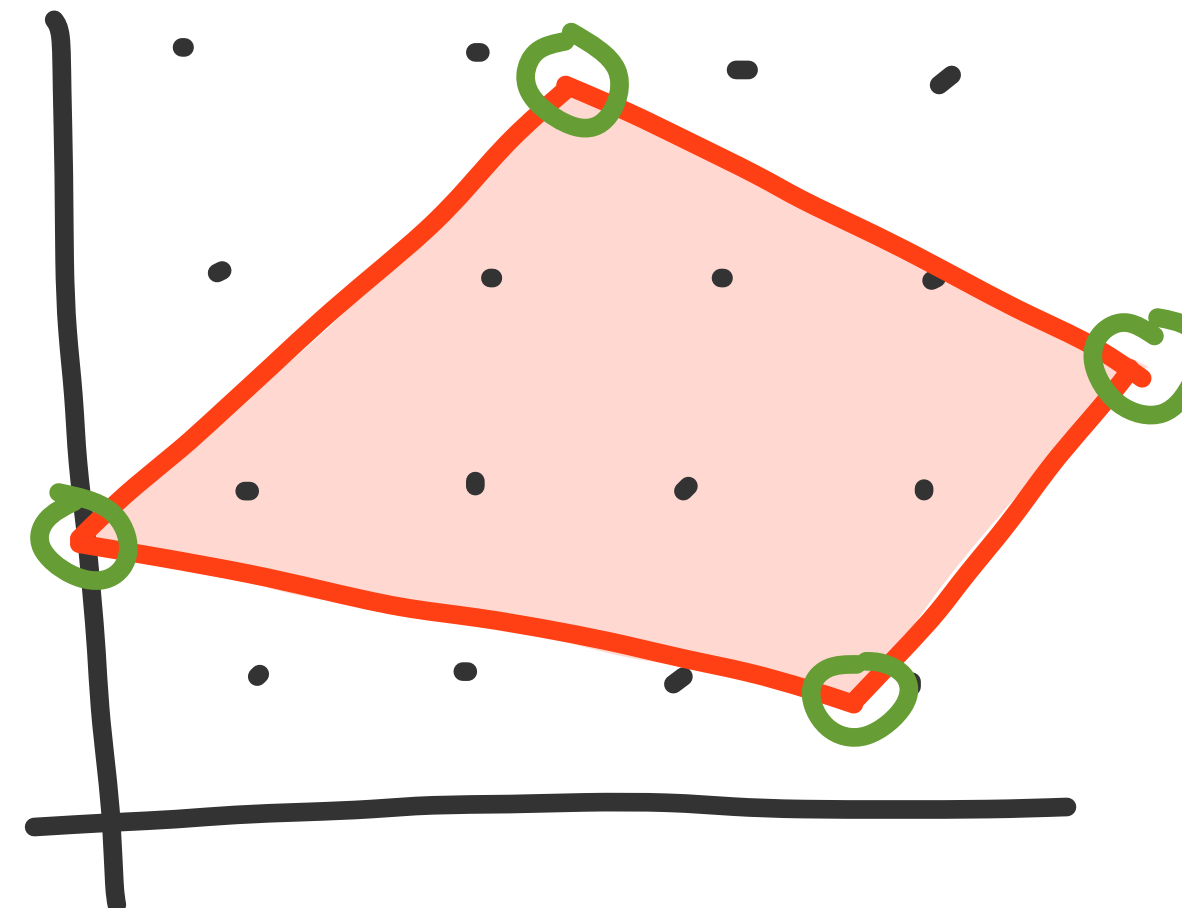
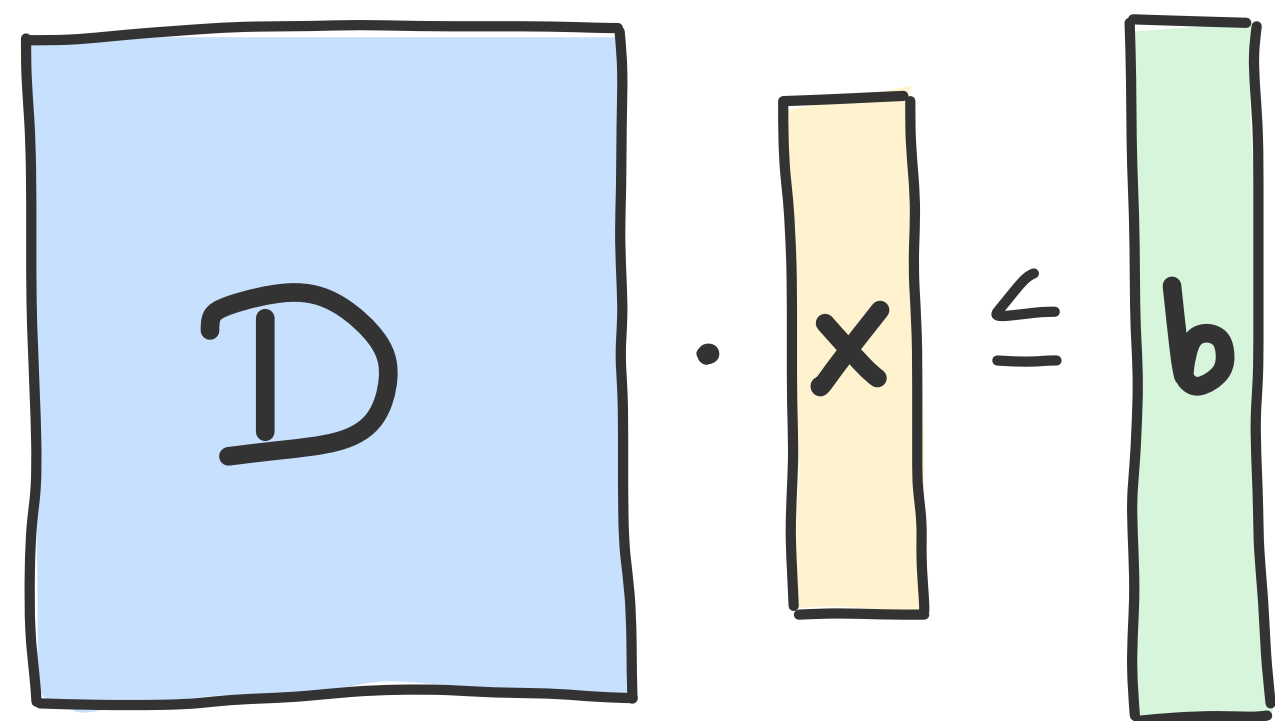
The Integer Hull



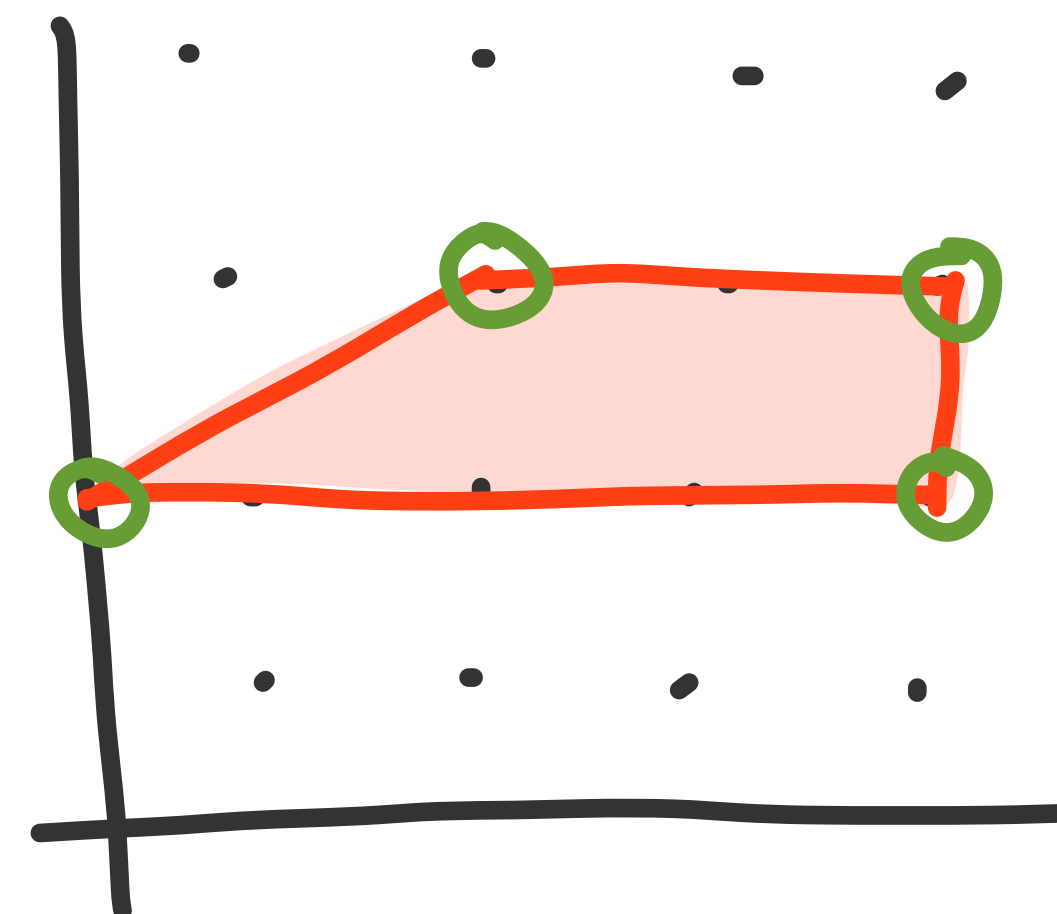
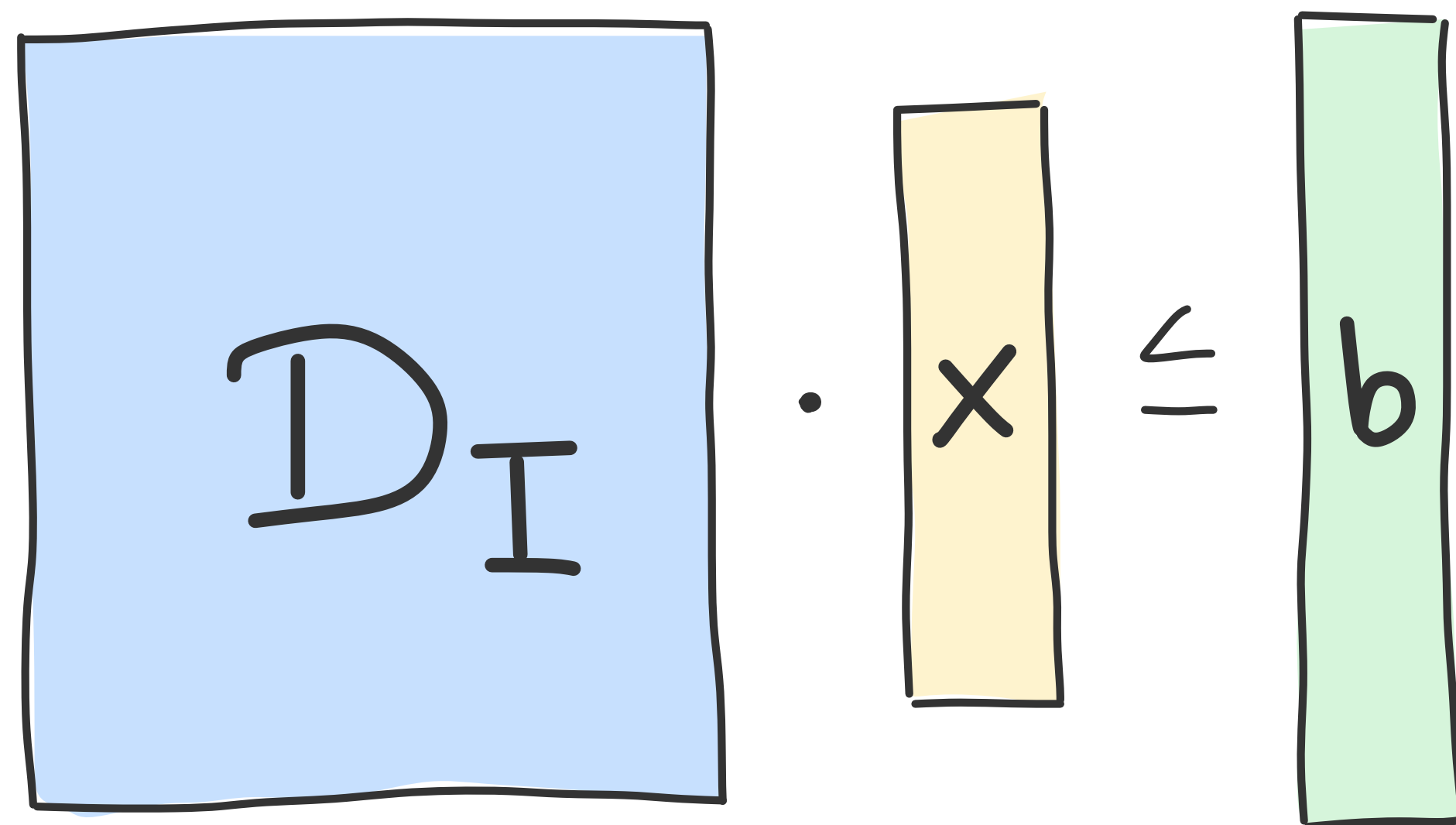
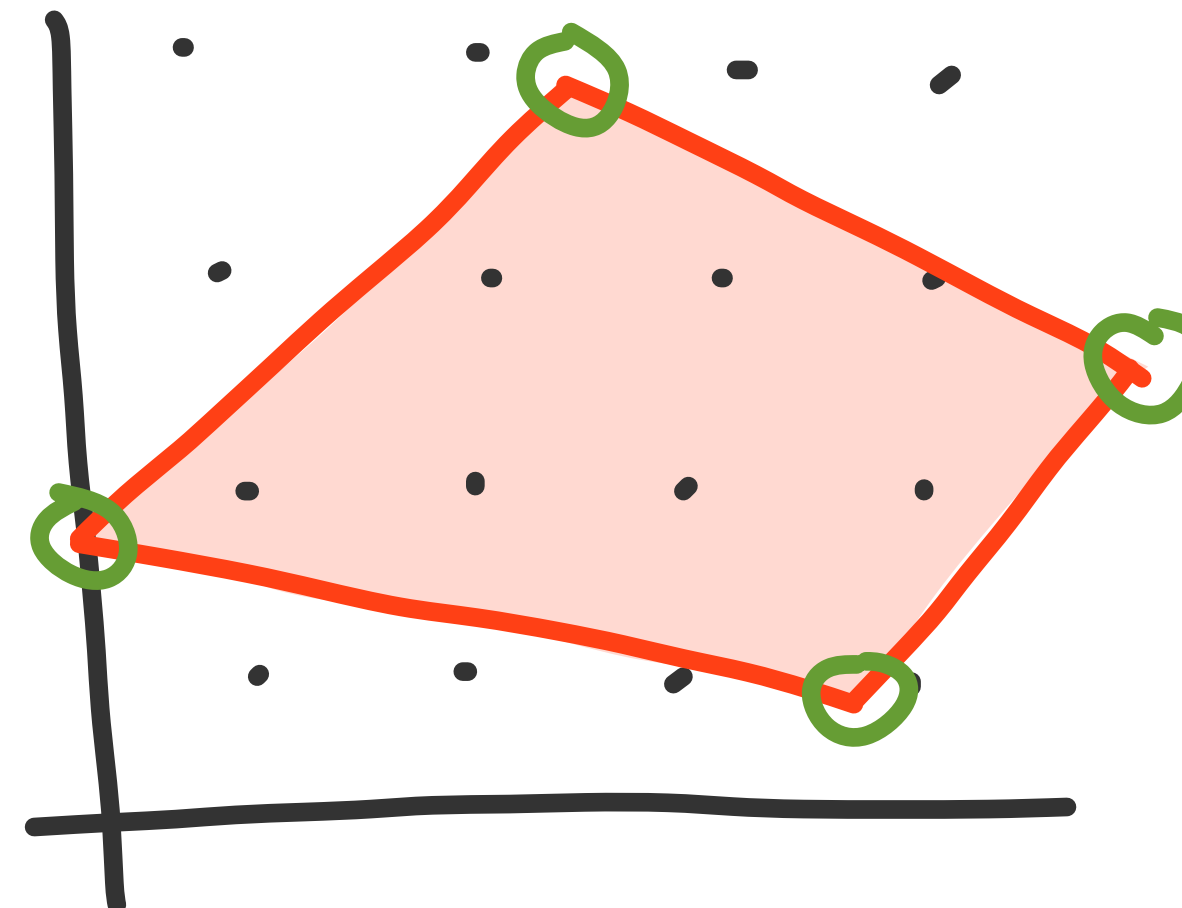
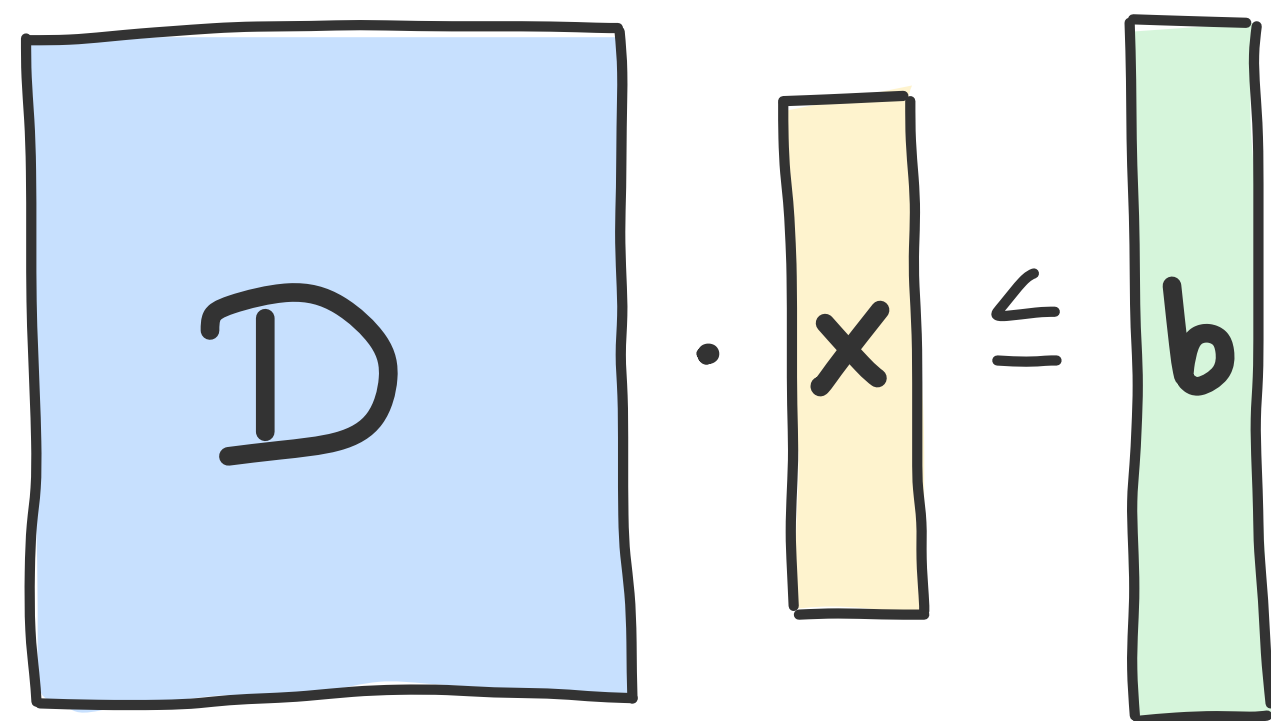
The Integer Hull



The Integer Hull



The Integer Hull



2-stage stochastic IPs

Theorem 1. ^{*} Let $D \in \mathbb{Z}^{m \times k}$ with non-repeating rows and $\|A\|_\infty \leq \Delta$. There exists a $R \in \mathbb{N}$ depending on k and Δ such that, for each $r \in \{0, \dots, R-1\}^m$, there exist $B_r \in \mathbb{Z}^{m' \times k}$, $C_r \in \mathbb{Z}^{m' \times m}$ and $f_r \in \mathbb{Z}^{m'}$ such that the following holds:

For each $b \in \mathbb{Z}^m$ with $b - r \in R \cdot \mathbb{Z}^m$, one has

$$P(b)_I = \{x \in \mathbb{R}^k : B_r x \leq f_r + C_r b\}.$$

integer hull

^{*}

“A parameterized linear formulation of the integer hull” by F. Eisenbrand and T. Rothvoss, 2025

2-stage stochastic IPs

Theorem 1. ^{*} Let $D \in \mathbb{Z}^{m \times k}$ with non-repeating rows and $\|A\|_\infty \leq \Delta$. There exists a $R \in \mathbb{N}$ depending on k and Δ such that, for each $r \in \{0, \dots, R-1\}^m$, there exist $B_r \in \mathbb{Z}^{m' \times k}$, $C_r \in \mathbb{Z}^{m' \times m}$ and $f_r \in \mathbb{Z}^{m'}$ such that the following holds:

For each $b \in \mathbb{Z}^m$ with $b - r \in R \cdot \mathbb{Z}^m$, one has

$$P(b)_I = \{x \in \mathbb{R}^k : B_r x \leq f_r + C_r b\}.$$

$$D \cdot x \leq b \xrightarrow{\text{for fixed } r} B_r \cdot x \leq f_r + C_r \cdot b$$

^{*}

“A parameterized linear formulation of the integer hull” by F. Eisenbrand and T. Rothvoss, 2025

2-stage stochastic IPs

Theorem 1. ^{*} Let $D \in \mathbb{Z}^{m \times k}$ with non-repeating rows and $\|A\|_\infty \leq \Delta$. There exists a $R \in \mathbb{N}$ depending on k and Δ such that, for each $r \in \{0, \dots, R-1\}^m$, there exist $B_r \in \mathbb{Z}^{m' \times k}$, $C_r \in \mathbb{Z}^{m' \times m}$ and $f_r \in \mathbb{Z}^{m'}$ such that the following holds:

For each $b \in \mathbb{Z}^m$ with $b - r \in R \cdot \mathbb{Z}^m$, one has


$$P(b)_I = \{x \in \mathbb{R}^k : B_r x \leq f_r + C_r b\}.$$

There is an fpt-sized description of the integer hull of $Dx \leq b$ (for fixed r)

^{*}

“A parameterized linear formulation of the integer hull” by F. Eisenbrand and T. Rothvoss, 2025

2-stage stochastic IPs

Theorem 1.  Let $A \in \mathbb{Z}^{m \times k}$ with non-repeating rows and $\|A\|_\infty \leq \Delta$. There exists a $R \in \mathbb{N}$ depending on k and Δ such that, for each $r \in \{0, \dots, R-1\}^m$, there exist $B_r \in \mathbb{Z}^{m' \times k}$, $C_r \in \mathbb{Z}^{m' \times m}$ and $f_r \in \mathbb{Z}^{m'}$ such that the following holds:

For each $b \in \mathbb{Z}^m$ with $b - r \in R \cdot \mathbb{Z}^m$, one has

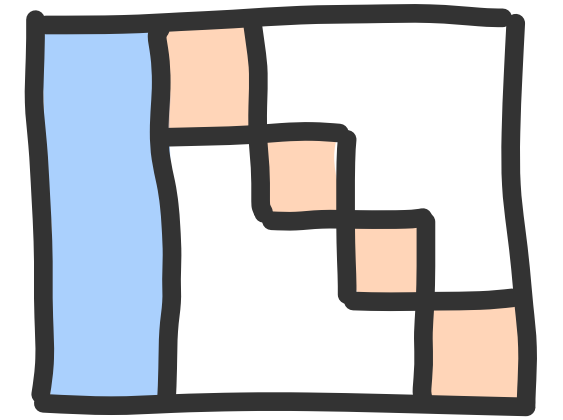
$$P(b)_I = \{x \in \mathbb{R}^k : B_r x \leq f_r + C_r b\}.$$

Furthermore, the number $R \in \mathbb{N}$, the matrices B_r and C_r , as well as the vector f_r can be computed in time depending on Δ and k only.



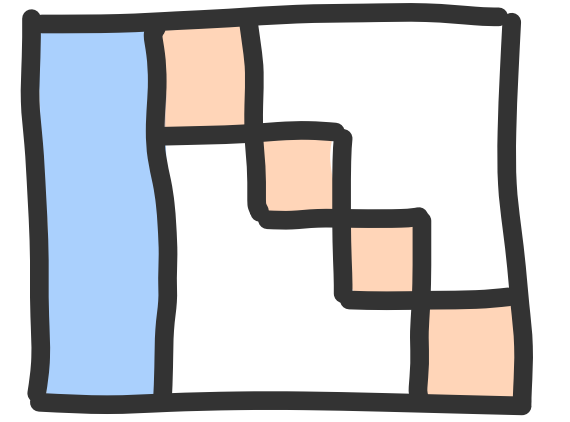
“A parameterized linear formulation of the integer hull” by F. Eisenbrand and T. Rothvoss, 2025

The Algorithm



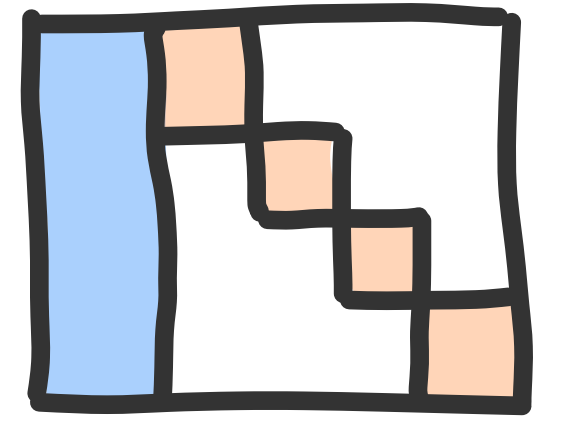
1. Compute R and guess remainder r for all D_i

The Algorithm



1. Compute R and guess remainder r for all D_i
2. Compute integer hull description of each D_i and replace them

The Algorithm



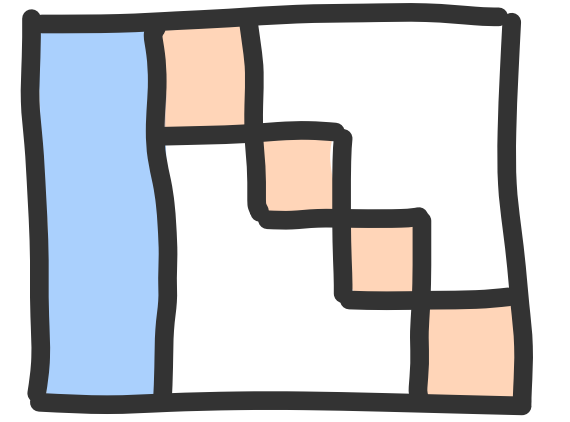
1. Compute R and guess remainder r for all D_i

2. Compute integer hull description of each D_i and replace them

$$\begin{bmatrix} A_1 & & & \\ & D_1 & & \\ & & D_2 & \\ & & & \ddots \\ & & & & D_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

$$\begin{bmatrix} A_1 & & & \\ & D_1^r & & \\ & & D_2^r & \\ & & & \ddots \\ & & & & D_n^r \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^r \\ \vdots \\ b_n^r \end{bmatrix}$$

The Algorithm



1. Compute R and guess remainder r for all D_i

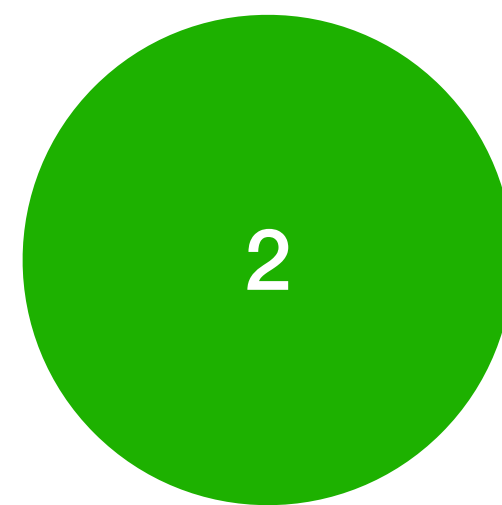
2. Compute integer hull description of each D_i and replace them

3. Solve the MIP

$$\begin{bmatrix} A_1 & D_1 & & \\ & A_2 & D_2 & \\ & & \ddots & \\ & & & A_n & D_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

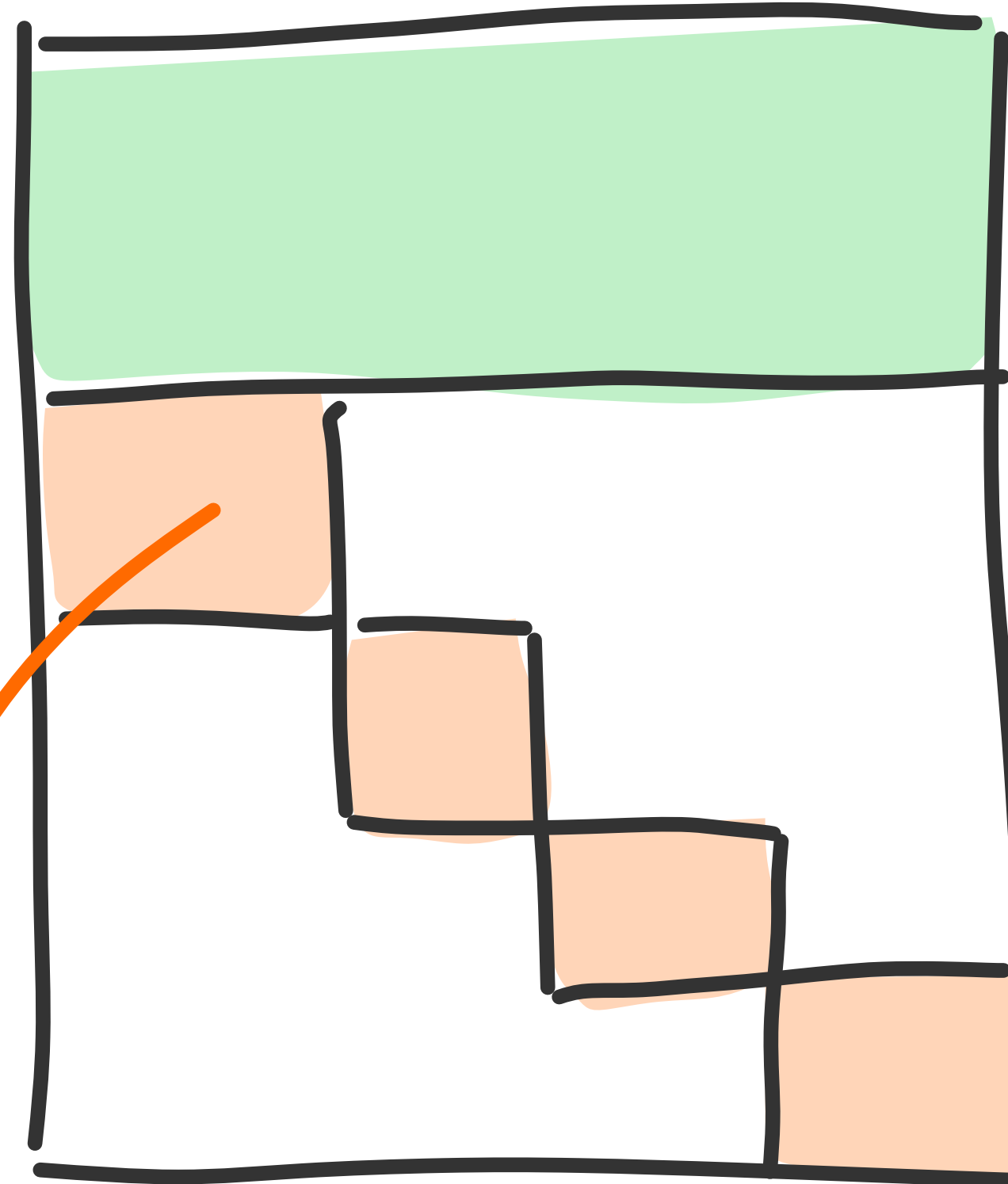
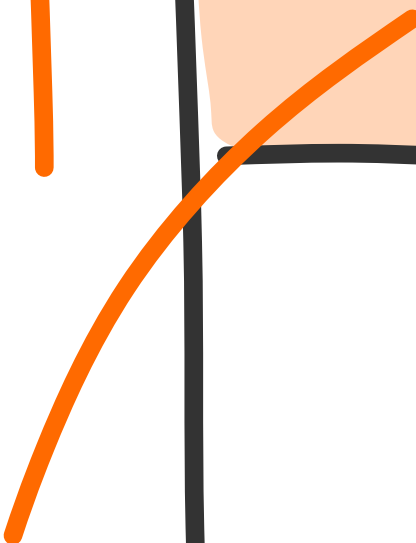
k

$$\begin{bmatrix} A_1 & D_1^r & & \\ & A_2 & D_2^r & \\ & & \ddots & \\ & & & A_n & D_n^r \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1^r \\ \vdots \\ b_n^r \end{bmatrix}$$



k

k



n-fold IPs

n-fold IPs

$$\begin{bmatrix} C_1 & C_2 & \dots & C_n \\ D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

n-fold IPs

$$\begin{bmatrix} C_1 & C_2 & \dots & C_n \\ D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

scheduling, knapsack-like problems, string problems,
social choice, ...

n-fold IPs

$$\begin{bmatrix} C_1 & C_2 & \dots & C_n \\ D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

n-fold IPs

$$\begin{bmatrix} C_1 & C_2 & \dots & C_n \\ D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

If dimension is small, we are done

n-fold IPs

A hand-drawn diagram illustrating an n-fold IP (Integer Programming) problem. It shows a large square matrix with a block structure. The top row of the matrix is divided into four green blocks labeled C_1 , C_2 , \dots , and C_n . Below this, the matrix is partitioned into diagonal blocks along the main diagonal, labeled D_1 , D_2 , \dots , and D_n . To the right of the matrix is a dot, followed by a vertical yellow column vector with elements x_1 , \vdots , and x_n . This is followed by an equals sign, and then a vertical green column vector with elements b_1 , \vdots , and b_m .

$$\begin{bmatrix} C_1 & C_2 & \dots & C_n \\ & D_1 & & \\ & & D_2 & \\ & & & \ddots \\ & & & & D_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

If dimension is small, we are done

If it is large, there are only few types of diagonal blocks

n-fold IPs

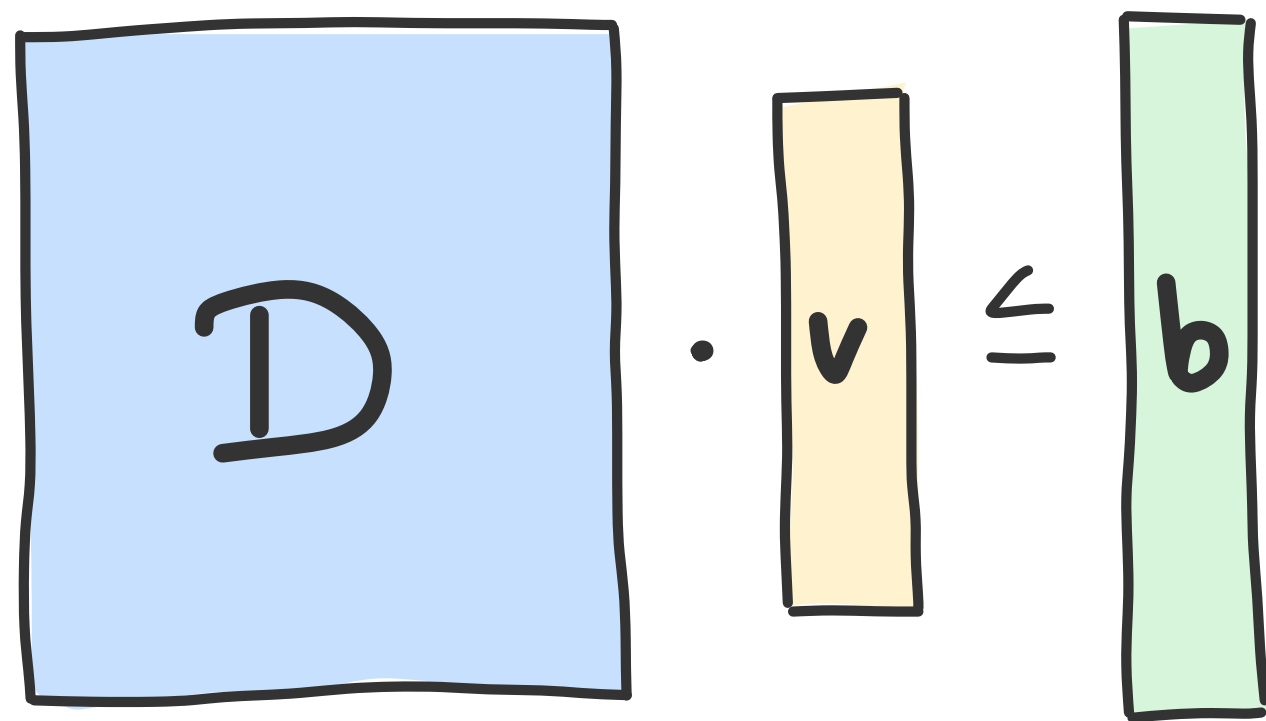
$\exists b', b''$ sign-compatible to b s.t. $b = b' + b''$ and $\forall v \in \mathbb{Z}_{\geq 0}^y$ with $D_i v = b$ there exist v', v'' with

1. $v = v' + v''$
2. $D_i v' = b'$
3. $D_i v'' = b''$

n-fold IPs

$\exists b', b''$ sign-compatible to b s.t. $b = b' + b''$ and $\forall v \in \mathbb{Z}_{\geq 0}^y$ with $D_i v = b$ there exist v', v'' with

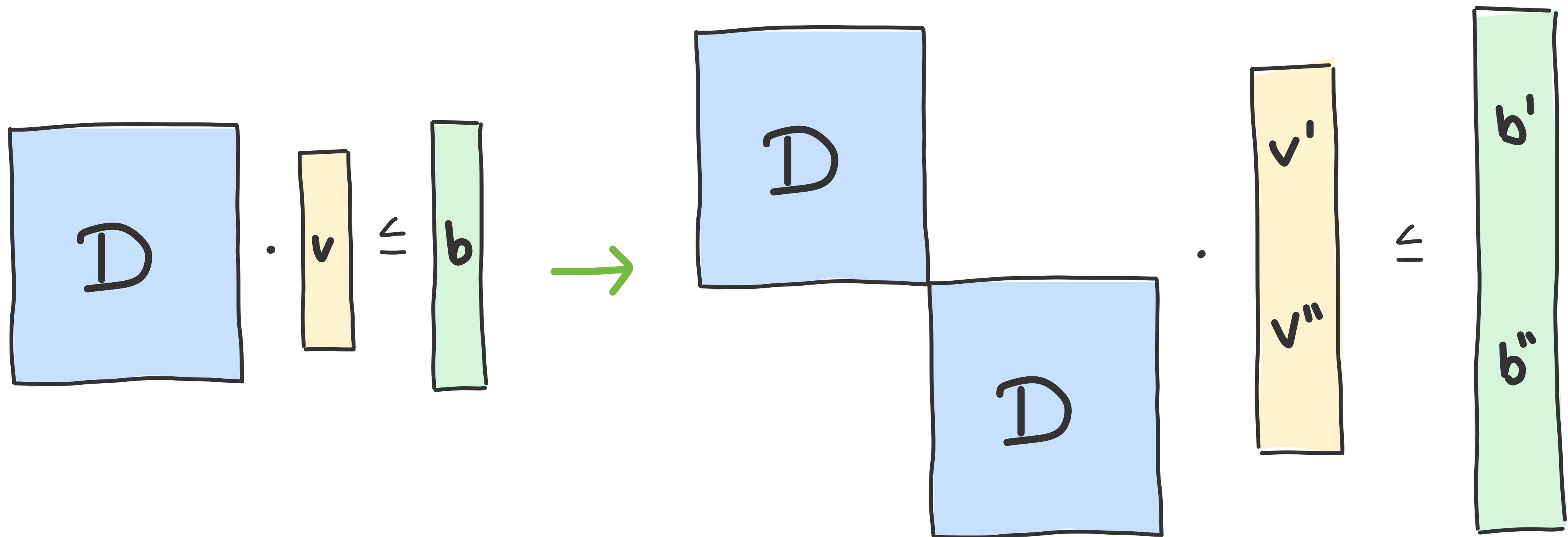
1. $v = v' + v''$
2. $D_i v' = b'$
3. $D_i v'' = b''$



n-fold IPs

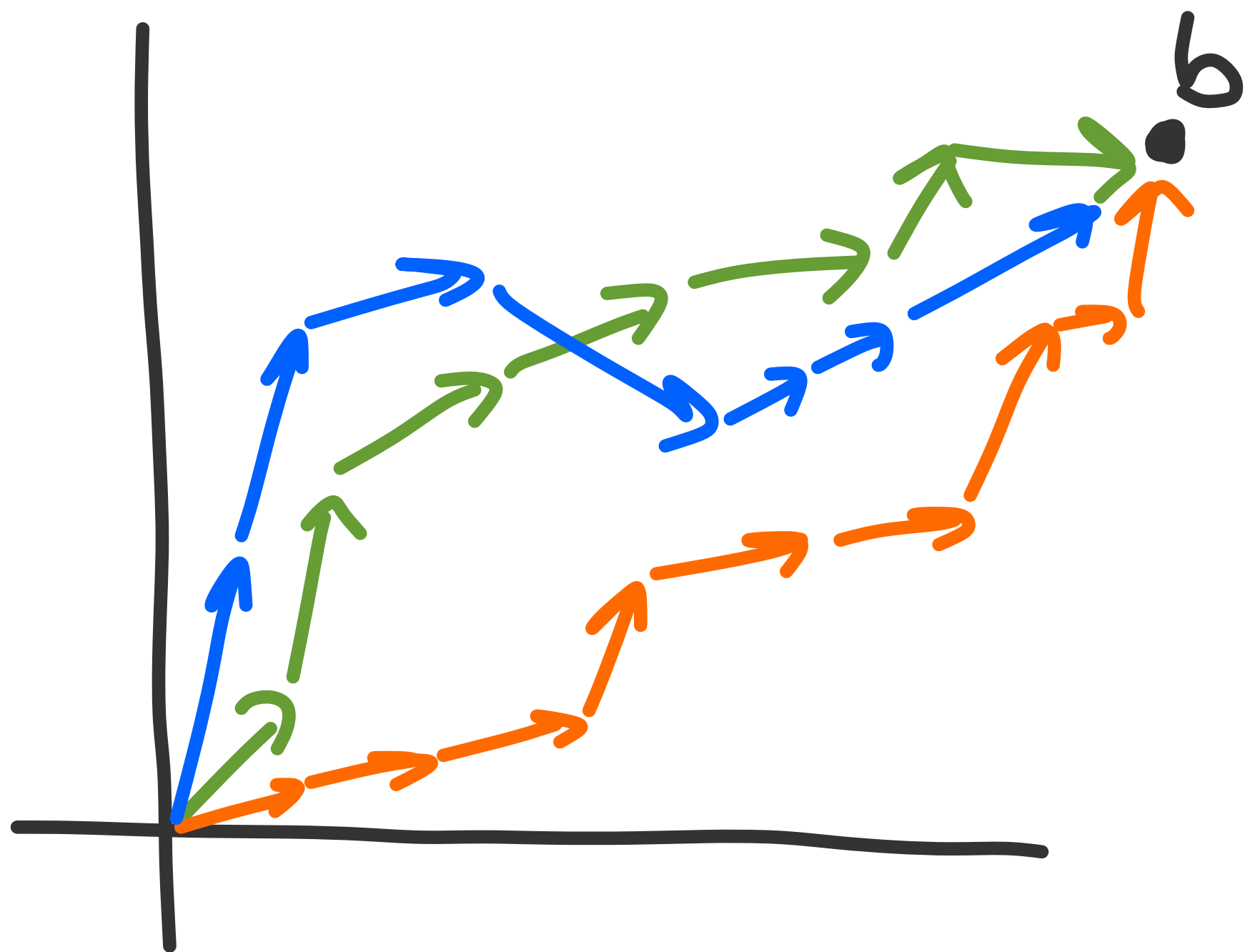
$\exists b', b''$ sign-compatible to b s.t. $b = b' + b''$ and $\forall v \in \mathbb{Z}_{\geq 0}^y$ with $D_i v = b$ there exist v', v'' with

1. $v = v' + v''$
2. $D_i v' = b'$
3. $D_i v'' = b''$



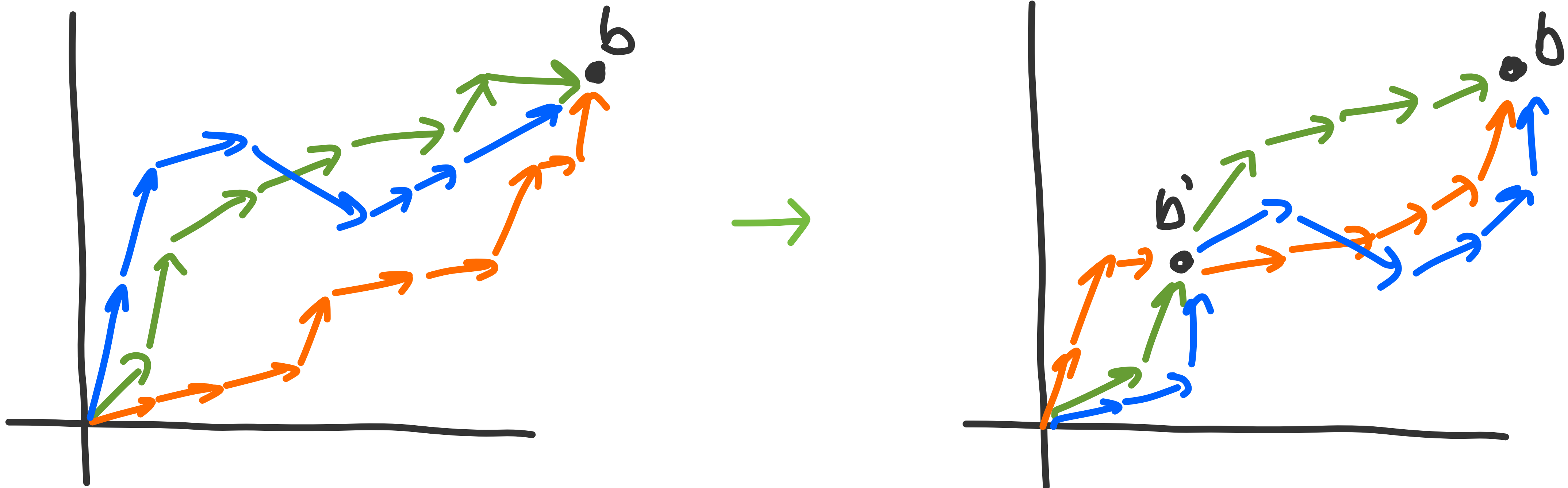
n-fold IPs - based on Klein Lemma

$\exists b', b''$ sign-compatible to b s.t. $b = b' + b''$ and $\forall v \in \mathbb{Z}_{\geq 0}^y$ with $D_i v = b$ there exist v', v'' with 1. $v = v' + v''$ 2. $D_i v' = b'$ 3. $D_i v'' = b''$



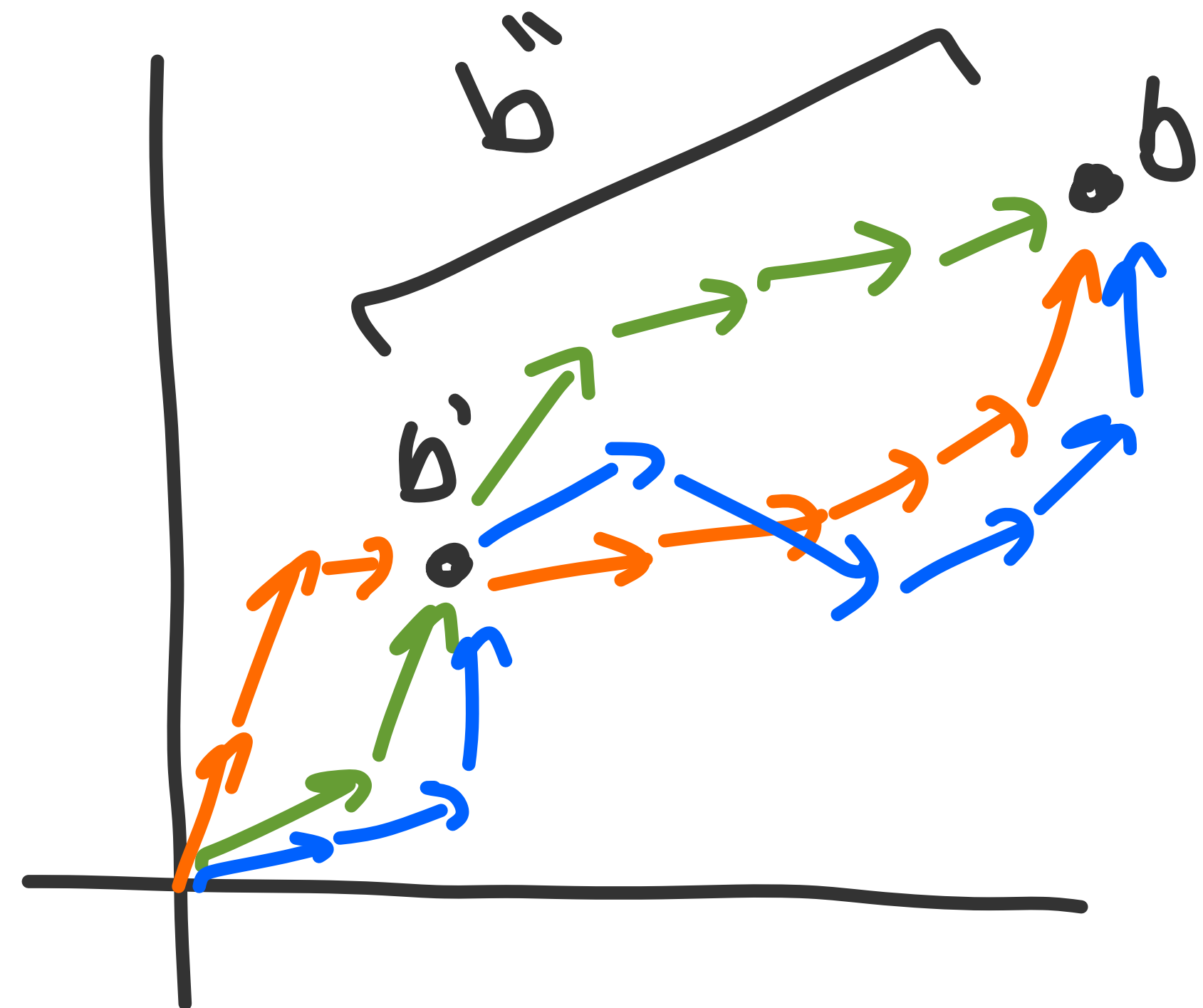
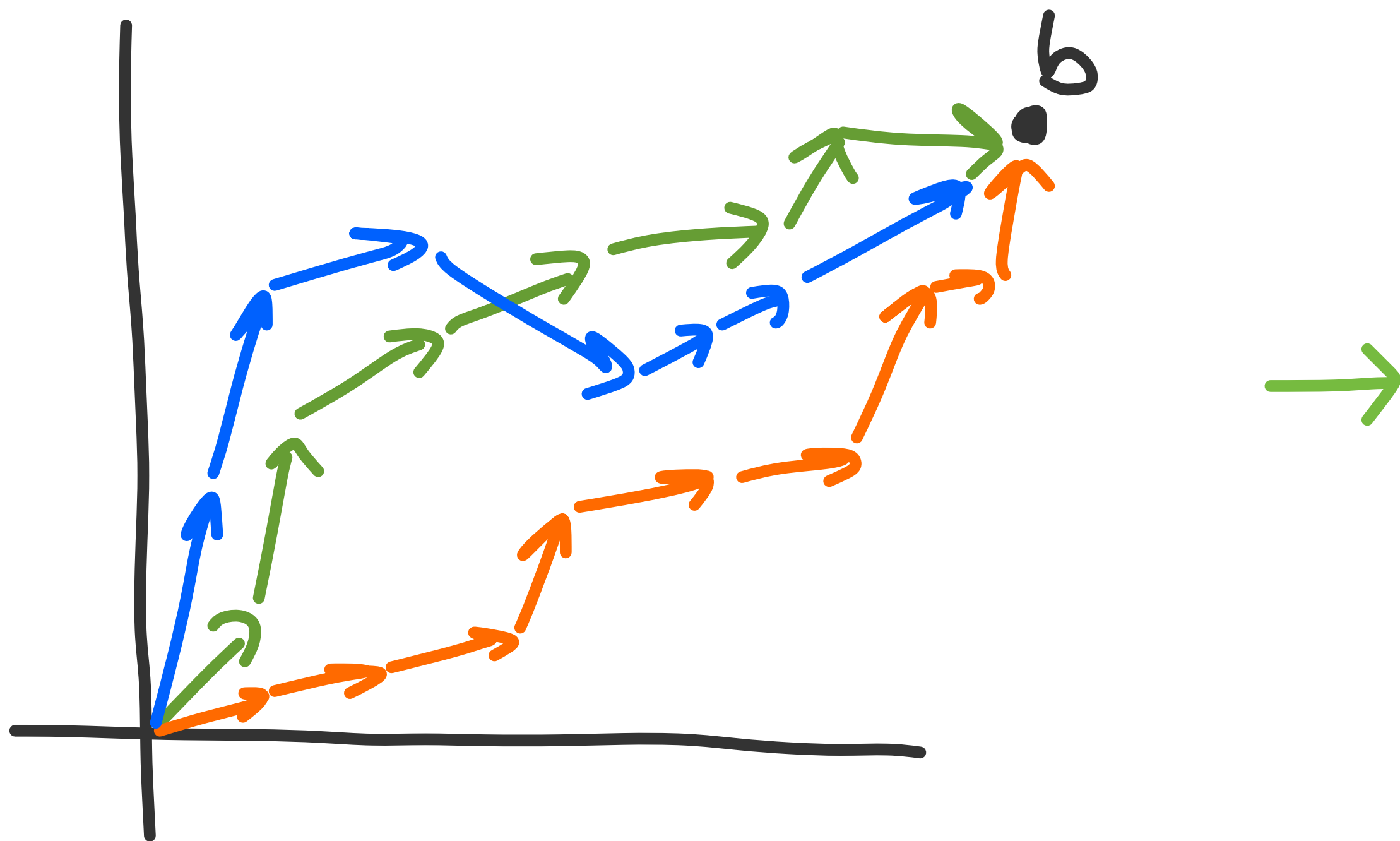
n-fold IPs - based on Klein Lemma

$\exists b', b''$ sign-compatible to b s.t. $b = b' + b''$ and $\forall v \in \mathbb{Z}_{\geq 0}^y$ with $D_i v = b$ there exist v', v'' with 1. $v = v' + v''$ 2. $D_i v' = b'$ 3. $D_i v'' = b''$

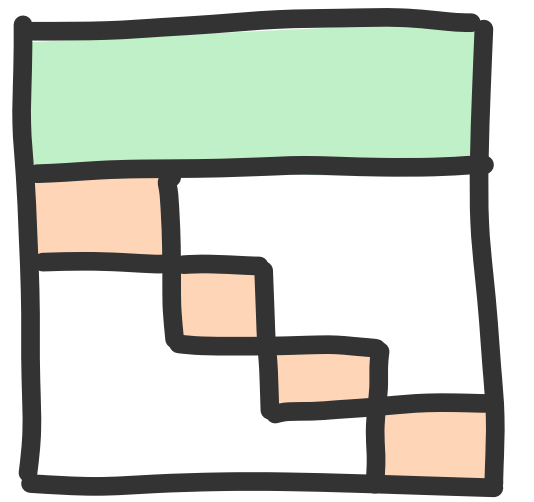


n-fold IPs - based on Klein Lemma

$\exists b', b''$ sign-compatible to b s.t. $b = b' + b''$ and $\forall v \in \mathbb{Z}_{\geq 0}^y$ with $D_i v = b$ there exist v', v'' with 1. $v = v' + v''$ 2. $D_i v' = b'$ 3. $D_i v'' = b''$

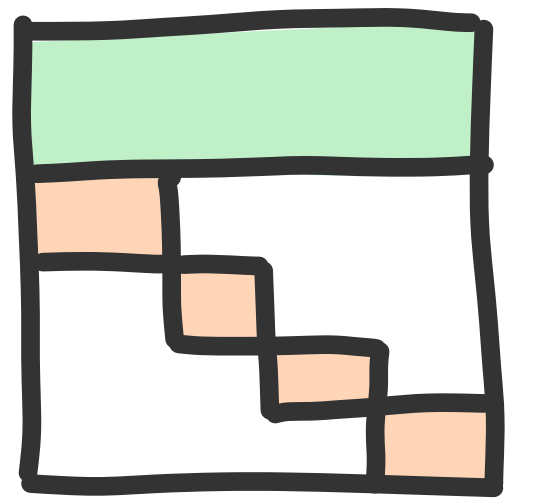


The Algorithm

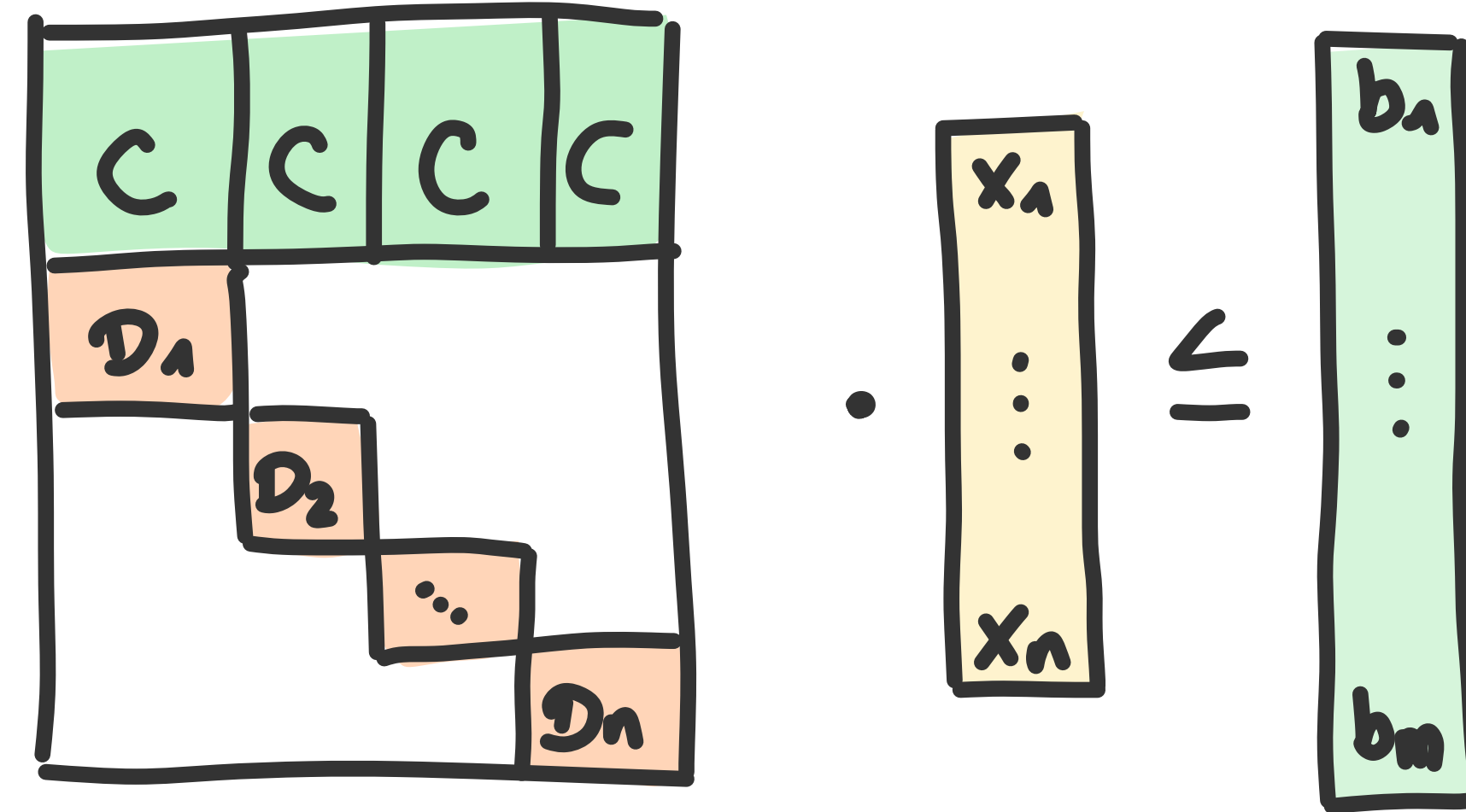


1. Breaking the bricks

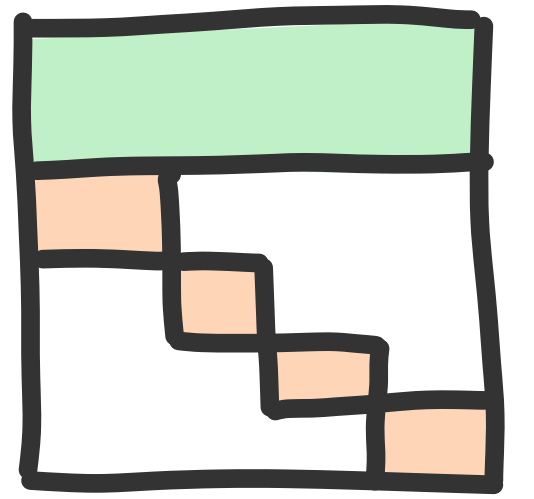
The Algorithm



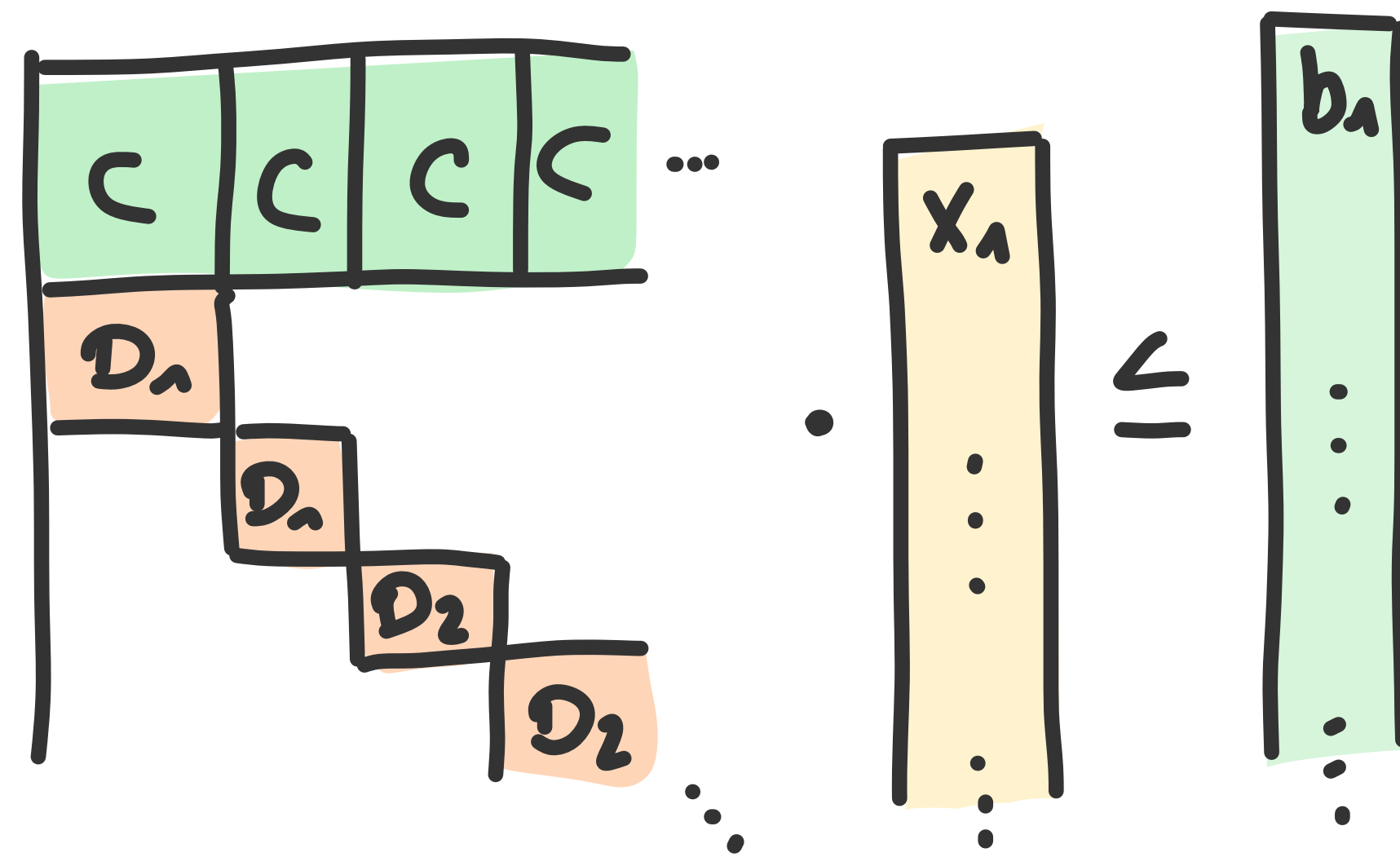
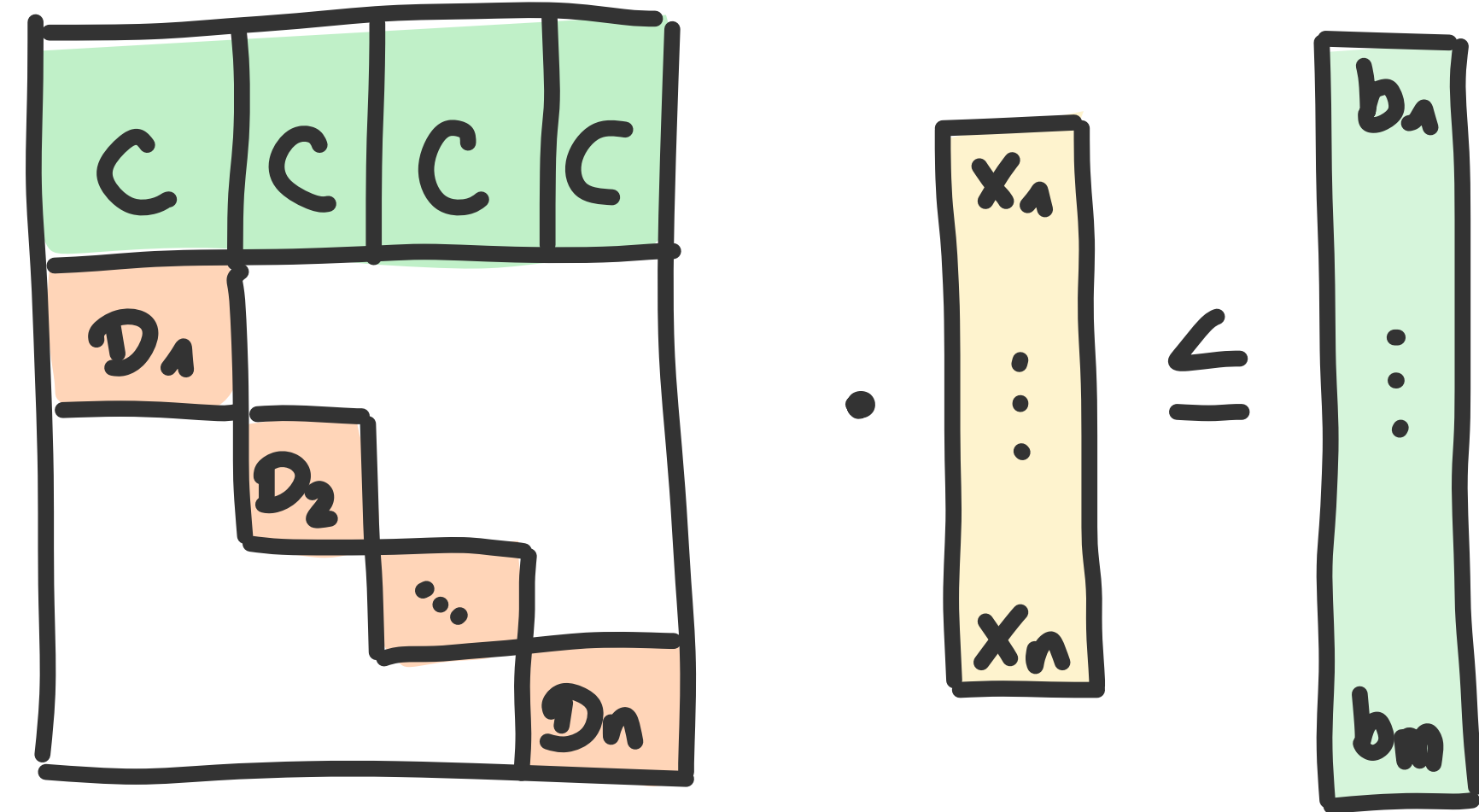
1. Breaking the bricks



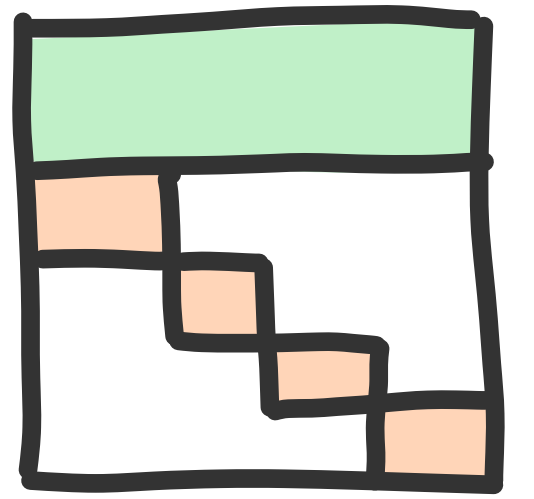
The Algorithm



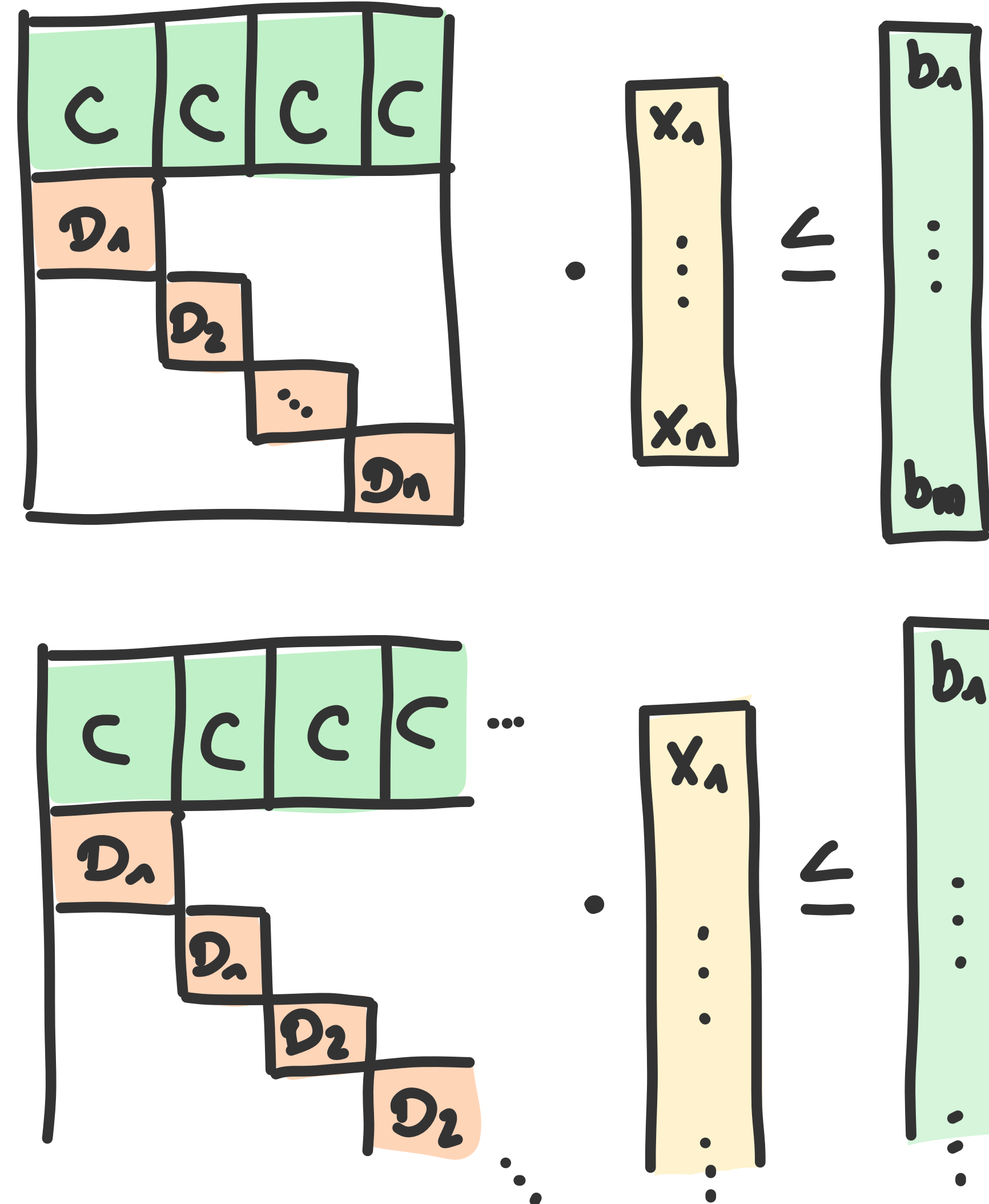
1. Breaking the bricks



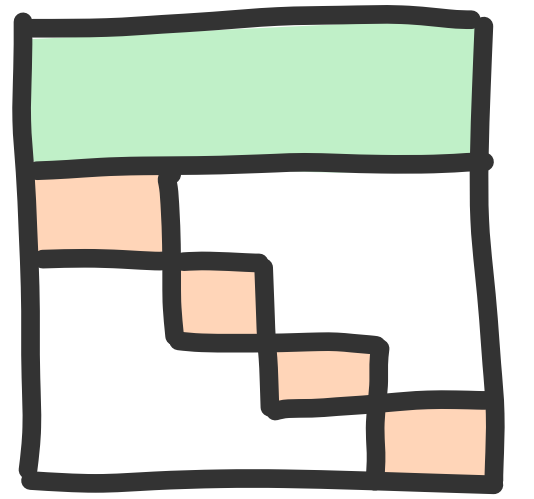
The Algorithm



1. Breaking the bricks

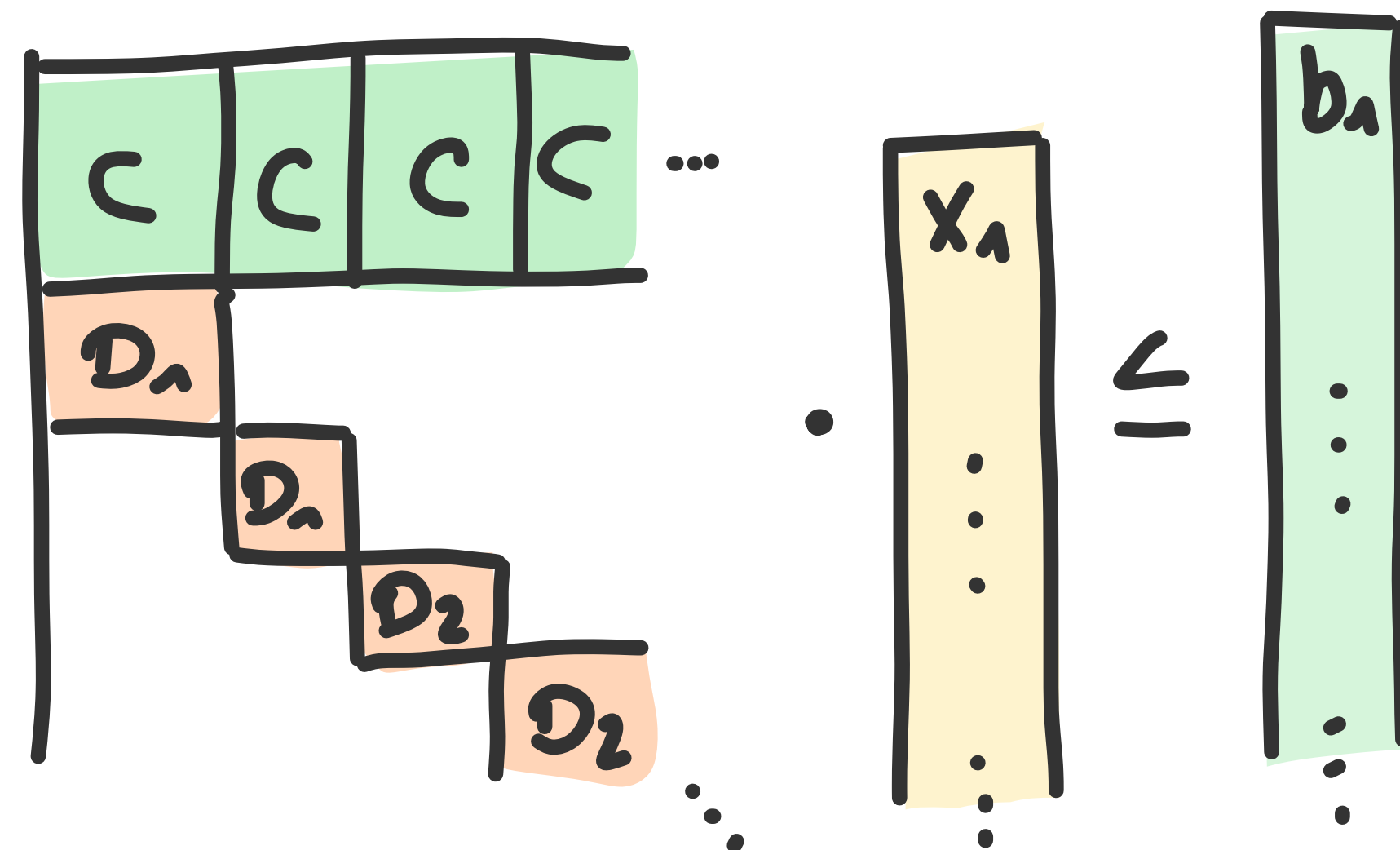
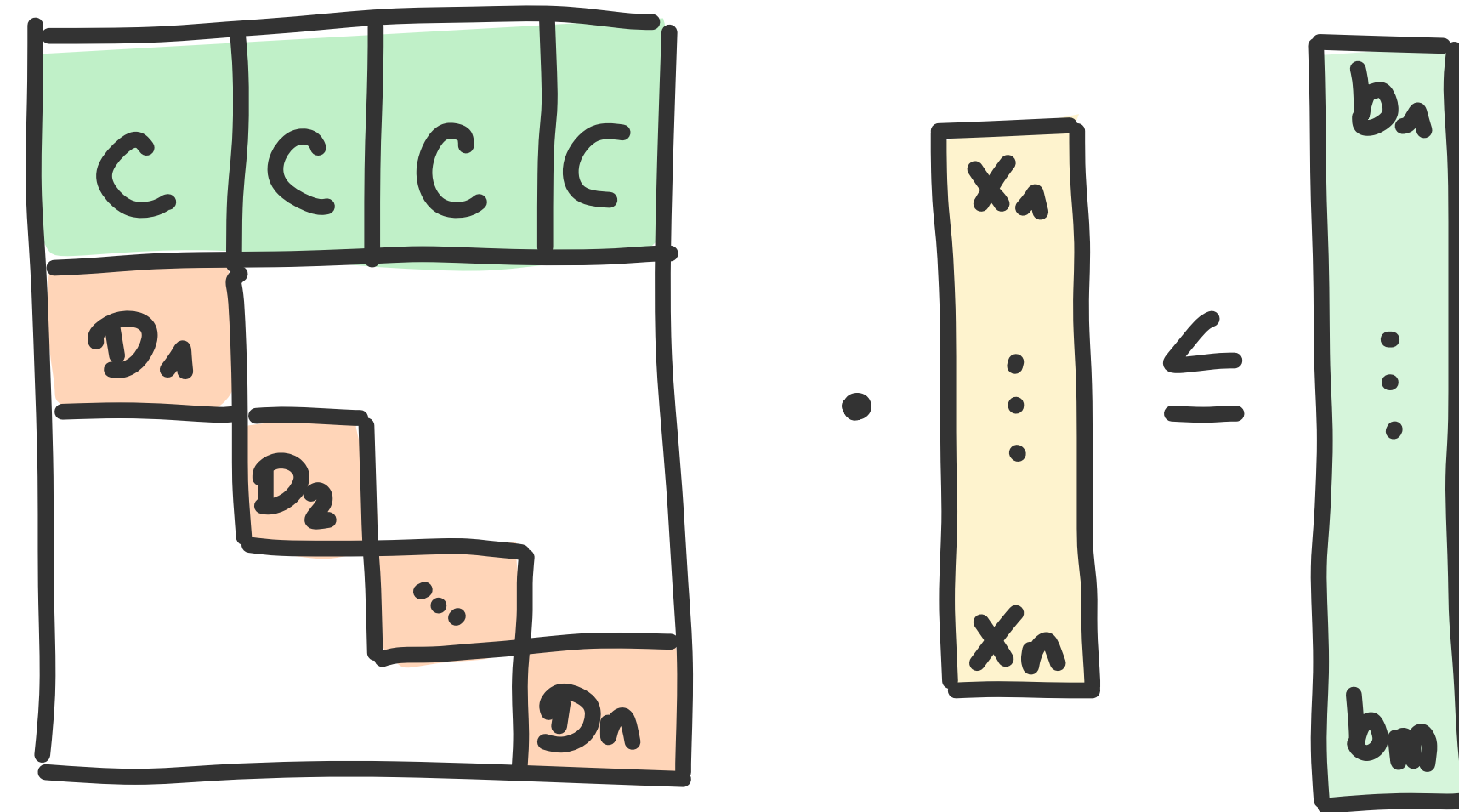


The Algorithm

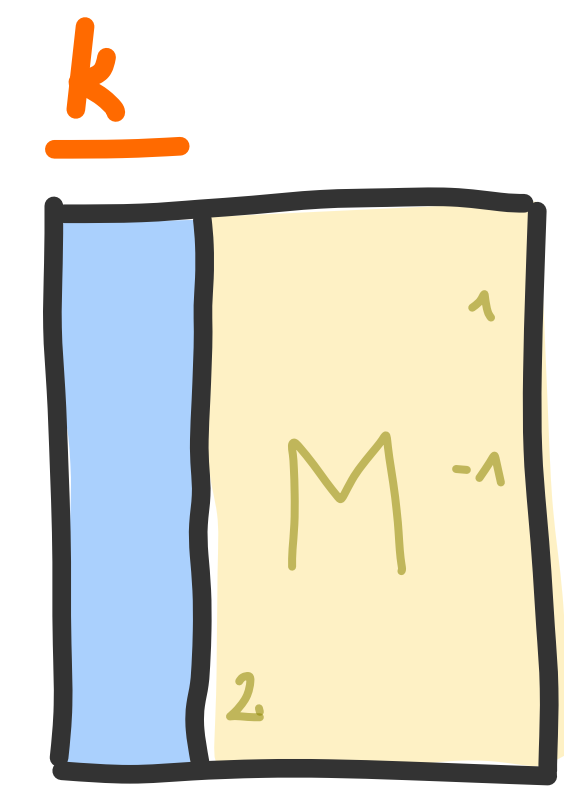
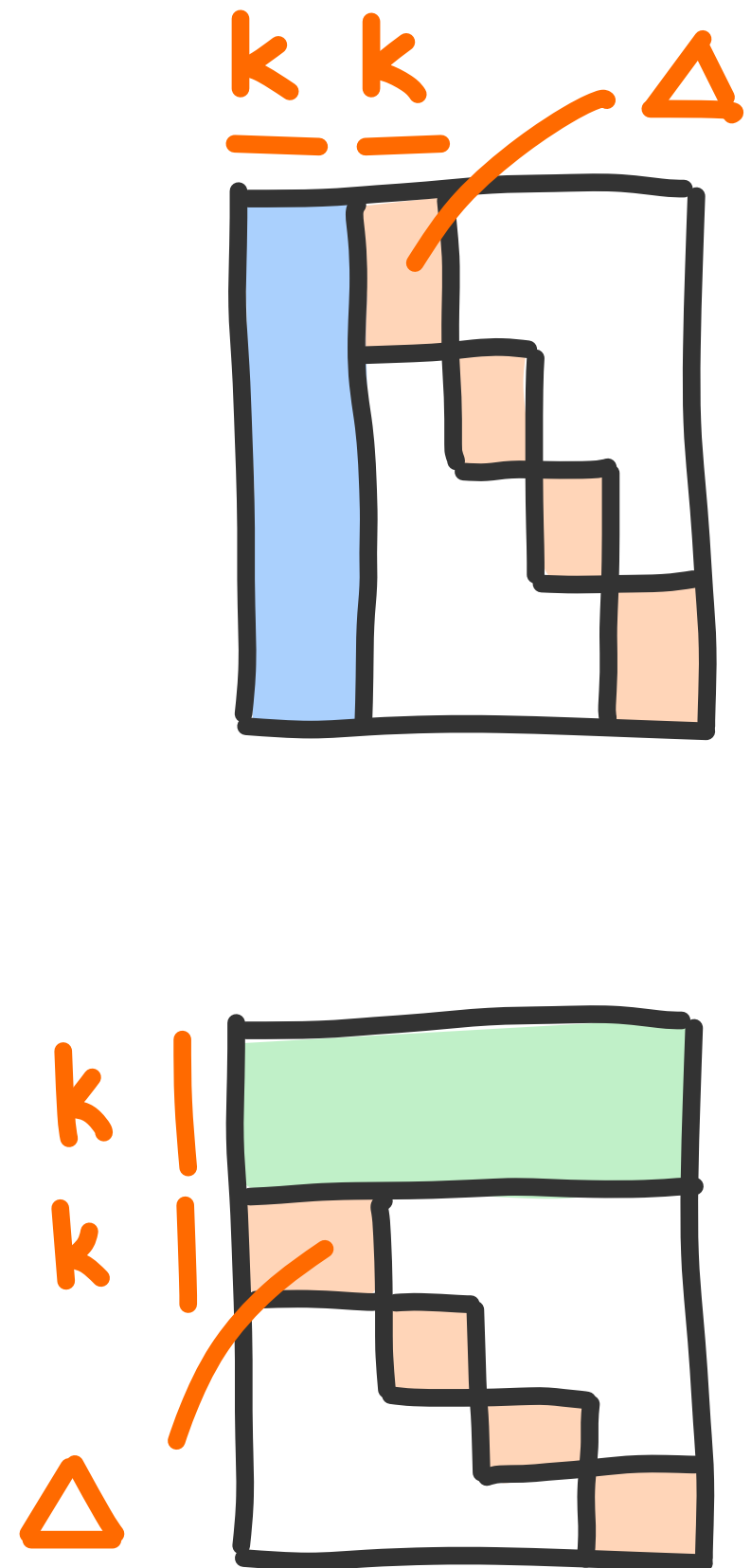
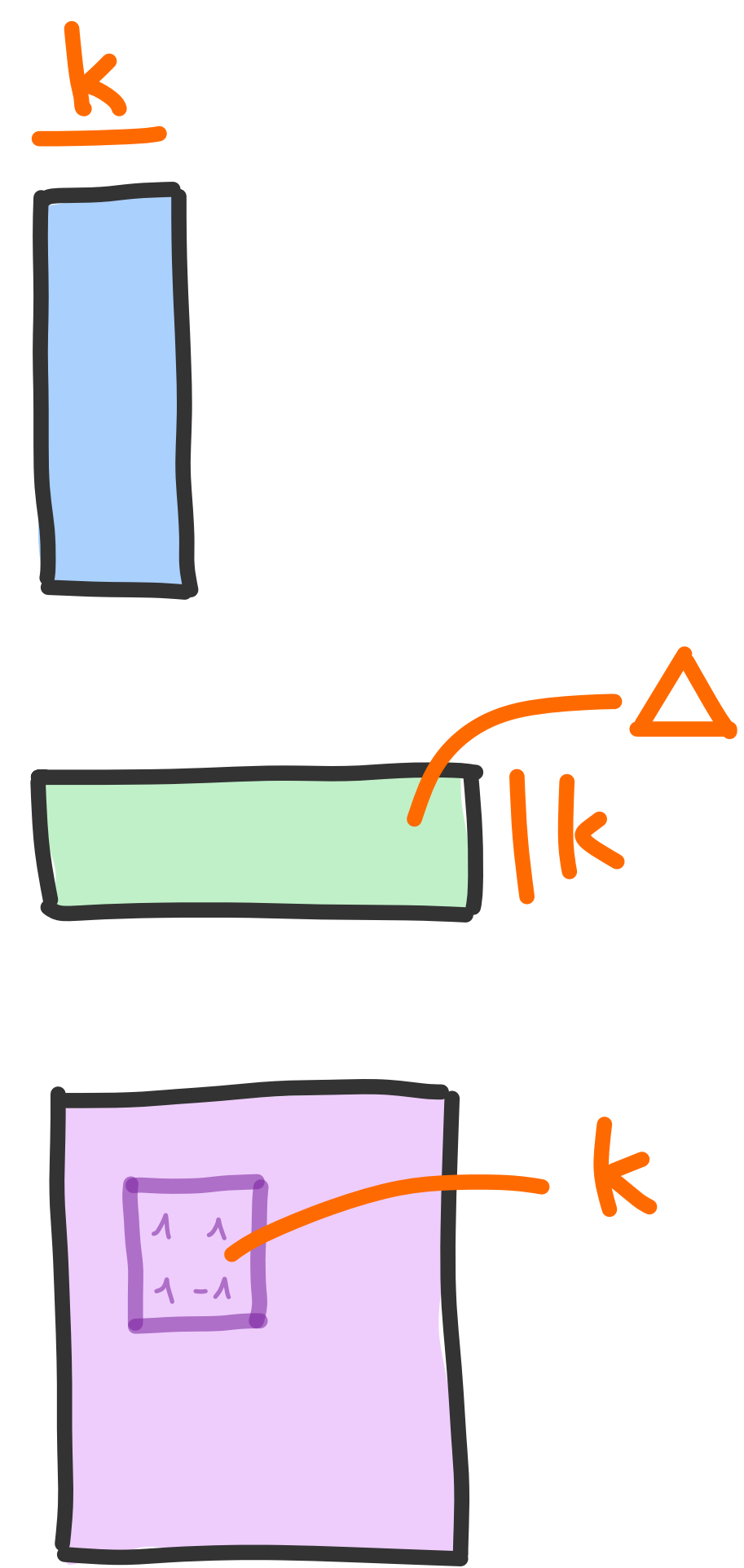


1. Breaking the bricks

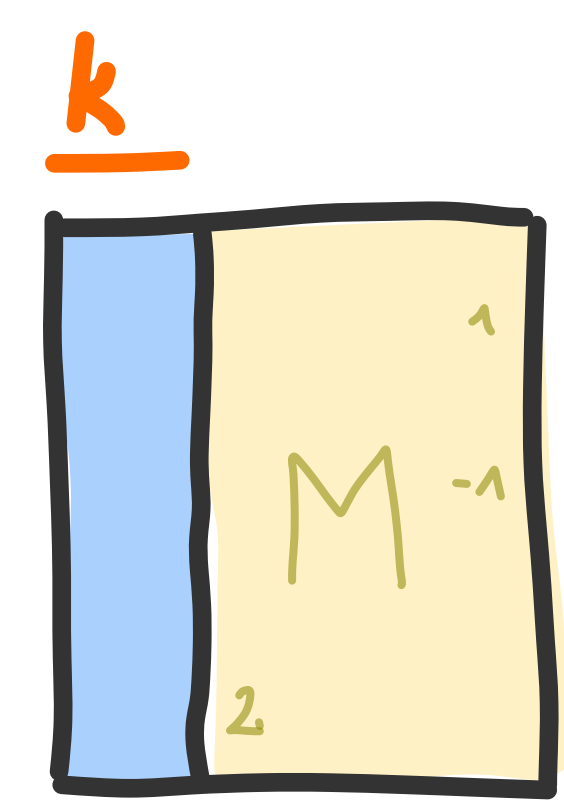
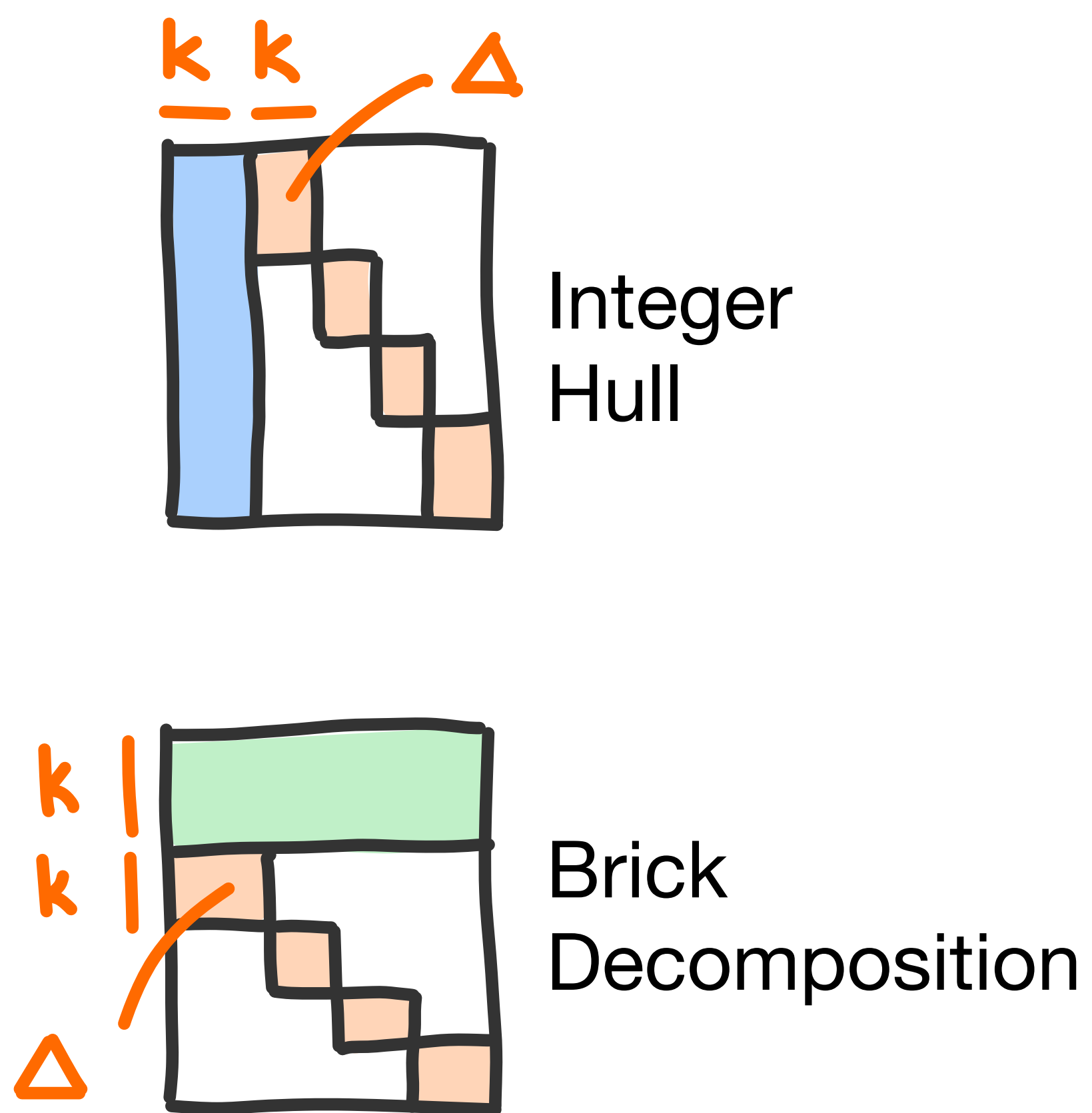
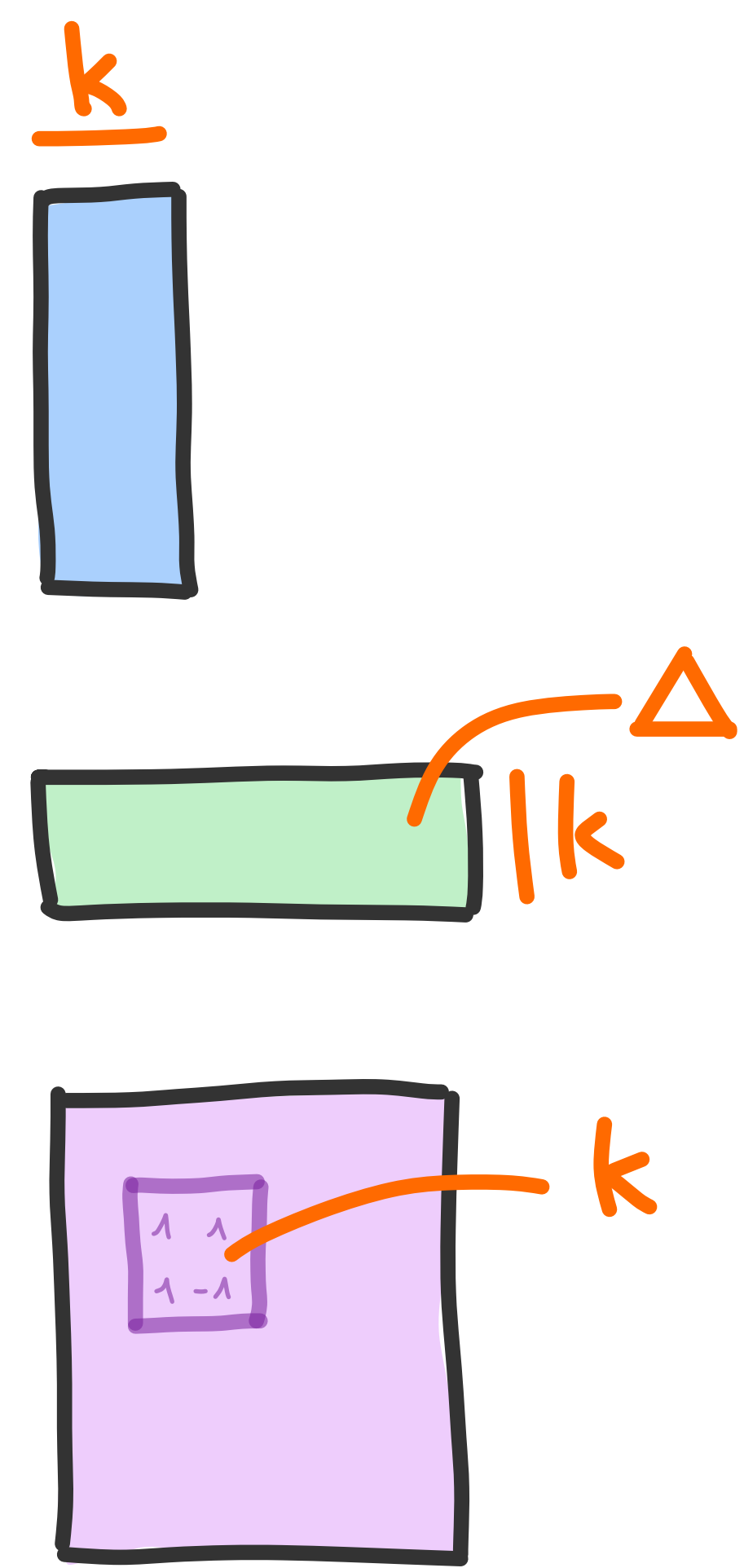
2. We can deal easily with the high-multiplicity instance :)



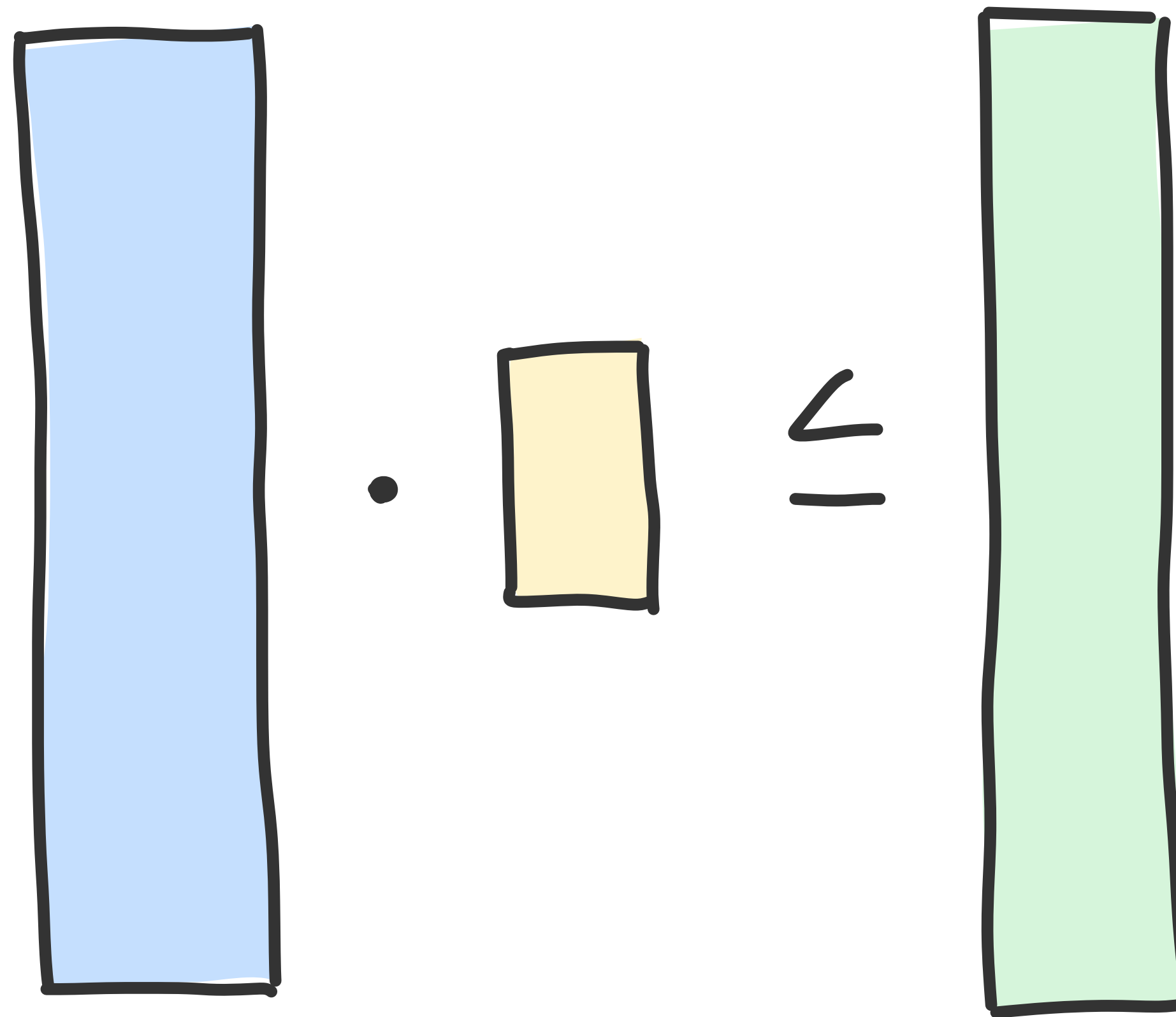
Integer Programming meets FPT



Integer Programming meets FPT



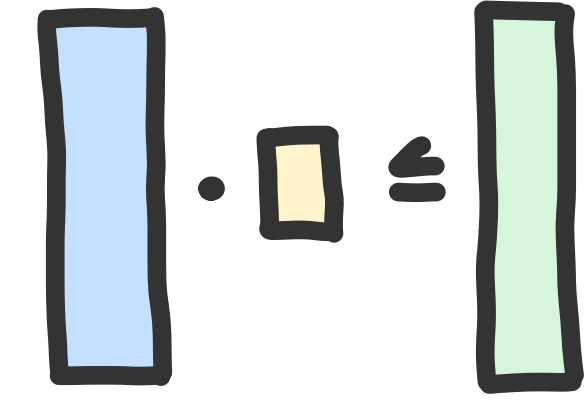
Lenstra-type Integer Programs



A diagram illustrating a Lenstra-type Integer Program. It features three vertical rectangles: a tall blue rectangle on the left, a small yellow rectangle in the middle, and a tall green rectangle on the right. A dot is positioned between the blue and yellow rectangles, and a less-than-or-equal-to symbol (\leq) is positioned between the yellow and green rectangles.

$$\text{Blue Rectangle} \cdot \text{Yellow Rectangle} \leq \text{Green Rectangle}$$

Lenstra-type Integer Programs



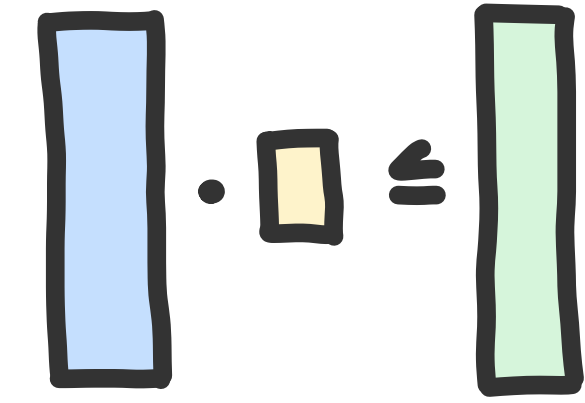
2^{n^3} Lenstra '83

$2^n n^{2.5n}$ Kannan '87

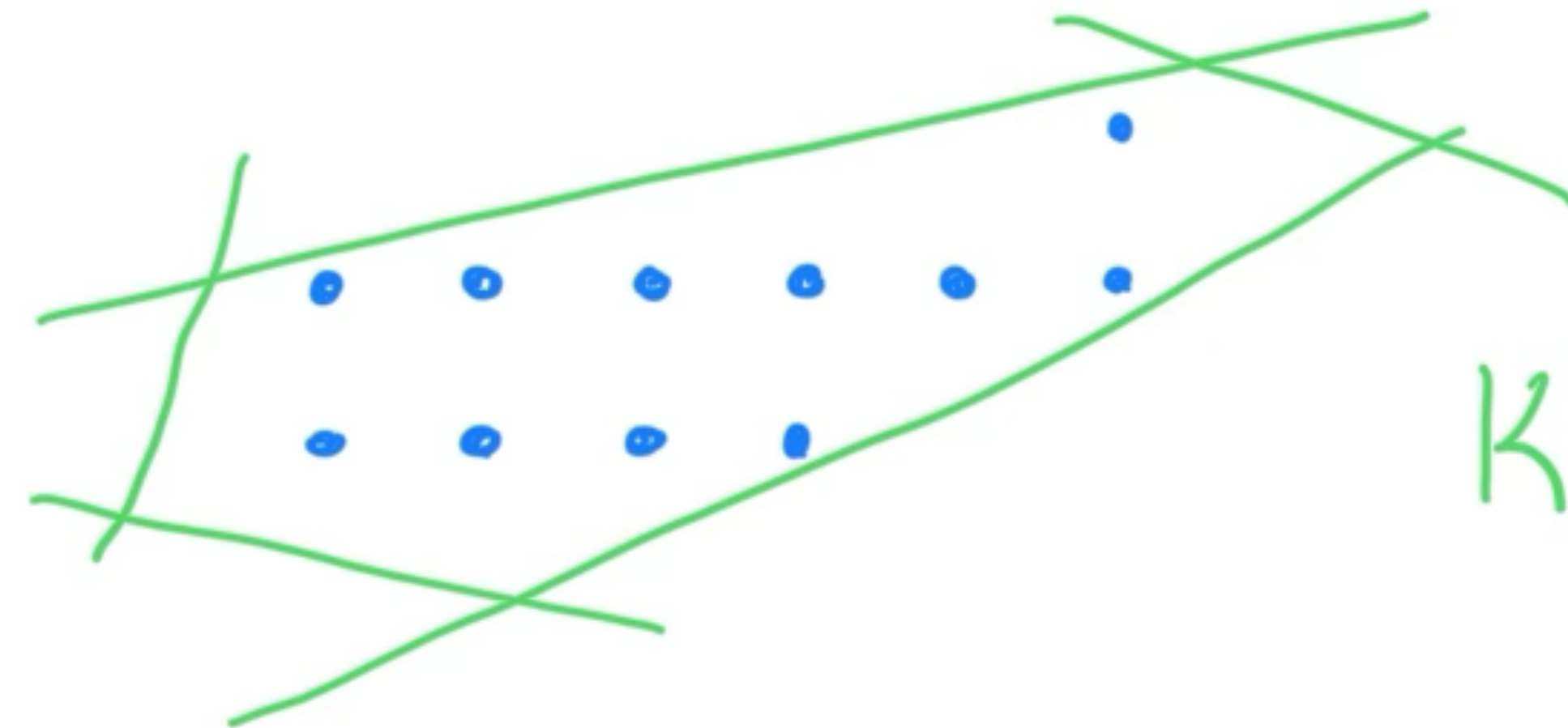
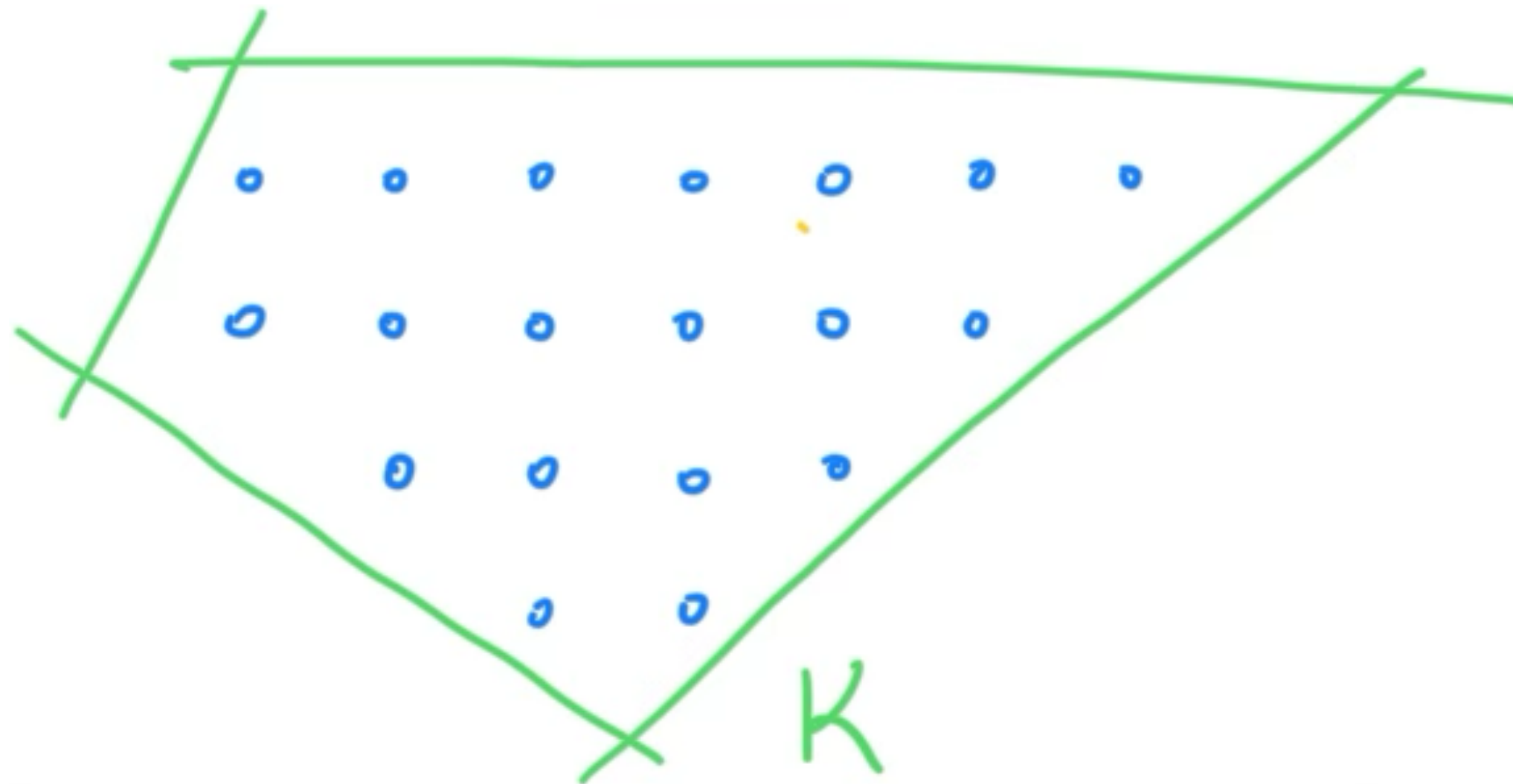
$2^n n^n$ Dadush, Peikert, Vempala '11, Dadush '12

$\log(n)^n$ Rothvoss, Reis '23

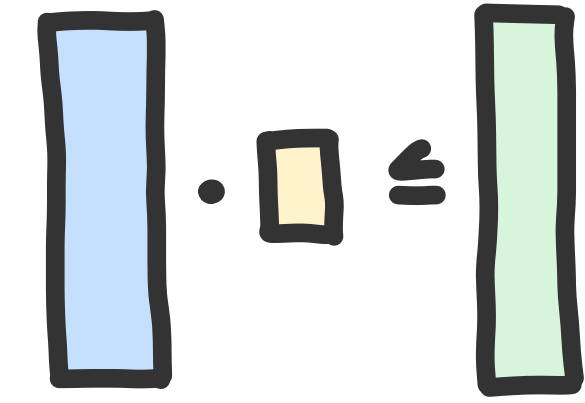
Lenstra-type Integer Programs



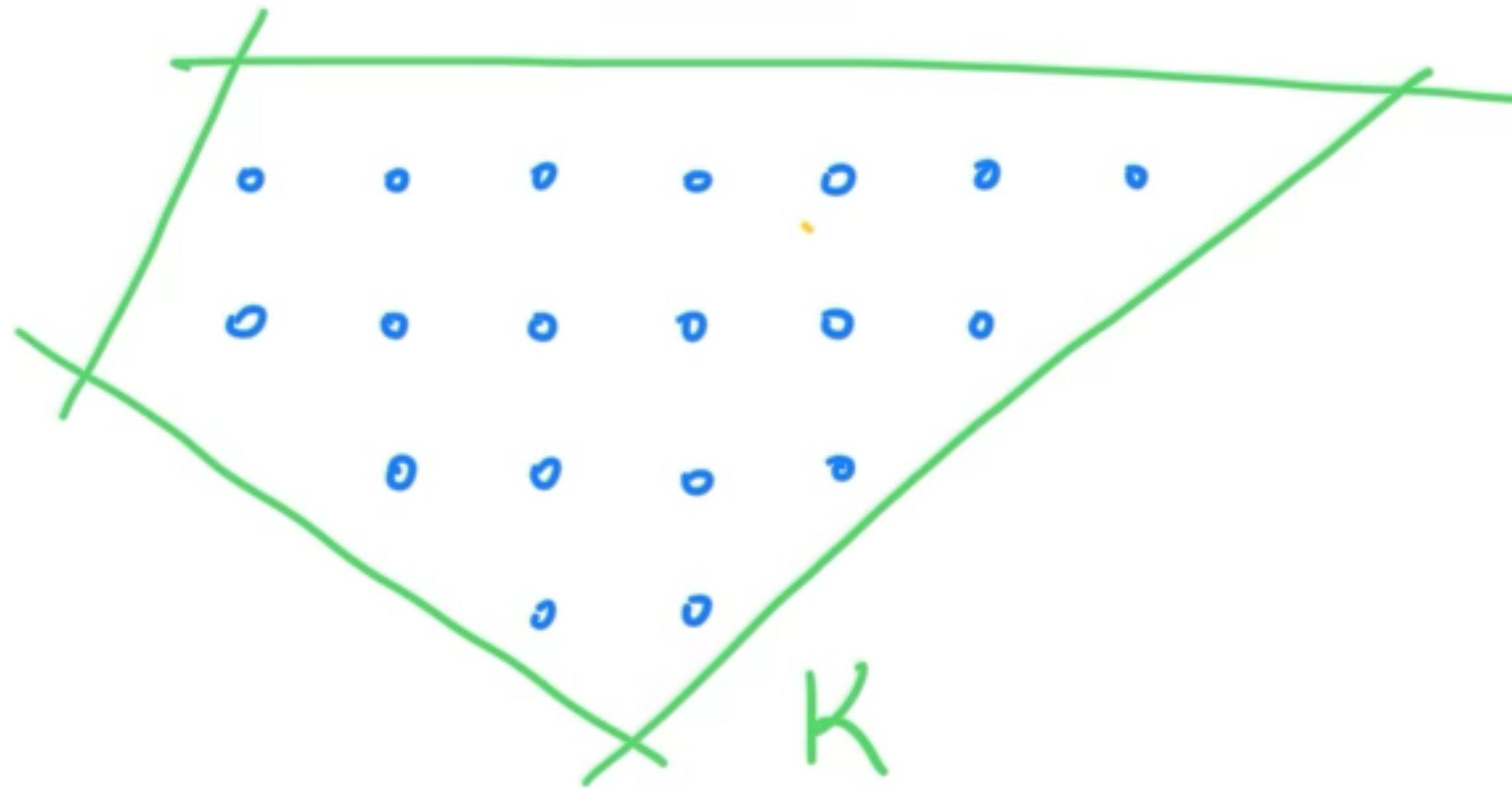
1. Decide whether K is “fat” or “flat”



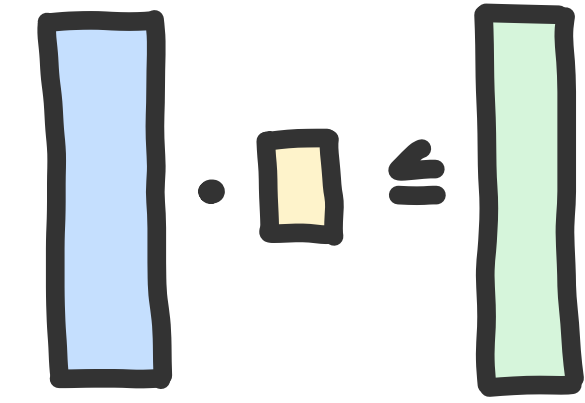
Lenstra-type Integer Programs



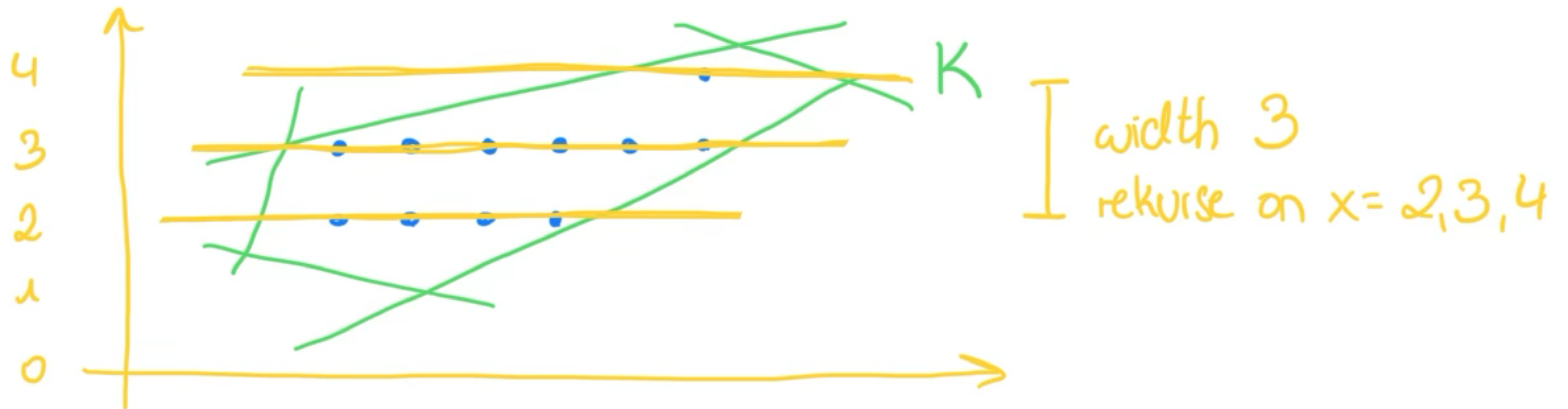
1.1. If “fat”: easy to find solution



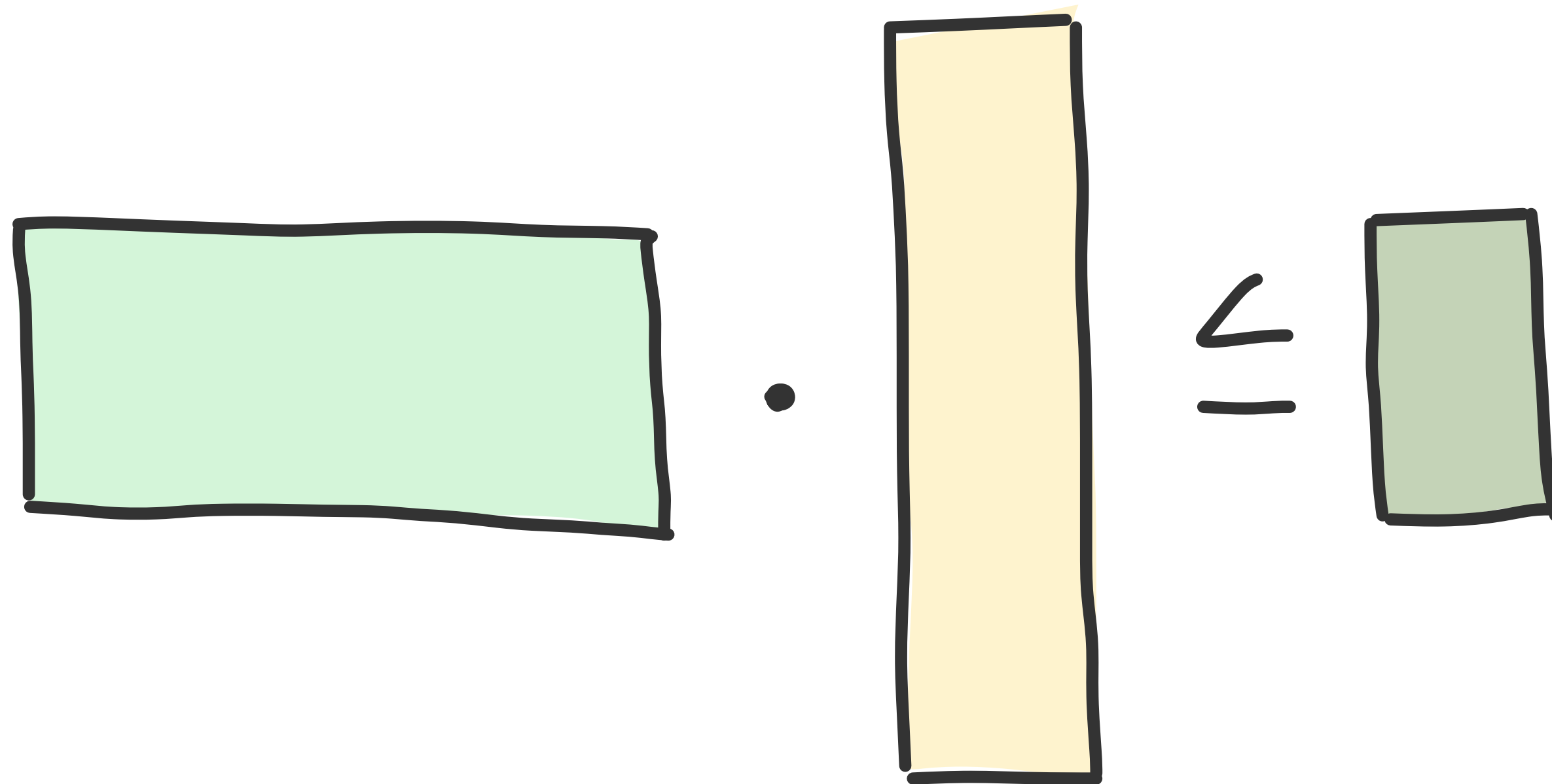
Lenstra-type Integer Programs



1.2. If “flat”: find flat dimension and decompose

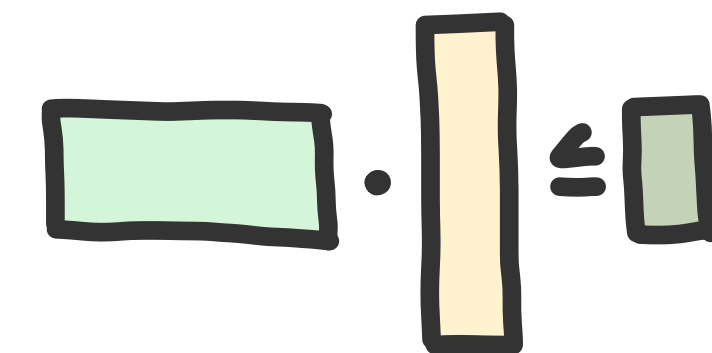


Papadimitriou-type Integer Programs



A diagram illustrating a Papadimitriou-type Integer Program inequality. It consists of three hand-drawn rectangles with black outlines. The first rectangle on the left is light green and is wider than it is tall. To its right is a small black dot representing a multiplication symbol. The second rectangle is light yellow and is very tall and narrow. To its right is a less-than-or-equal-to symbol (\leq). The final rectangle on the right is a darker green and is wider than it is tall, similar in proportions to the first rectangle.

Papadimitriou-type Integer Programs



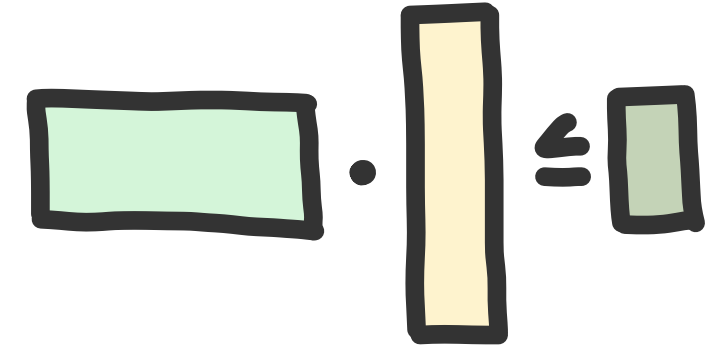
$$n^{2m+2} \cdot (m\Delta + m ||b||^\infty)^{(m+1)(2m+1)} \quad \text{Papadimitriou '81}$$

$$n \cdot (m\Delta)^{2m} \cdot ||b||_1^2 \quad \text{Eisenbrand, Weismantel '18}$$

$$O(nm) \cdot (m\Delta)^{2m} \cdot \log(||b||_\infty) \quad \text{Jansen, Rohwedder '19}$$

$$O(nm) \cdot (\sqrt{m}\Delta)^{2m} \quad \text{Jansen Rohwedder '22}$$

Papademitriou-type Integer Programs

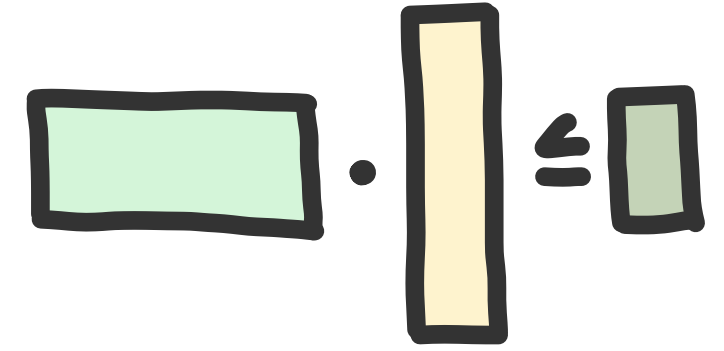


Graver elements: inclusion—wise
minimal (\subseteq) kernel elements

$$\subseteq : x_i \leq y_i \quad \forall i$$

and sign—compatible

Papademitriou-type Integer Programs



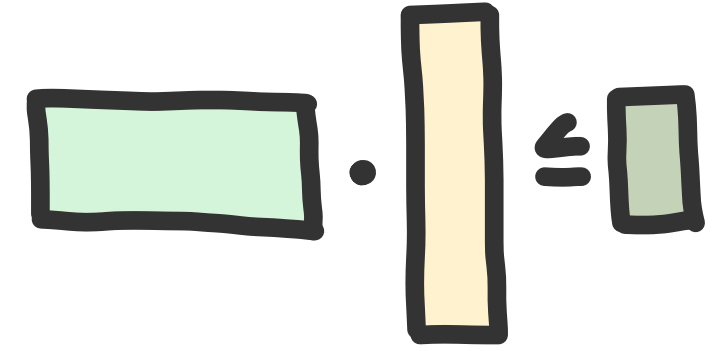
Graver elements: inclusion—wise
minimal (\sqsubseteq) kernel elements

$$\sqsubseteq : x_i \leq y_i \quad \forall i$$

and sign—compatible

$$\begin{pmatrix} 3 \\ 5 \\ -3 \end{pmatrix} \sqsubseteq \begin{pmatrix} 5 \\ 5 \\ -4 \end{pmatrix} \quad \begin{pmatrix} 3 \\ -2 \\ 4 \end{pmatrix} \not\sqsubseteq \begin{pmatrix} 5 \\ 0 \\ 2 \end{pmatrix}$$

Papademitriou-type Integer Programs



Graver elements: inclusion—wise
minimal (\sqsubseteq) kernel elements

$$\sqsubseteq : x_i \leq y_i \quad \forall i$$

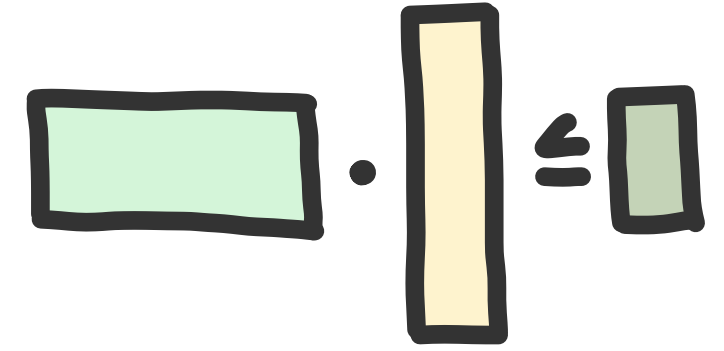
and sign—compatible

$$\begin{pmatrix} 3 \\ 5 \\ -3 \end{pmatrix} \sqsubseteq \begin{pmatrix} 5 \\ 5 \\ -4 \end{pmatrix} \quad \begin{pmatrix} 3 \\ -2 \\ 4 \end{pmatrix} \not\sqsubseteq \begin{pmatrix} 5 \\ 0 \\ 2 \end{pmatrix}$$

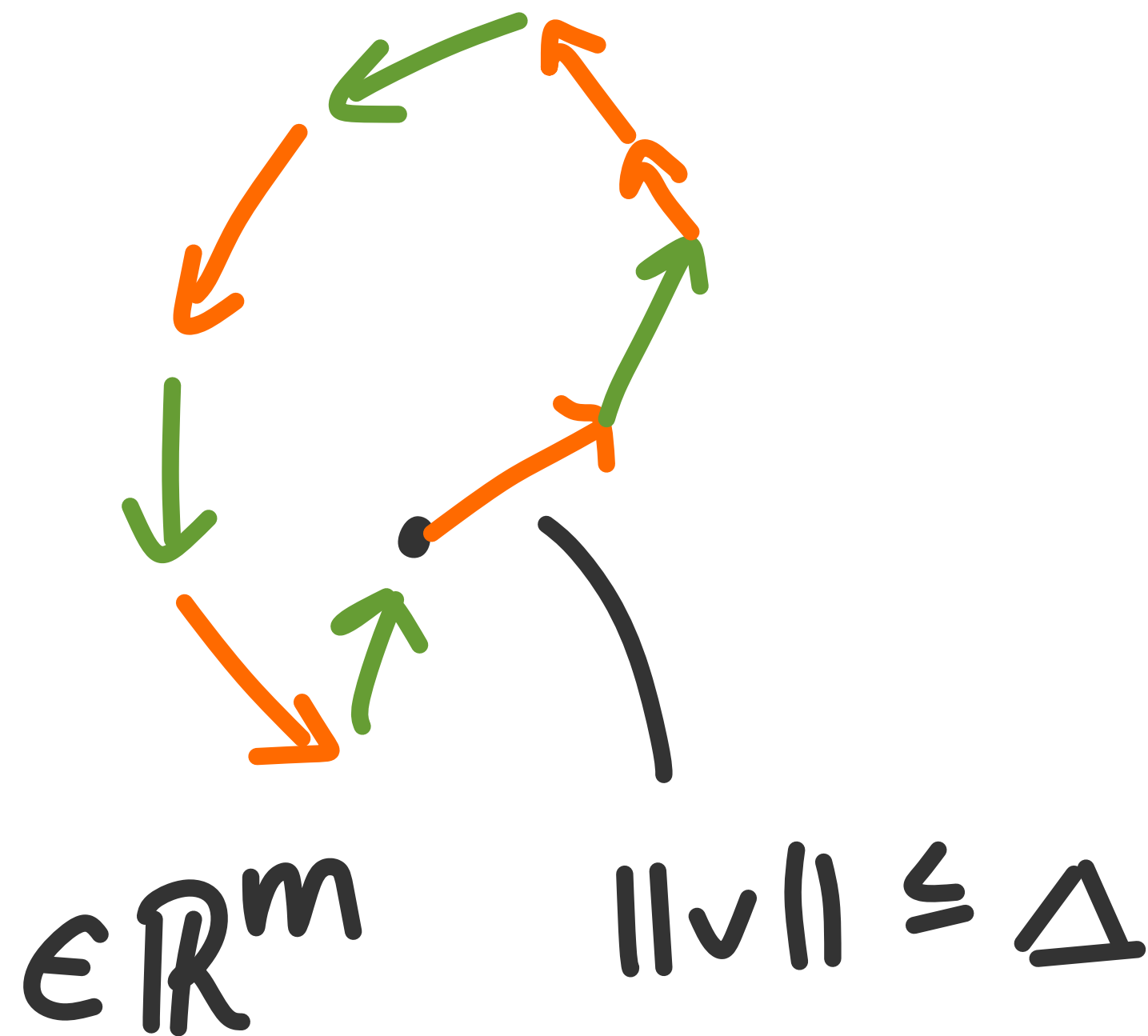
Papadimitriou-type Integer Programs

$\|g\|_\infty \leq \Delta^{f(k)}$ via Steinitz lemma

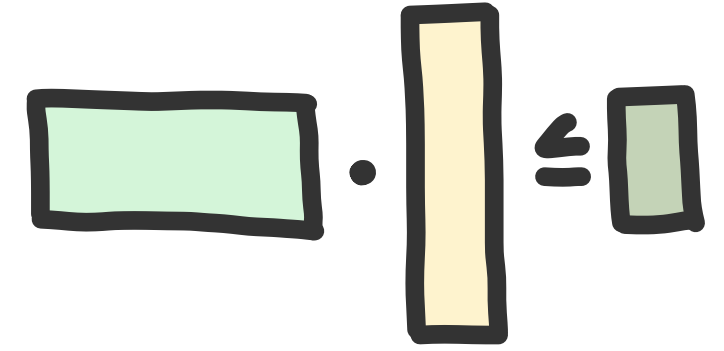
Papadimitriou-type Integer Programs



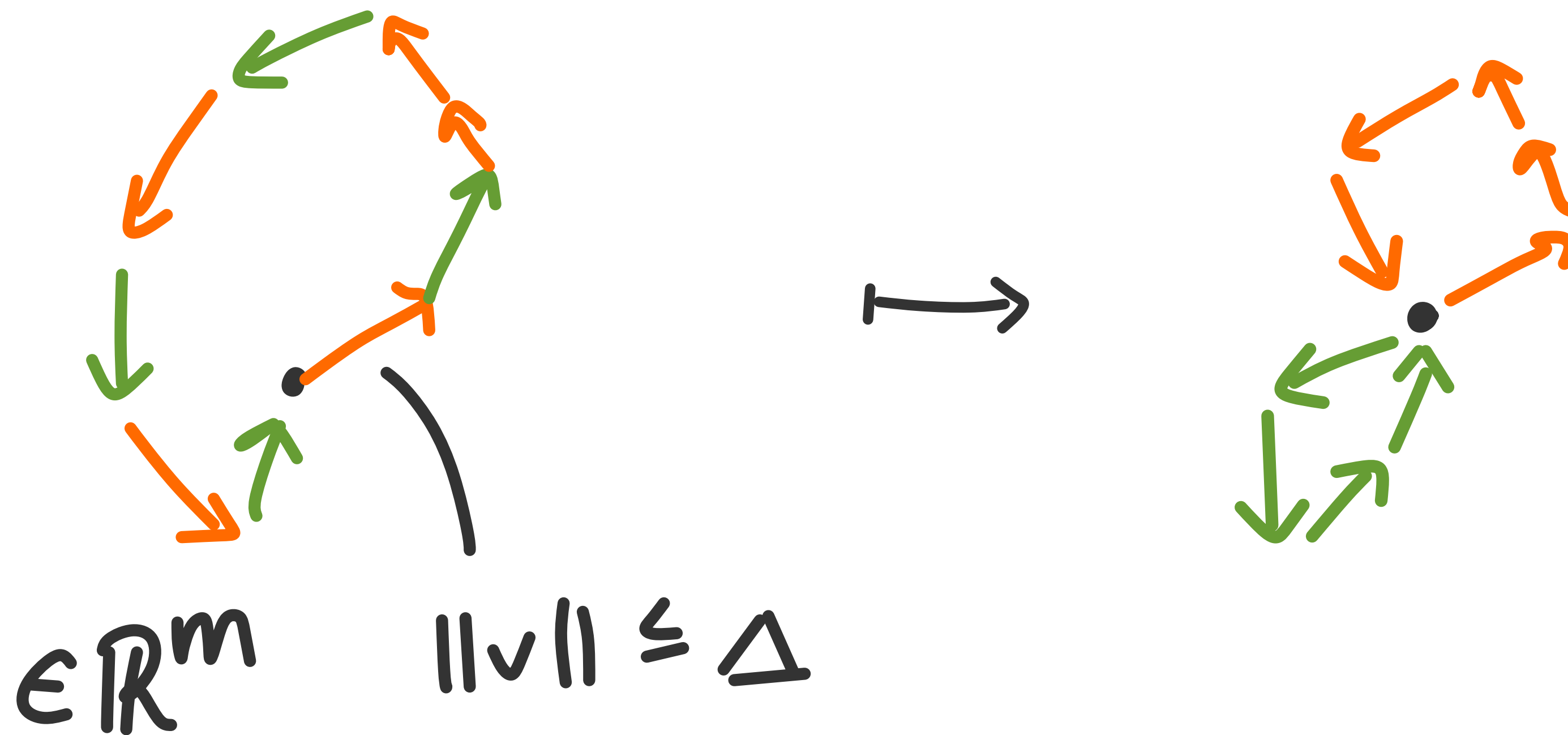
$\|g\|_\infty \leq \Delta^{f(k)}$ via Steinitz lemma



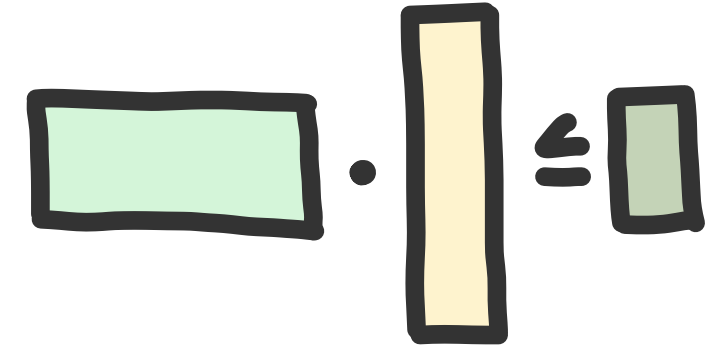
Papadimitriou-type Integer Programs



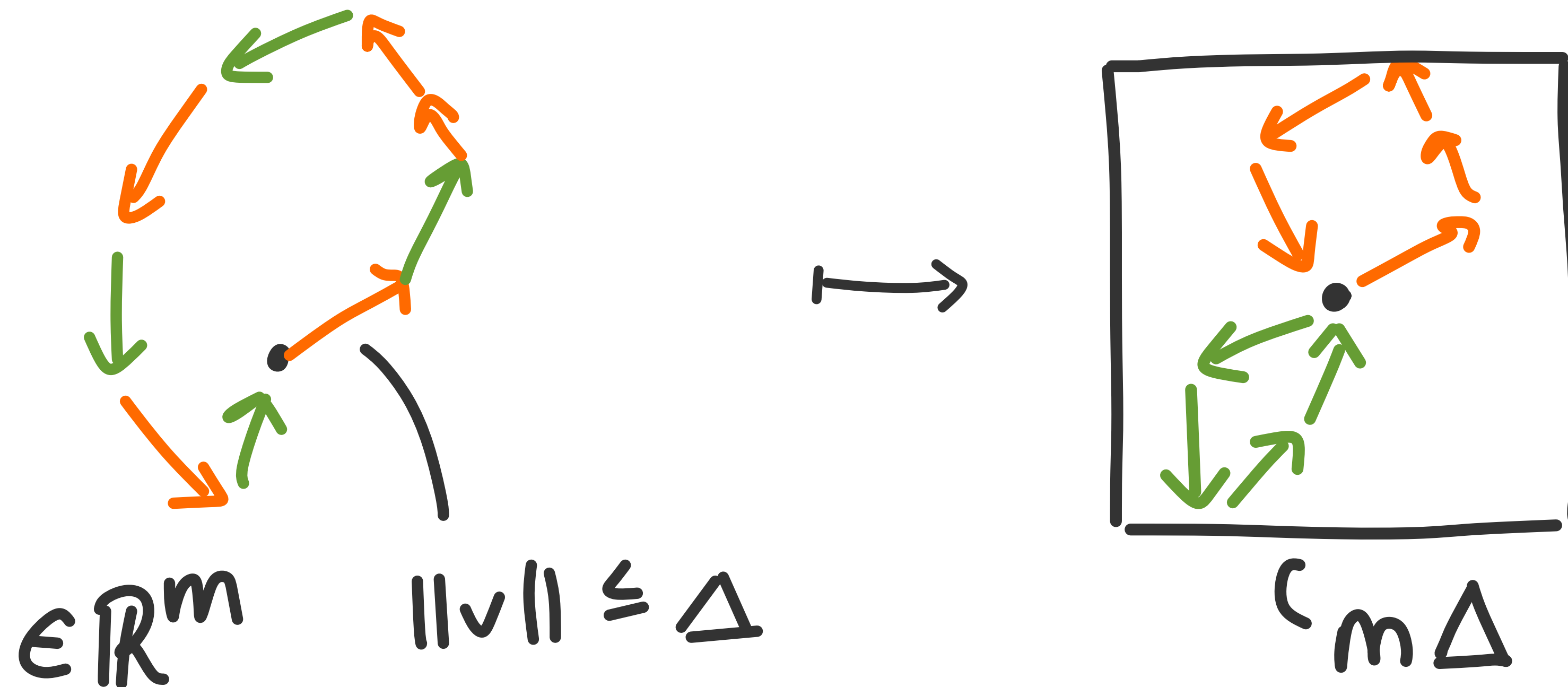
$\|g\|_\infty \leq \Delta^{f(k)}$ via Steinitz lemma



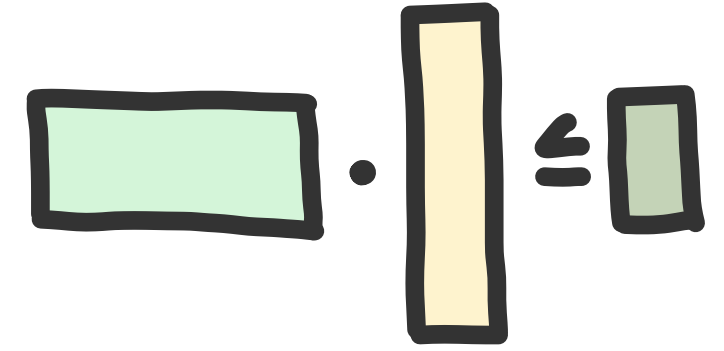
Papademitriou-type Integer Programs



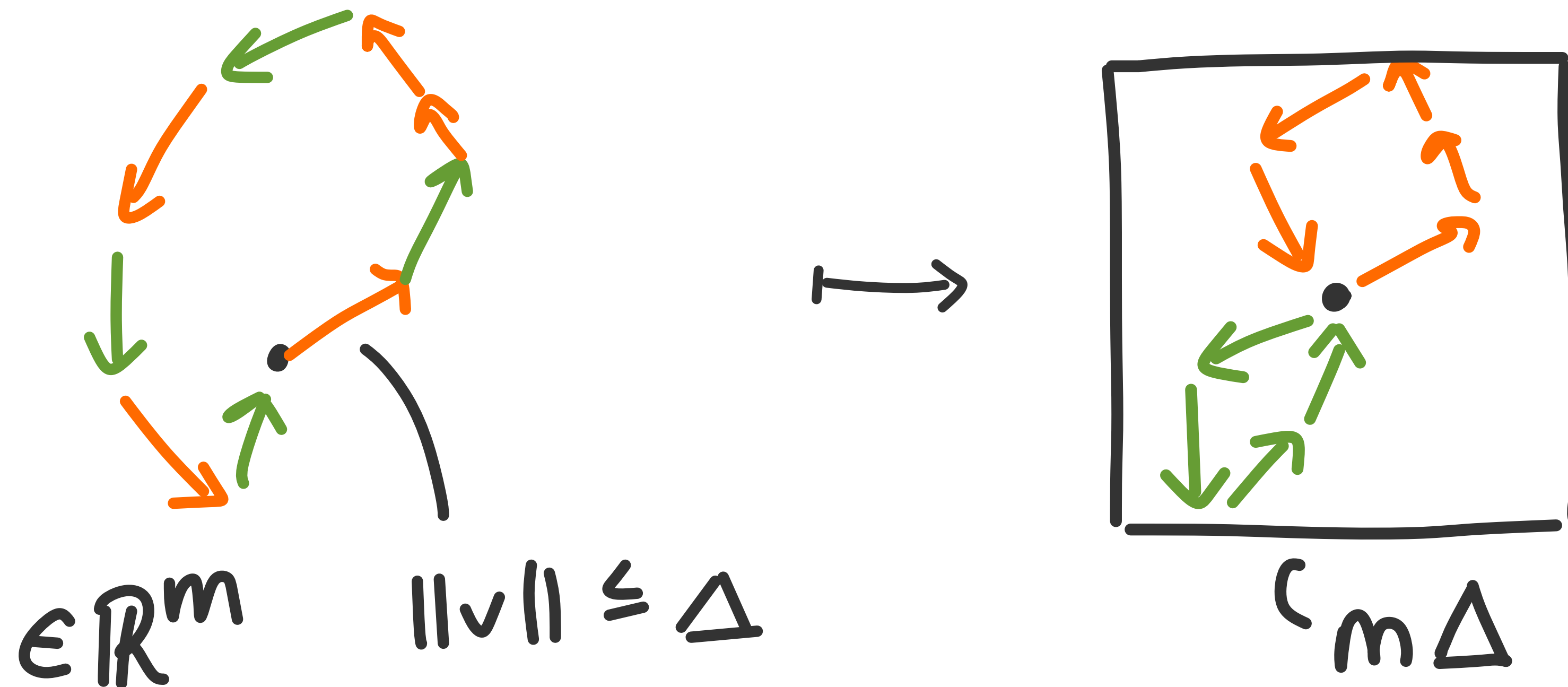
$\|g\|_\infty \leq \Delta^{f(k)}$ via Steinitz lemma

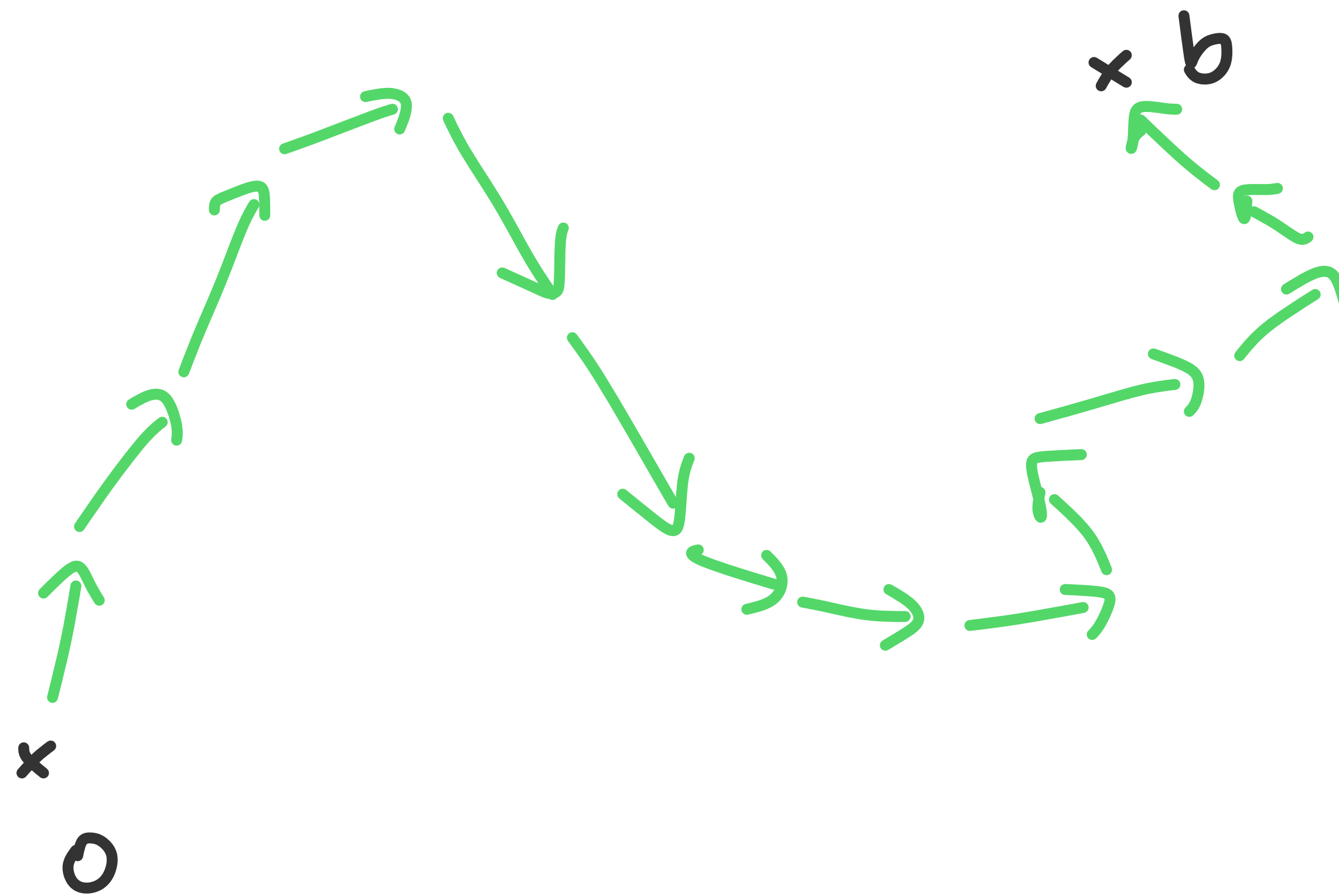


Papademitriou-type Integer Programs



$\|g\|_\infty \leq \Delta^{f(k)}$ via Steinitz lemma





Papademitriou-type Integer Programs

