Intro till gitter / lattices

Kryptografins svärd och sköld

chopingu 🐧

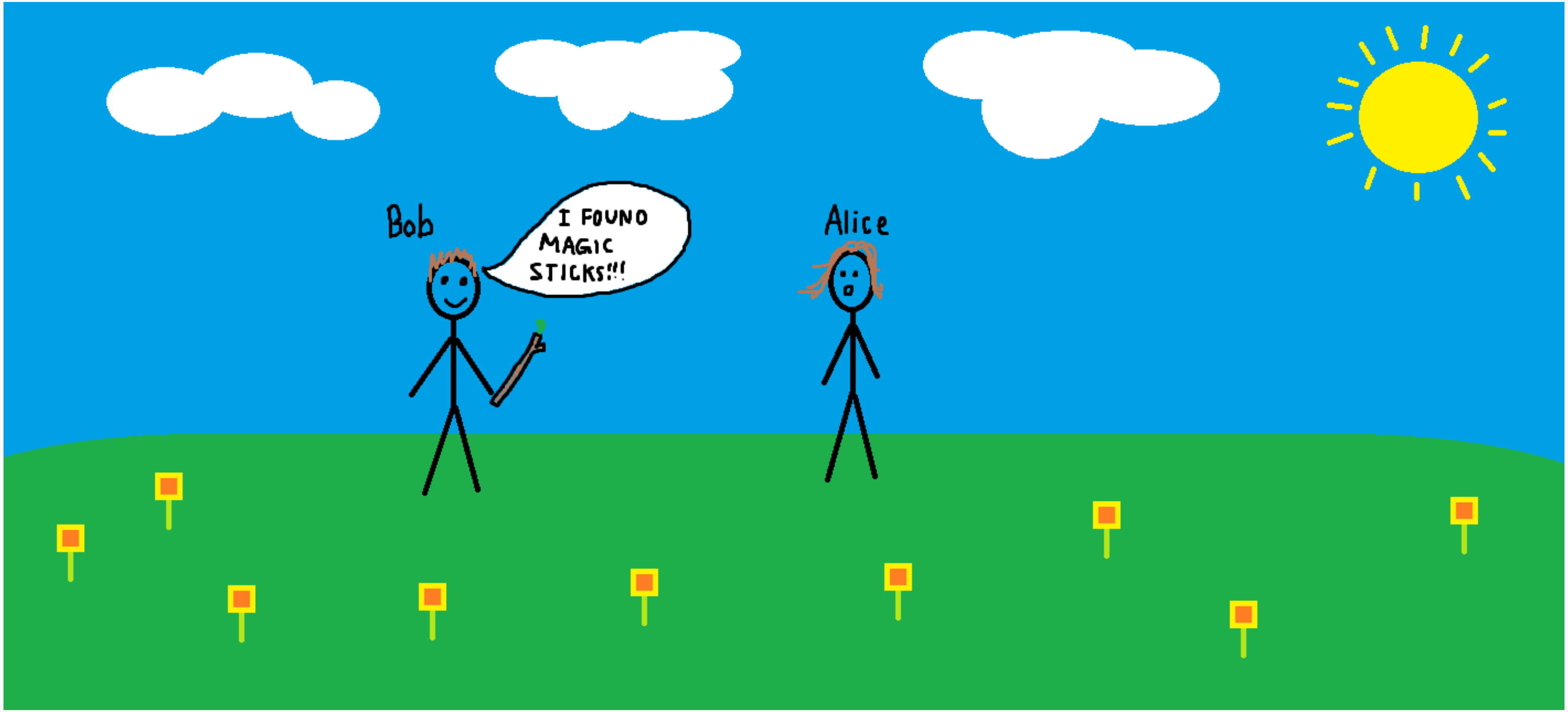
SNHT Bootcamp

Innehåll

1. Magiska pinnar 🌲
2. Matematisk definition
3. Gitter
4. Varför gitter?
5. Hur använder vi svärdet?
6. Exempel
7. Lösa challs

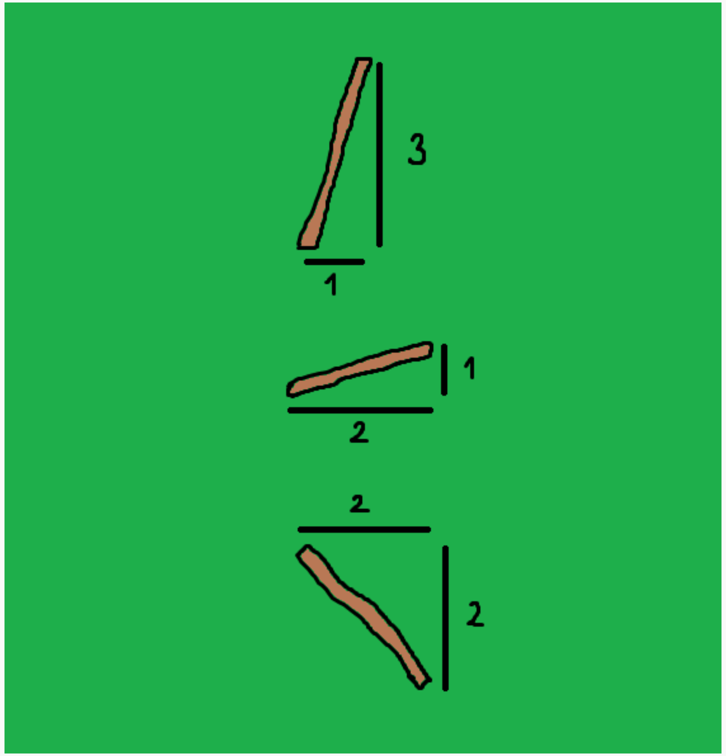
Magiska pinnar 🌲

Magiska pinnar 🌲

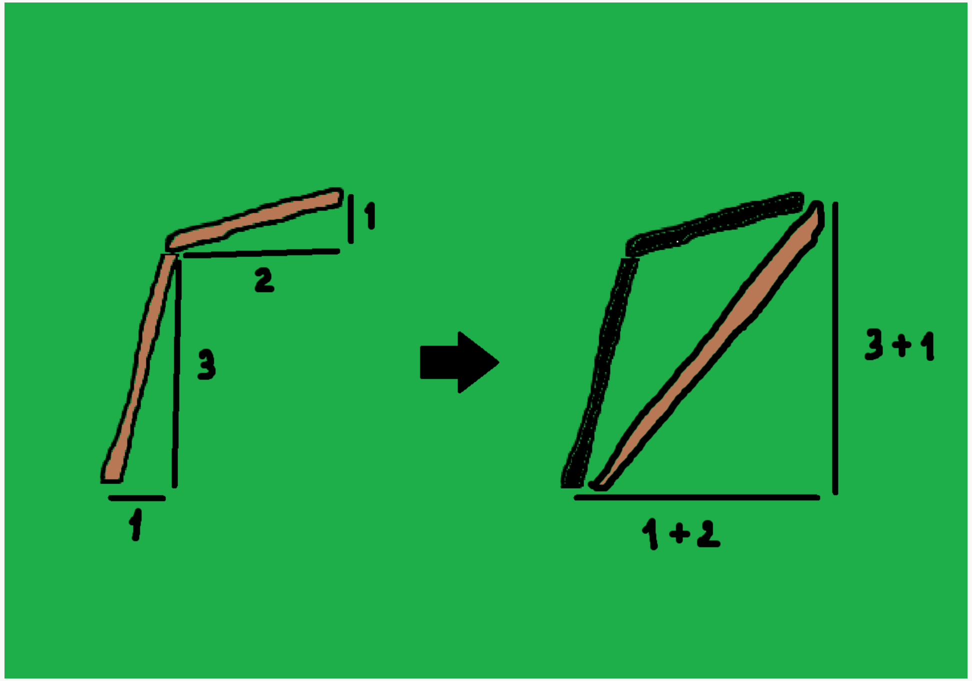


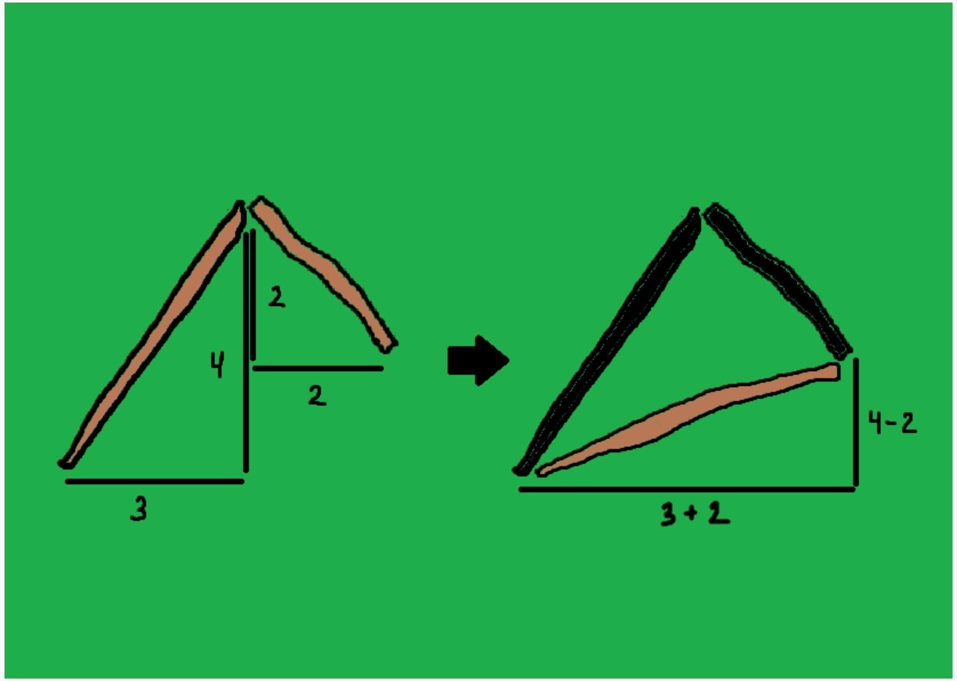
Magiska pinnar 🌲

- 3 magiska pinnar till början
- Förändring i x och y
- Pinnarna kan dupliceras
- Pinnarna kan kombineras

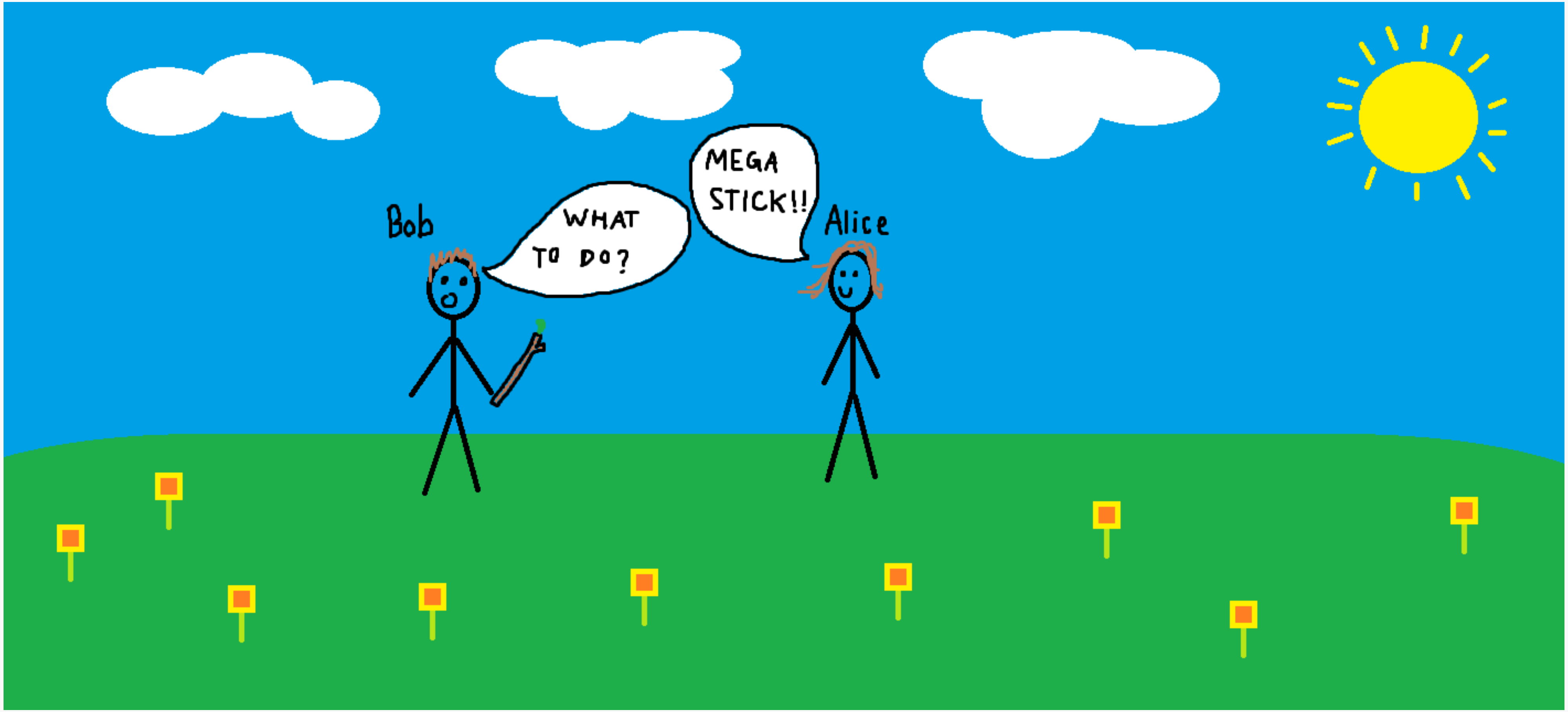


Magiska pinnar 🌲

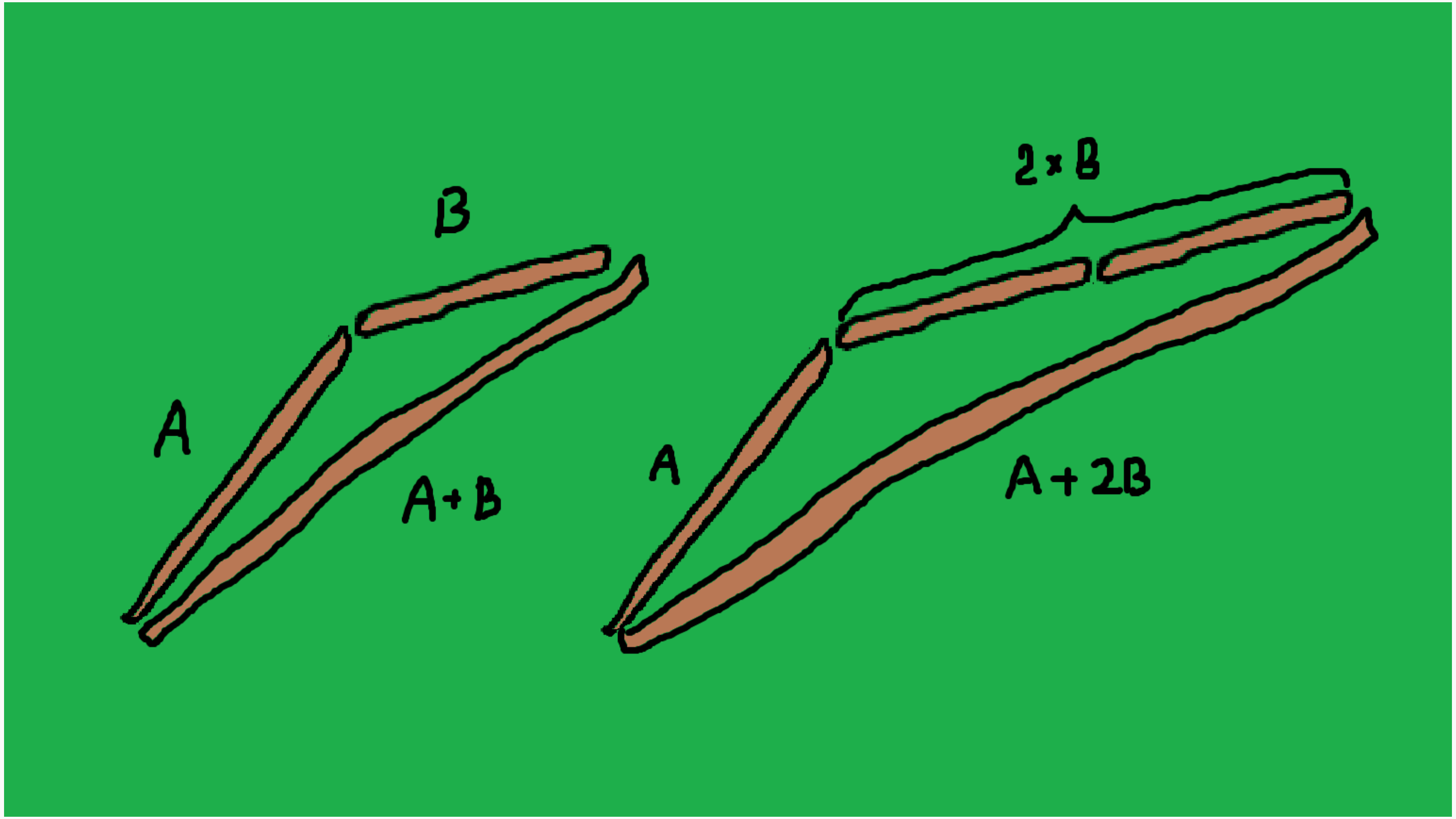




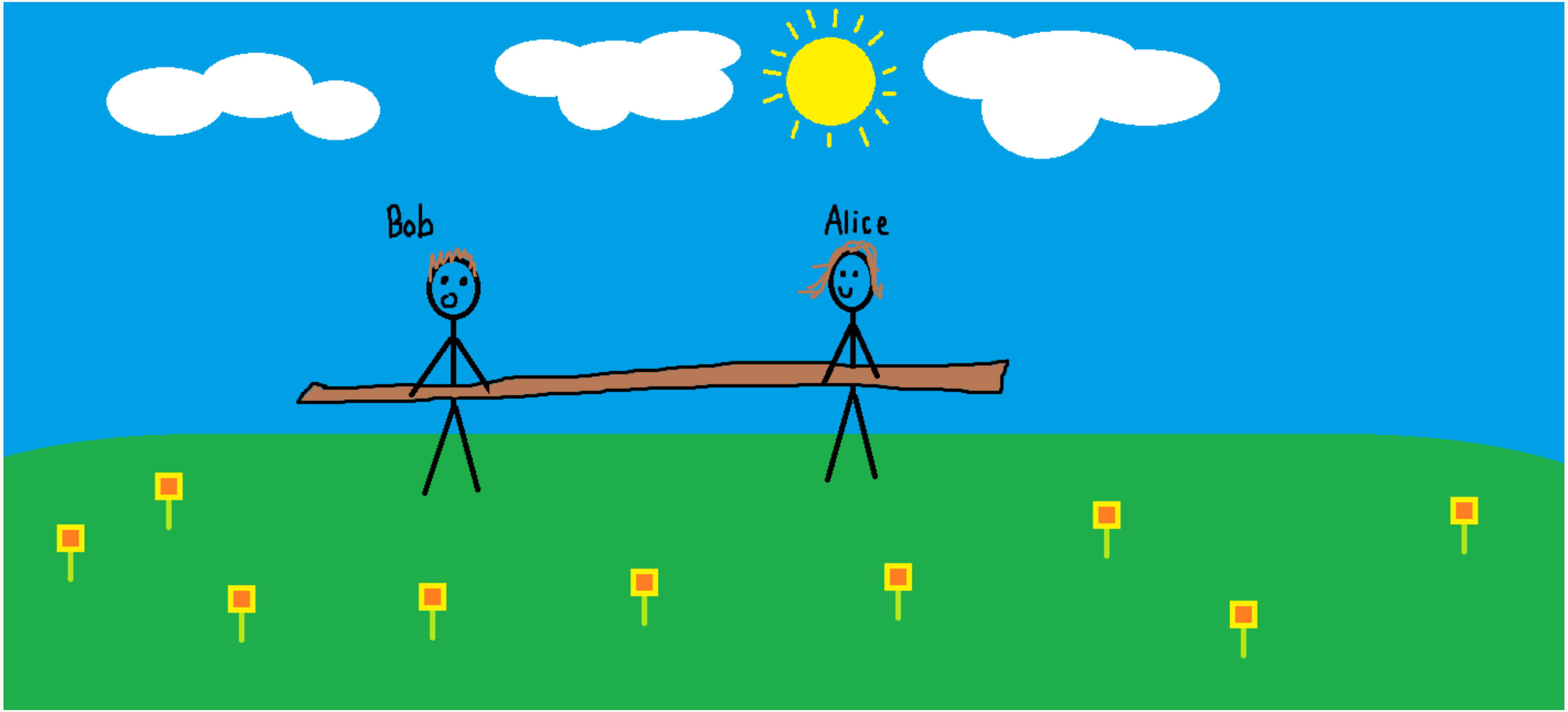
Magiska pinnar 🌲



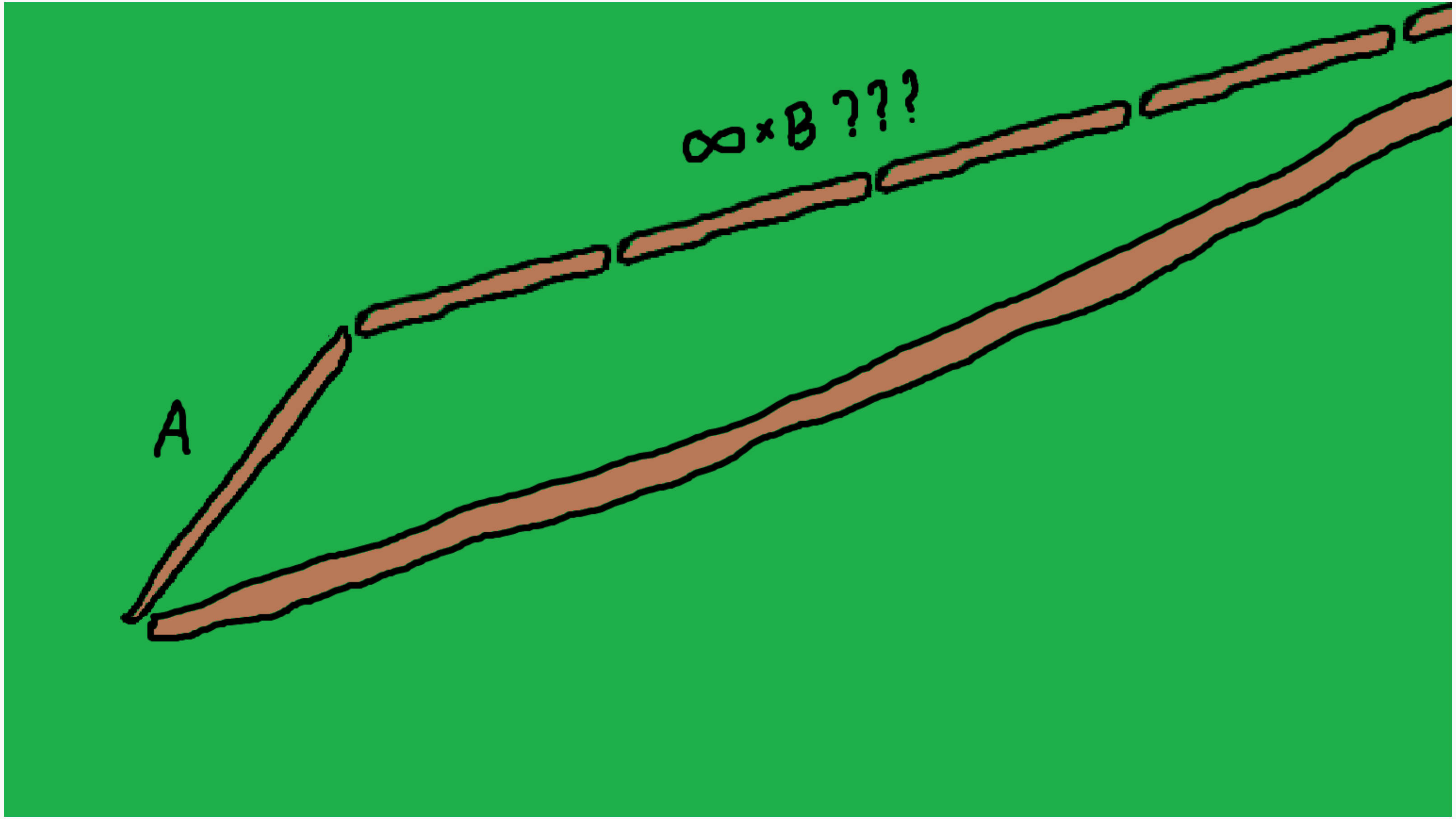
Magiska pinnar 🌲



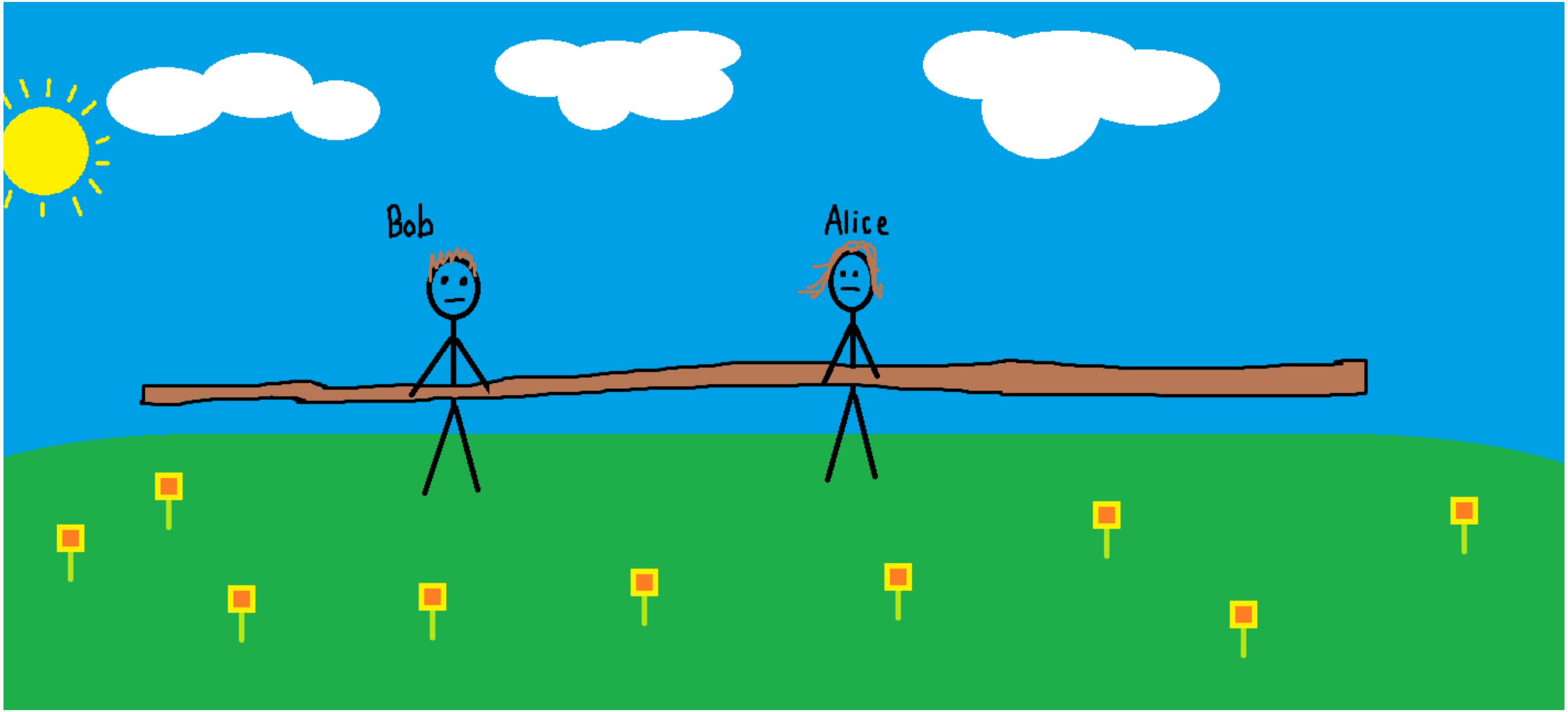
Magiska pinnar 🌲



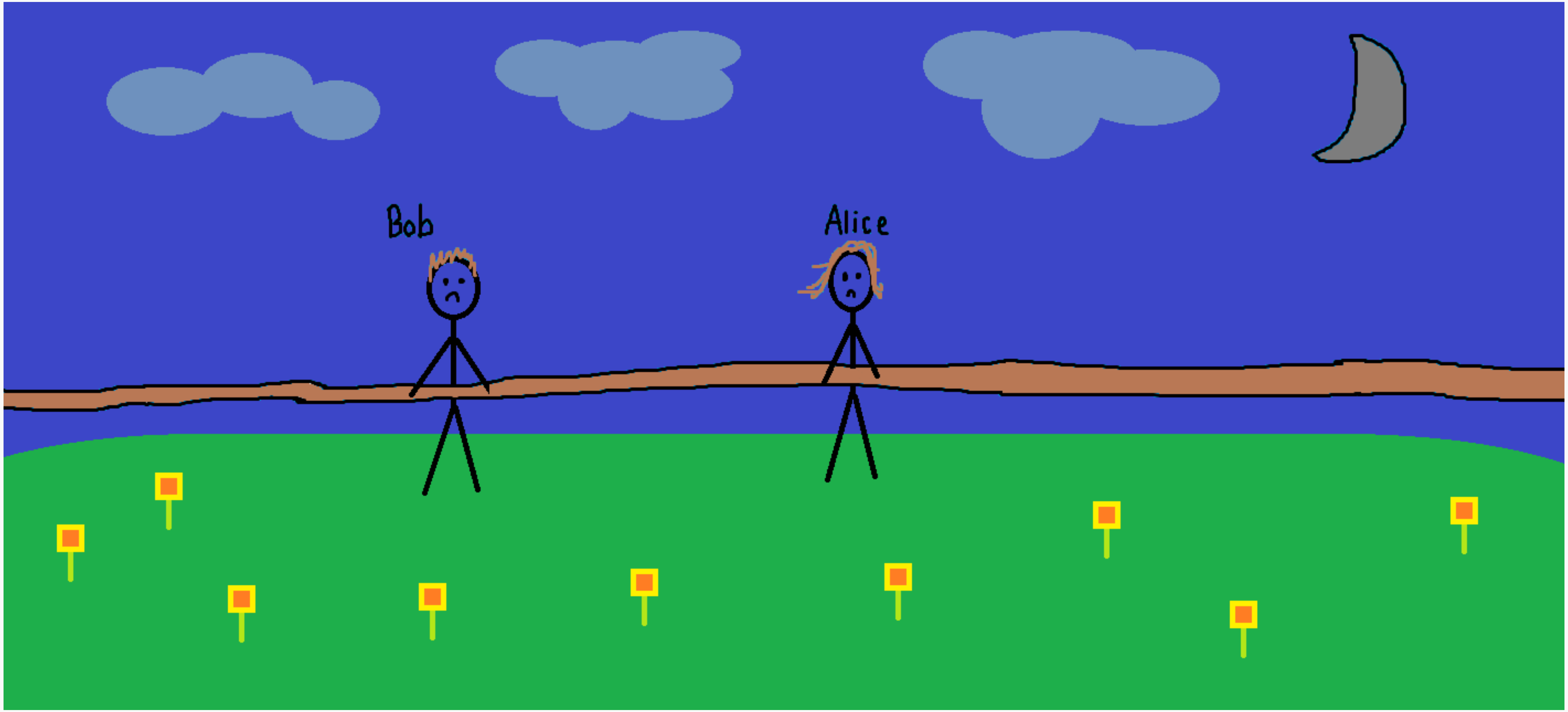
Magiska pinnar 🌲



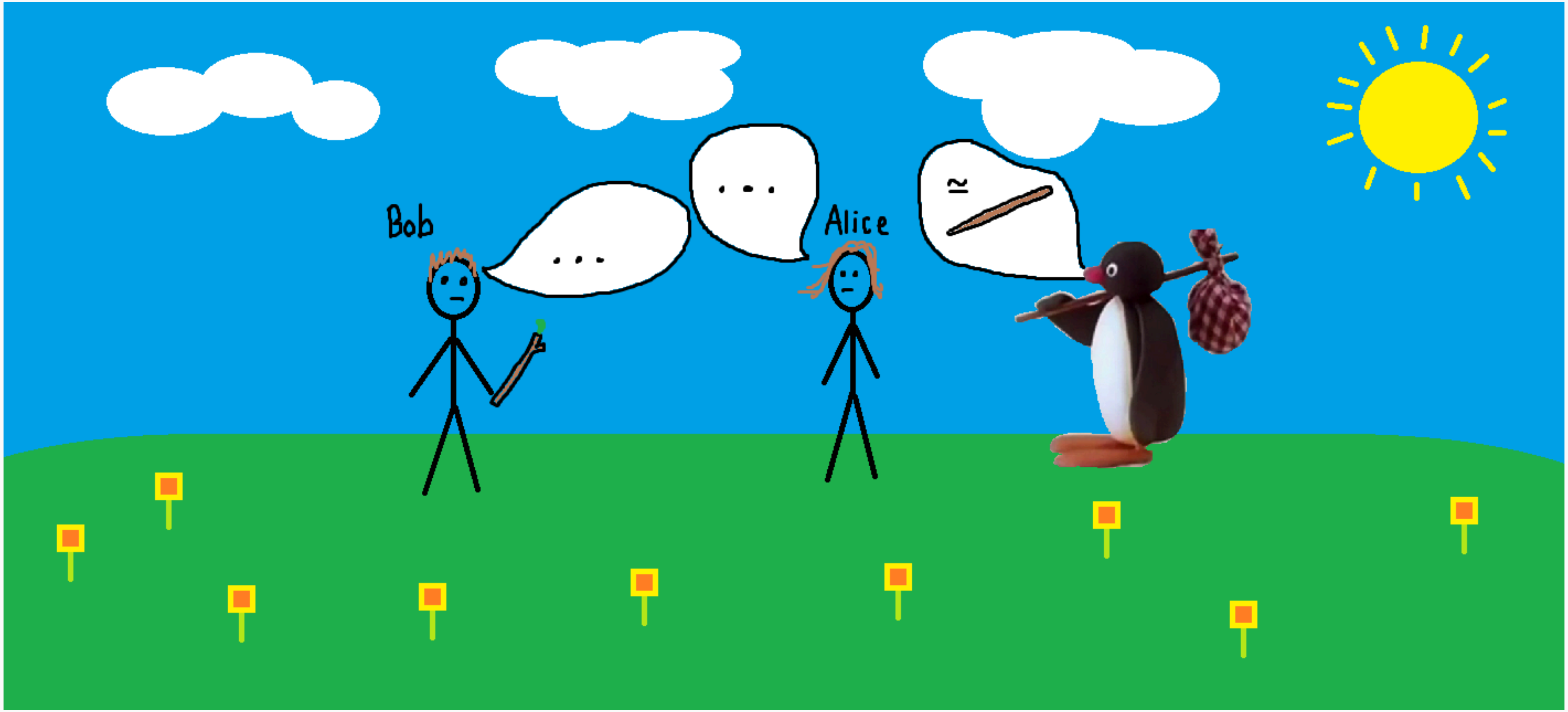
Magiska pinnar 🌲



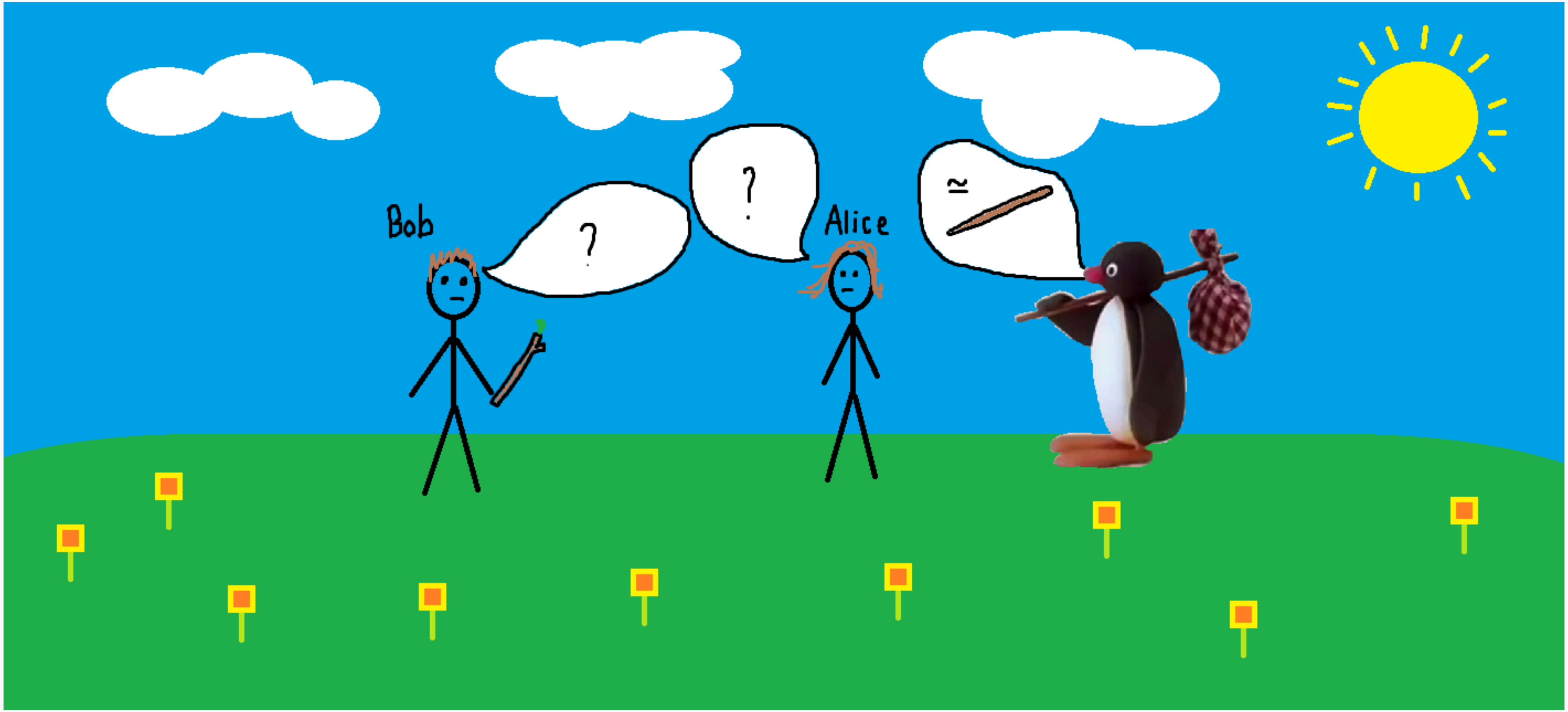
Magiska pinnar 🌲



Magiska pinnar 🌲



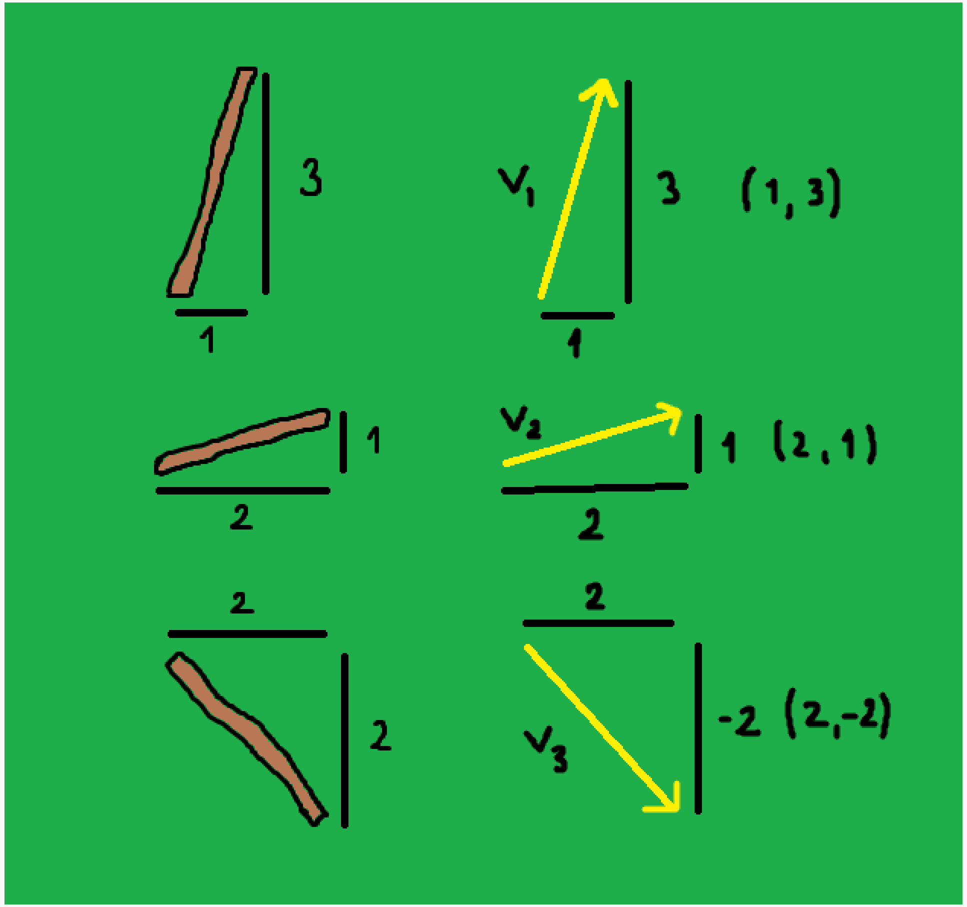
Magiska pinnar 🌲



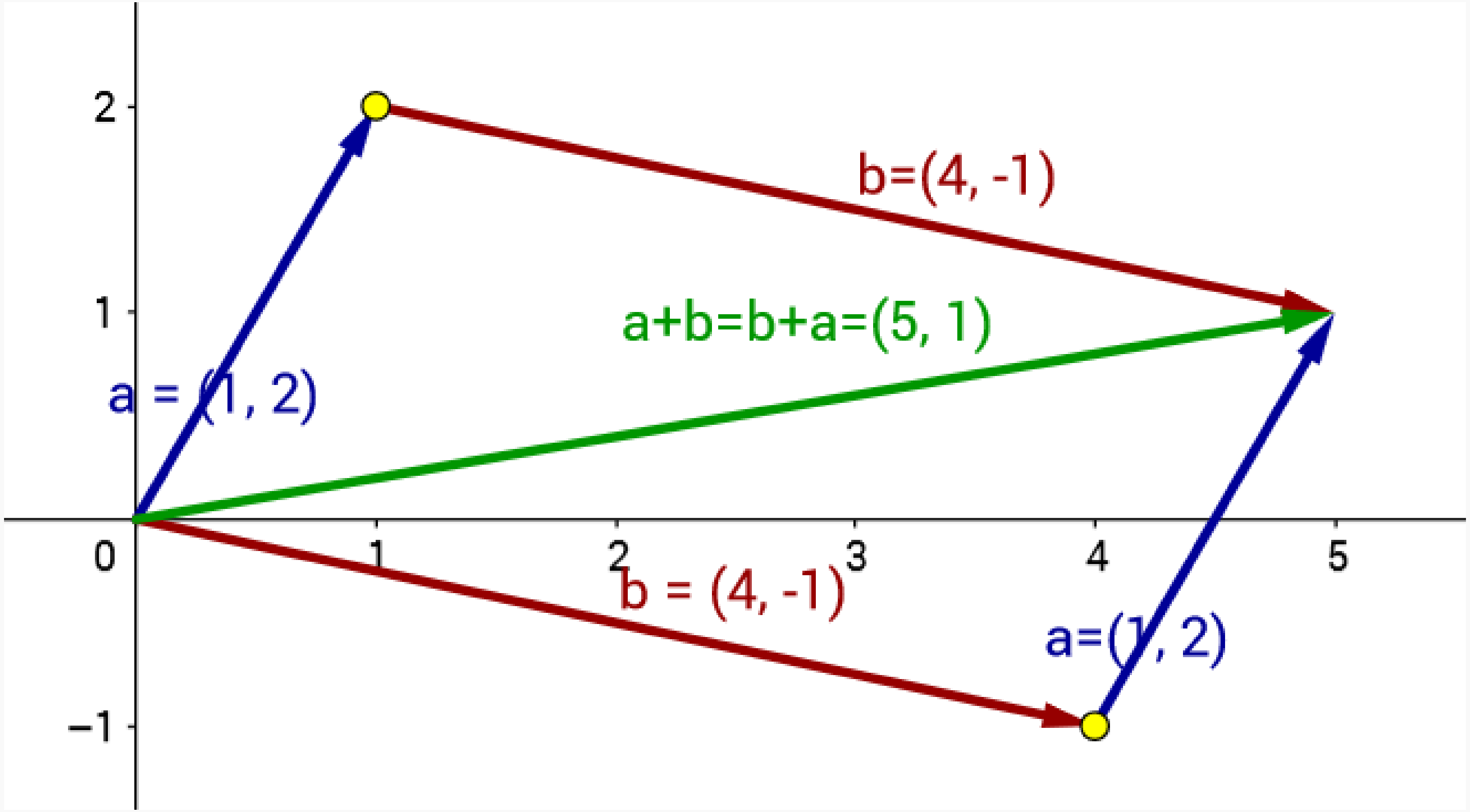
Matematisk definition

Matematisk definition

- Är det bara vektorer?
- Vektor notation $v_i = (\Delta x, \Delta y)$
- Vektor addition som vanligt
- I princip...



Matematisk definition



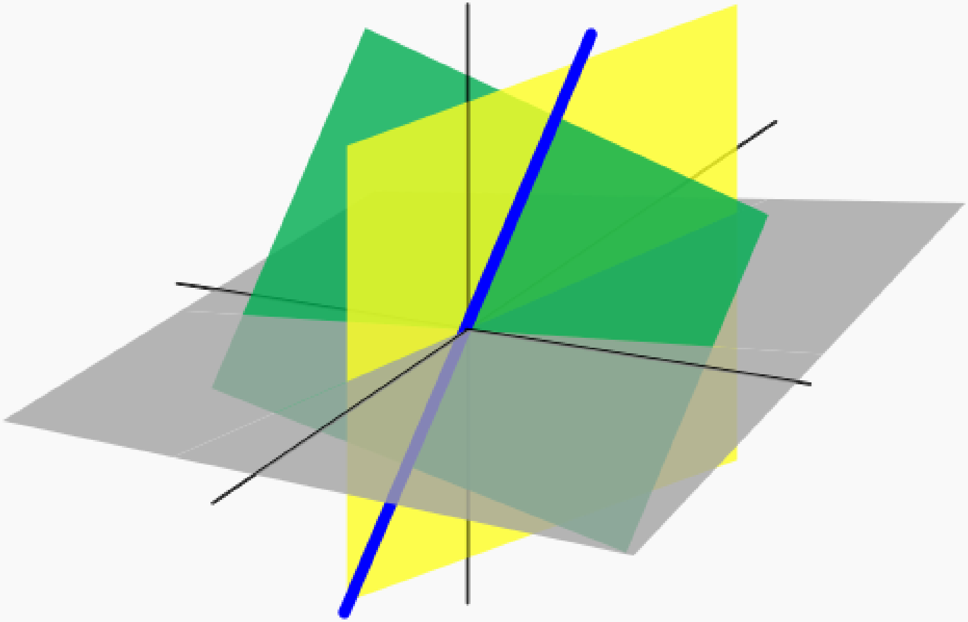
Matematisk definition

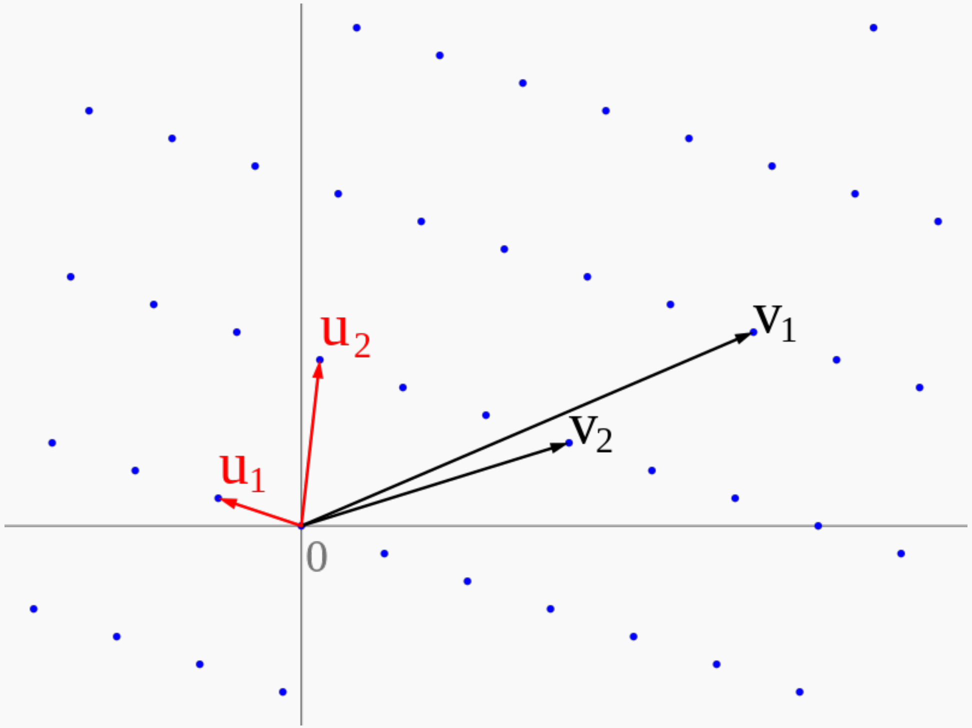
- Varför inte vanlig vektormatte?
- Går inte att såga magiska pinnar
- Heltalskombinationer
- Bildar ett gitter

$$0.5v_1 \text{ eller } \frac{1}{3}v_1$$

$$a_1v_1 + a_2v_2 + a_3v_3$$
$$\text{där } a_{\{1,2,3\}} \in \mathbb{Z}$$

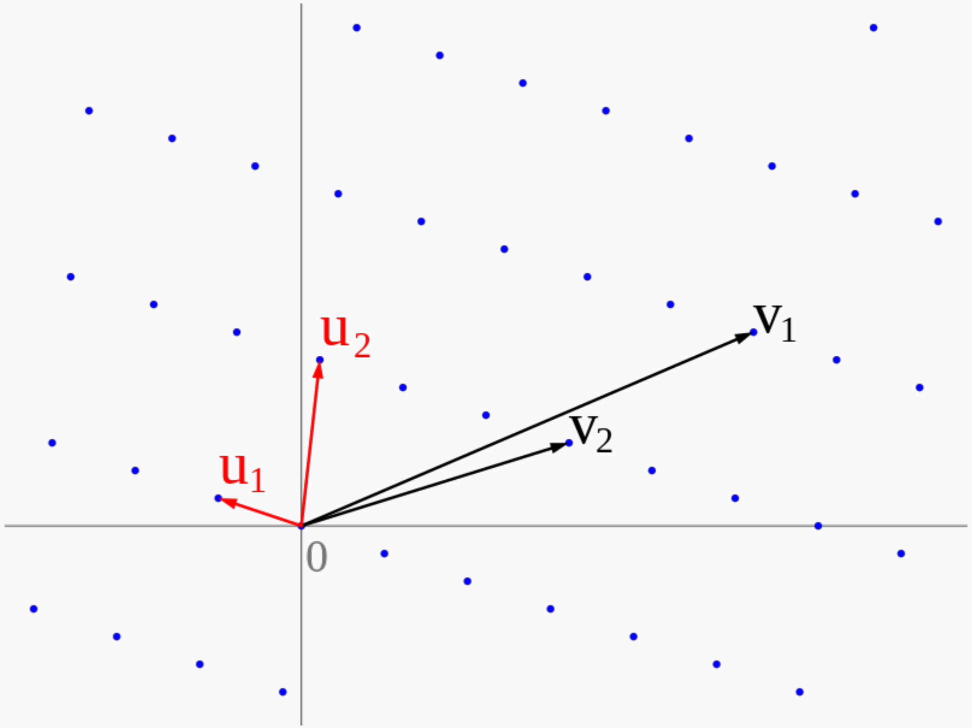
Matematisk definition





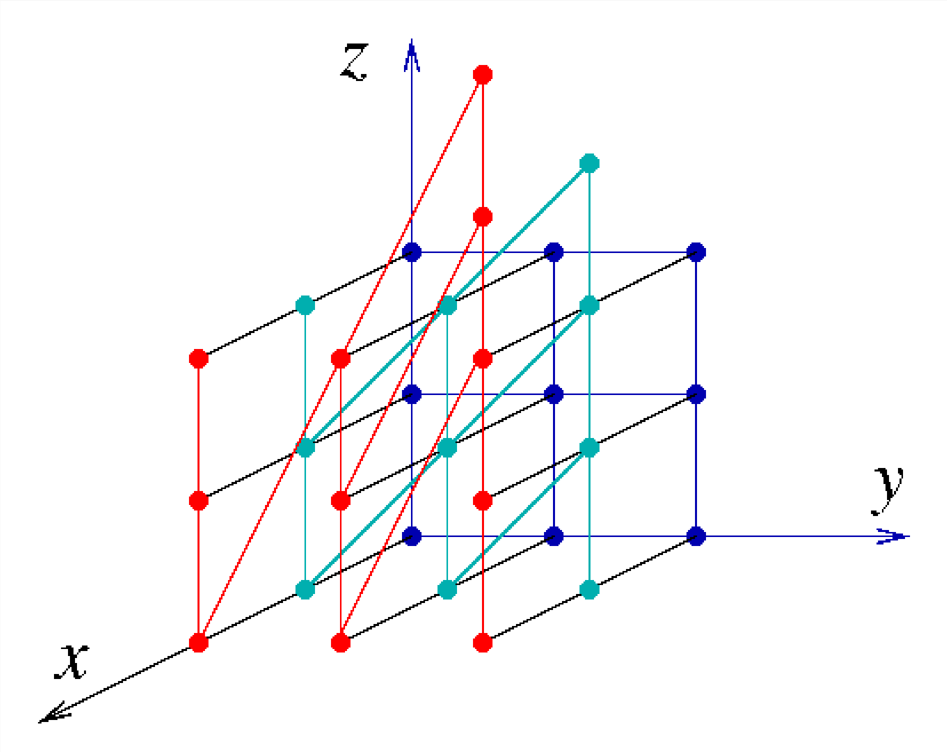
Matematisk definition

- 2 dimensioner
- $v = (x, y)$



Matematisk definition

- 3 dimensioner
- $v = (x, y, z)$



Matematisk definition

- n dimensioner
- Hittade ingen bra bild :(

$$v = (v_1, v_2, ..., v_n)$$

Gitter

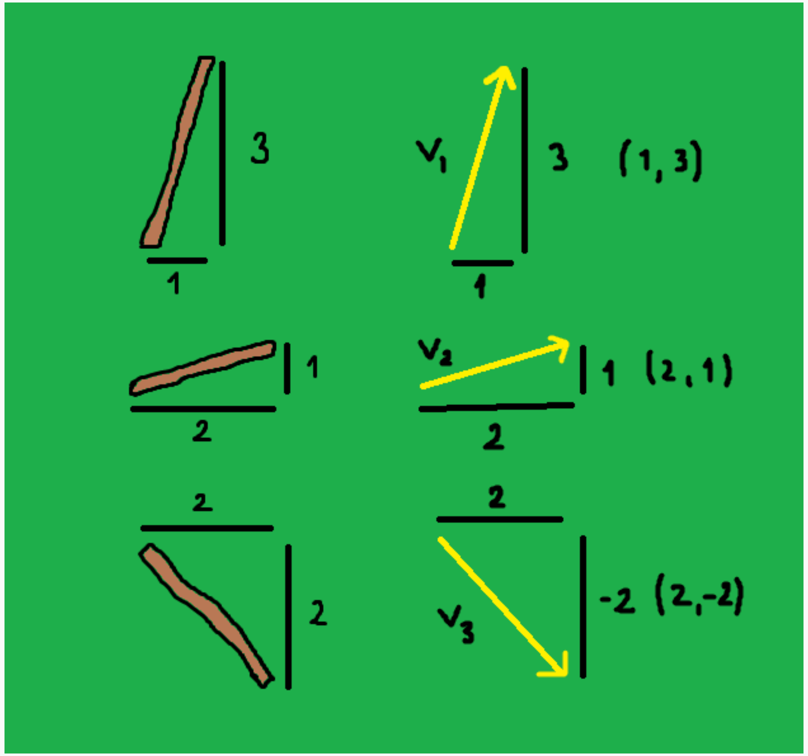
Gitter

- Bas - första pinnarna
- Basmatris
- Gittret är mängden möjliga vektorer/pinnar

$$M = \begin{pmatrix} -v_1- \\ -v_2- \\ -v_3- \\ \vdots \\ -v_n- \end{pmatrix}$$

$$L = \left\{ \sum_{i=1}^{n} a_i v_i : a_i \in \mathbb{Z} \right\}$$

Gitter



$$v_1 = (1,3), v_2 = (2,1), v_3 = (2,-2)$$

$$M = \begin{pmatrix} - v_1 - \\ - v_2 - \\ - v_3 - \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 2 & 1 \\ 2 & -2 \end{pmatrix}$$

$$L = \{a_1 v_1 + a_2 v_2 : a_1, a_2 \in \mathbb{Z}\}$$

Gitter

$$\begin{aligned} v &= 2v_1 + 3v_2 - 5v_3 \\ &= 2(1,3) + 3(2,1) - 5(2,-2) \\ &= (2,6) + (6,3) + (-10,10) \\ &= (-2,19) \end{aligned}$$

$$\begin{pmatrix} -v_1- \\ -v_2- \\ -v_3- \end{pmatrix} \begin{matrix} +2 \\ +3 \\ -5 \end{matrix} \qquad \begin{pmatrix} -2v_1- \\ -3v_2- \\ -(-5)v_3- \end{pmatrix}$$

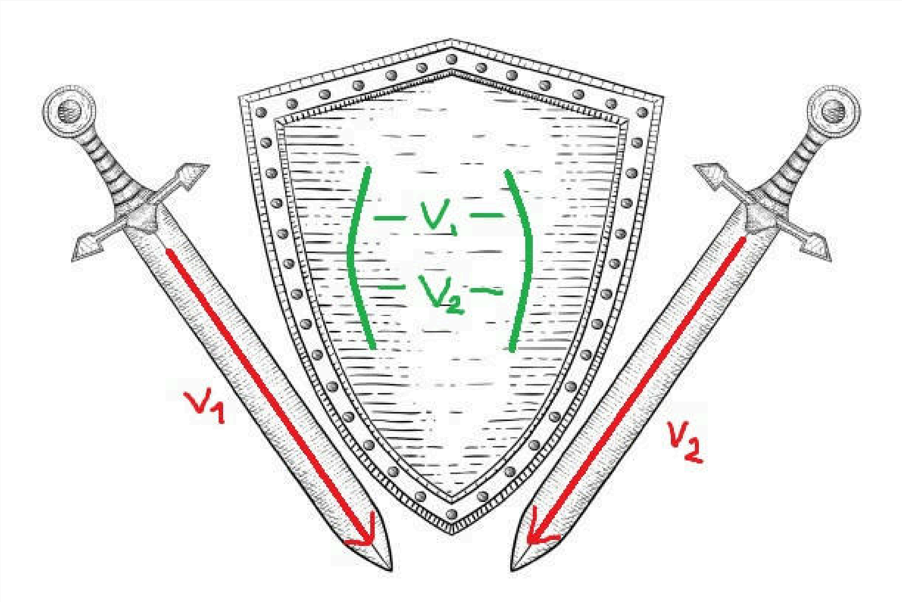
$$\begin{pmatrix} 1 & 3 \\ 2 & 1 \\ 2 & -2 \end{pmatrix} \begin{matrix} +2 \\ +3 \\ -5 \end{matrix} \qquad \begin{pmatrix} 2 & 6 \\ 6 & 3 \\ -10 & 10 \end{pmatrix}$$

$$(-2, 19)$$

Varför gitter?

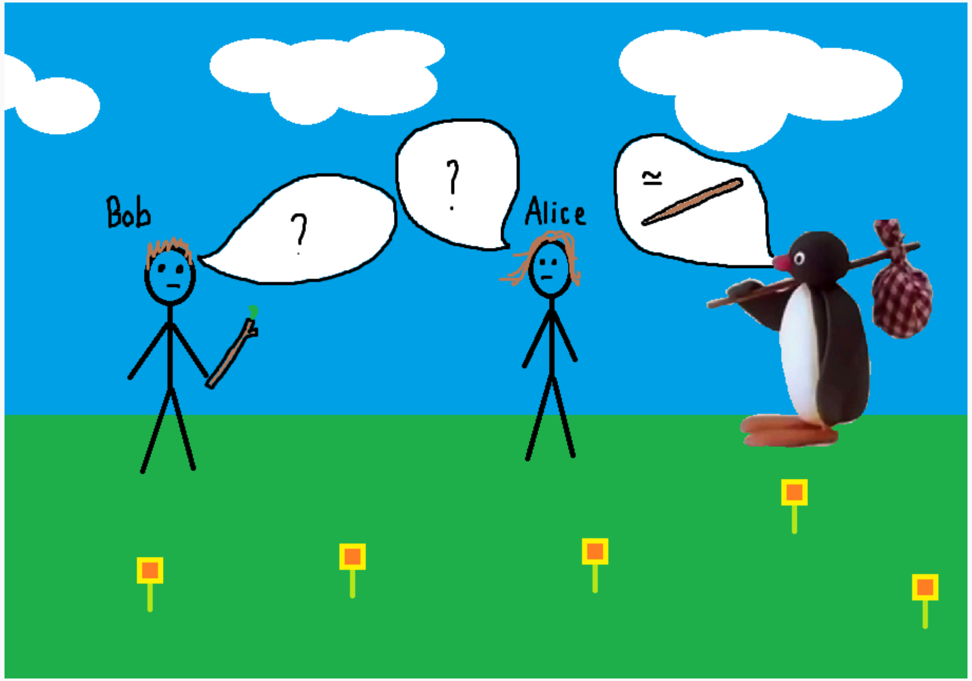
Varför gitter?

- Kryptografins svärd och sköld
- Svärd - knäcker svåra problem
- Sköld - skapar svåra problem



Gitter som sköld

- Många svåra problem i gitter
- Shortest Vector Problem “SVP”
- Closest Vector Problem “CVP”
- LWE, Ring-LWE
- Post-quantum



Gitter som svärd

- Varför skulle gitter hjälpa?
  - Mycket är heltalskombinationer
  - Ofta är saker “litet” relativt
- LLL - Lenstra–Lenstra–Lovász
- Dålig RNG (Minecraft) -> LLL
- Dålig RSA nyckel -> LLL
- Läcker liter bitar -> LLL

RSA:

$$\varphi(pq) = pq - p - q + 1$$

p, q — 1024 bits

$pq, \varphi(pq)$ ~ 2028 bits

LLL - L^3

- Reducerar din bas
- Hittar “korta” vektorer
- Reduceringar till en viss gräns
- Blackbox tool
- SageMath

```
from sage.all import *

M = Matrix(...) # din bas
M_reduced = M.LLL()

for v in M_reduced:
    print(v) # "korta" vektorer
```

Hur använder vi svärdet?

Hur använder vi svärdet?

- Konstruera en bas medvetet
- Reducera basen med LLL
- Leta bland små vektorer

```
from sage.all import *

M = Matrix(...) # din bas
M_reduced = M.LLL()

for v in M_reduced:
    print(v) # "korta" vektorer
```

Hur använder vi svärdet?

- Hur konstruerar vi basen?
- Beror på situation
- Många tricks
- Lös problem och bygg intuition
- Jag visar några exempel

```
from sage.all import *

M = Matrix(...) # din bas
M_reduced = M.LLL()

for v in M_reduced:
    print(v) # "korta" vektorer
```

Exempel

Generell form

$$y = a_1x_1 + a_2x_2 + ... + a_nx_n$$

- Leta efter liknande uttryck
- $a_1, ..., a_n$ är "små" jämfört med $x_1, ..., x_n, y$
- $x_1, ..., x_n, y$ är kända, medan $a_1, ..., a_n$ sökes
- Genom konstruktion av bas kan LLL hitta sådana $a_1, ..., a_n$

Generell form

$$M = \begin{pmatrix} -x_1 & 1 & 0 & \dots & 0 \\ -x_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -x_n & 0 & 0 & \dots & 1 \\ y & 0 & 0 & \dots & 0 \end{pmatrix} \quad \begin{pmatrix} -x_1 & 1 & 0 & \dots & 0 \\ -x_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -x_n & 0 & 0 & \dots & 1 \\ y & 0 & 0 & \dots & 0 \end{pmatrix} \begin{matrix} +a_1 \\ +a_2 \\ \vdots \\ +a_n \\ 1 \end{matrix} \to \begin{pmatrix} -a_1x_1 & a_1 & 0 & \dots & 0 \\ -a_2x_2 & 0 & a_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_nx_n & 0 & 0 & \dots & a_n \\ y & 0 & 0 & \dots & 0 \end{pmatrix}$$

- $0 = 1y - a_1x_1 - a_2x_2 - \dots - a_nx_n$
- Vektorerna i basen är stora ty $x_1, \dots, x_n, y$ är det
- Gittret som bildas av M innehåller vektorn $v = (0, a_1, \dots, a_n)$
- Eftersom v är liten kan den finnas i reduceringen av M

Generell form

$$M = \begin{pmatrix} -x_1 & 1 & 0 & \dots & 0 \\ -x_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -x_n & 0 & 0 & \dots & 1 \\ y & 0 & 0 & \dots & 0 \end{pmatrix}$$

- Börja simpelt
- Dummy-values och printa
- Kolla antalet bitar i `M_reduced`

```
from sage.all import *

x = [...] # x-values
y = ...   # y-value
n = len(x)
M = Matrix(n + 1, n + 1)
M.set_block(0, 1, identity_matrix(n))
for i in range(n):
    M[i, 0] = -x[i]

M[n, 0] = y

M_reduced = M.LLL()
for v in M_reduced:
    if v[0] == 0:
        print(v) # pray
```

Modulo

$$y \equiv a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \pmod{p}$$

$$\Leftrightarrow y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n - kp, \quad k \in \mathbb{Z}$$

- Exakt samma uttryck fast med modulo
- Samma villkor
  - $a_1, \ldots, a_n$ är "små" jämfört med $x_1, \ldots, x_n, y$
  - $x_1, \ldots, x_n, y$ är kända, medan $a_1, \ldots, a_n$ sökes
- p är hyfsat stort
- k är i storleksordning $\left\lfloor \frac{y}{p} \right\rfloor$ vilket ofta är litet

Modulo

$$M = \begin{pmatrix} -x_1 & 1 & 0 & \dots & 0 & 0 \\ -x_2 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -x_n & 0 & 0 & \dots & 1 & 0 \\ y & 0 & 0 & \dots & 0 & 1 \\ p & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{matrix} +a_1 \\ +a_2 \\ \vdots \\ +a_n \\ 1 \\ k \end{matrix} \quad \to \quad \begin{pmatrix} -a_1 x_1 & a_1 & 0 & \dots & 0 & 0 \\ -a_2 x_2 & 0 & a_2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -a_n x_n & 0 & 0 & \dots & a_n & 0 \\ y & 0 & 0 & \dots & 0 & 1 \\ kp & 0 & 0 & \dots & 0 & 0 \end{pmatrix}$$

- $0 = 1y - a_1 x_1 - a_2 x_2 - \dots - a_n x_n + kp$
- Innehåller liten vektor $(0, a_1, \dots, a_n, 1)$

Modulo

$$M = \begin{pmatrix} -x_1 & 1 & 0 & \dots & 0 & 0 \\ -x_2 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -x_n & 0 & 0 & \dots & 1 & 0 \\ y & 0 & 0 & \dots & 0 & 0 \\ p & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{array}{c} +? \\ +? \\ \vdots \\ +? \\ p \\ -y \end{array} \quad \rightarrow \quad \begin{pmatrix} -?\,x_1 & ? & 0 & \dots & 0 & 0 \\ -?\,x_2 & 0 & ? & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -?\,x_n & 0 & 0 & \dots & ? & 0 \\ yp & 0 & 0 & \dots & 0 & 0 \\ -yp & 0 & 0 & \dots & 0 & 0 \end{pmatrix}$$

- Tvinga LLL till att göra det vi vill
  I.e stoppa den från att göra det vi inte vill

Knapsack

- Alice har en säck med värden $x_1, \ldots, x_n$ som är “stora”
- Bob har valt en delmängd av dessa, 1 eller 0 styck av varje
- Du vet totala värdet y av Bobs val men inte vilka värden
- Kan skrivas om till vårt uttryck

$$y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n$$

$$a_1, \ldots, a_n \in \{0, 1\}$$

$$\begin{pmatrix} -x_1 & 1 & 0 & \ldots & 0 \\ -x_2 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -x_n & 0 & 0 & \ldots & 1 \\ y & 0 & 0 & \ldots & 0 \end{pmatrix}$$

Centrering

- LLL kommer främja a_i som 0
- Vektorn $(2, 0, 0, \ldots, 0)$ är ju kortare än målvektorn $(0, 1, 0, 1, 1, 0, 1, 1)$
- Centrera värdena kring $-\frac{1}{2}$
- $a_i = 0$ ger $-\frac{1}{2}$ och $a_i = 1$ ger $\frac{1}{2}$ som är lika “stora”

$$\begin{pmatrix} -x_1 & 1 & 0 & \ldots & 0 \\ -x_2 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -x_n & 0 & 0 & \ldots & 1 \\ y & -\frac{1}{2} & -\frac{1}{2} & \ldots & -\frac{1}{2} \end{pmatrix}$$

Centrering

- Använd bara heltalsvärden i din bas
- Centrera kring -1 istället

$$\begin{pmatrix} -x_1 & 2 & 0 & \dots & 0 \\ -x_2 & 0 & 2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -x_n & 0 & 0 & \dots & 2 \\ y & -1 & -1 & \dots & -1 \end{pmatrix}$$

$$(0, 2a_1 - 1, 2a_2 - 1, \dots, 2a_n - 1)$$

```
from sage.all import *

x = [...] # x-values
y = ...   # y-value
n = len(x)
M = Matrix(n + 1, n + 1)
M.set_block(0, 1, 2 * identity_matrix(n))
for i in range(n):
    M[i, 0] = -x[i]
    M[n, i + 1] = -1 # centrering

M[n, 0] = y

M_reduced = M.LLL()
for v in M_reduced:
    if v[0] == 0:
        print(v) # don't forget +1 then / 2
```

Subset Sum

- Generalisering av knapsack
- Bob kan nu välja flera av varje värde i Alices säck
- Kan fortfarande skrivas om till vårt uttryck med centrering
- Svårare att tvinga LLL
- SVP vs CVP

$$y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n$$

$$a_1, \ldots, a_n \in [0, l]$$

$$\begin{pmatrix} -x_1 & 2 & 0 & \ldots & 0 \\ -x_2 & 0 & 2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -x_n & 0 & 0 & \ldots & 2 \\ y & -l & -l & \ldots & -l \end{pmatrix}$$

CVP

- Closest vector problem
- Gör om CVP till approximativ SVP
- Blackbox tool
- Blupper linineq solver
- $y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n$
  $a_1, \ldots, a_n \in [0, l]$

```
from sage.all import *

x = [...] # x-values
y = ...   # y-value
n = len(x)
M = Matrix(n, n+1)
M.set_block(0, 1,  identity_matrix(n))
for i in range(n):
    M[i, 0] = x[i]

M_reduced = M.LLL()
target = [y] + [l//2 for _ in range(n)]
v = cvp(M_reduced, target)

assert v[0] == y
```

Viktning

- Vad händer om det redan finns litet $y - \sum_{i=1}^{n} a_i x_i$ som är icke-noll
- Bestraffa med viktning
- Stor vikt W
- W gånger icke-noll blir stort
- W gånger noll blir noll
- Första kolumnen gånger W

$$\begin{pmatrix} -x_1 & 1 & 0 & \dots & 0 \\ -x_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -x_n & 0 & 0 & \dots & 1 \\ y & 0 & 0 & \dots & 0 \end{pmatrix}$$

$$\begin{pmatrix} -Wx_1 & 1 & 0 & \dots & 0 \\ -Wx_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -Wx_n & 0 & 0 & \dots & 1 \\ Wy & 0 & 0 & \dots & 0 \end{pmatrix}$$

Viktning

- Testa olika vikter
- Finns en balans
- Vikten kan brute:as

```
from sage.all import *

W = 2**256
x = [...] # x-values
y = ...   # y-value
n = len(x)
M = Matrix(n + 1, n + 1)
M.set_block(0, 1, identity_matrix(n))
for i in range(n):
    M[i, 0] = - W * x[i]

M[n, 0] = W * y

M_reduced = M.LLL()
for v in M_reduced:
    if v[0] == 0:
        print(v)
```

Viktning

$$\begin{pmatrix} -Wx_1 & 2W & 0 & \dots & 0 \\ -Wx_2 & 0 & 2W & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -Wx_n & 0 & 0 & \dots & 2W \\ Wy & -1W & -1W & \dots & -15W \end{pmatrix}$$

- Var kreativ!
- Centreringen kan t.ex viktas

```
from sage.all import *

W = 2**256
x = [...] # x-values
y = ...   # y-value
n = len(x)
M = Matrix(n + 1, n + 1)
M.set_block(0, 1, identity_matrix(n))
for i in range(n):
    M[i, 0] = - W * x[i]

M[n, 0] = W * y

M_reduced = M.LLL()
for v in M_reduced:
    if v[0] == 0:
        print(v)
```

Lösa challs