

1. Una hormiga artificial vive en un mundo bidimensional cuadriculado y desarrolla un comportamiento que le permite seguir un rastro de feromonas a lo largo de un conjunto de casillas previamente marcadas (el tamaño del rastro es de una casilla). La hormiga ocupa una sola casilla y puede encarar las casillas que se encuentran arriba, a la derecha, a la izquierda y debajo de la posición en la que se encuentra. La hormiga puede llevar a cabo tres acciones: moverse a una celda hacia adelante (actFORWARD), girar a la izquierda permaneciendo en la misma casilla (actTURN_L) y girar a la derecha permaneciendo en la misma casilla (actTURN_R). La hormiga puede percibir si la casilla que tiene delante (en el sentido del movimiento) tiene feromona.

- a) Especificar un sistema de reglas para controlar el comportamiento de la hormiga en el seguimiento del rastro de la feromona. Suponer inicialmente a la hormiga en una casilla en la que puede percibir el rastro de feromona.
- b) Resuelva el problema anterior con la siguiente restricción: la hormiga, una vez que llega a una nueva casilla, no puede girar más de 180 grados en el mismo sentido de giro.

c) Modifique el conocimiento necesario para que la hormiga pase por todas las casillas con feromona del mapa feromonaz.map

d) Proponga un mapa en donde el comportamiento definido para el apartado anterior no consiga pasar por todas las casillas con feromona.

1. DATOS

- * hormiga artificial en un mundo bidimensional
- * hormiga desarrolla comportamiento → le permite seguir rastro feromonas
- + hormiga ocupa 1 casilla
- hormiga → arriba, abajo, izquierda, derecha (casilla donde puede encarar)
- * actFORWARD, actTURN-L, actTURN-R
- * hormiga puede percibir si la casilla debante → tiene feromona.

a) Sistema de Reglas.

Ahora la hormiga se
puede quedar girando
en círculos cuando no
hay

Agent :: ActType think () {

if (FEROMONA-) {- // Si hay feromona, nos movemos
accion: actFORWARD; hacia ella

else

accion : actTURN-R. // Si no giramos

return accion

}

b) Corregir el comportamiento \rightarrow no puede girar + 180° mismo sentido giro.

c)

- Añadimos variable de giro \rightarrow int giro

.h

```
private : bool FEROMONA_ ;  
int giro ;
```

public :

Agent () {

FEROMONA_ = false ;

giro = 0 ;

}

\rightarrow no puede girar + 180° mismo sentido giro.

: CPP

```
Agent::ActType Agent::Think(){  
    int accion = 0 ;
```

if (FEROMONA_) {

accion = act FORWARD ; \rightarrow

giro = 0 ; // va a una

{ else if (giro != 2) { // No 180°

giro ++ ;

accion = act TURN-R ;

}

0: no he girado,

puedo girar

1. 90°

2. 180°

nueva
casilla

no he
girado

c) Comportamiento para que pase por todas las casillas

agent :: AgentType Agent :: think () {

int accion = 0;

if (FEROMONA_ && giro != 2) {

accion = actFORWARD;
giro = 0;

g.
} else {

giro++;
accion = actTURN-R;

g.
return accion

!!

// Si he girado l veces @
// porque ya he pasado
// por ahí hace poco

l,
HE UN SALTO Y
GIRE CTRA VEZ

d) Propón un mapa en donde el comportamiento definido para el apartado anterior no consiga pasar por toda las casillas con feromonas.

Igual que el mapa 2 solo que metemos un pequeño camino de por medio \rightarrow no lo reconoce

F F F F F
F F F F F
F F F F F
F F F F F
F F F F F

F \rightarrow no lo reconoce.

2. La avispa hembra del género *Sphex*, deja sus huevos dentro de un grillo que ha paralizado y ha llevado a su nido. Las larvas de la avispa salen del grillo y se alimentan de él. La avispa presenta el siguiente comportamiento: lleva el grillo paralizado a su nido, lo deja en el umbral del nido, entrar dentro del nido para ver si todo está correcto, sale, y entonces arrastra al grillo hacia su interior. Si el grillo se mueve cuando la avispa está en el interior haciendo la inspección preliminar, la avispa saldrá del nido, volverá a colocar el grillo en el umbral, pero no dentro, y repetirá el procedimiento de entrar en el nido para ver si todo está correcto. Si el grillo se mueve otra vez mientras la avispa está dentro del nido, ésta volverá a salir y colocar el grillo en el umbral, entrando de nuevo en el nido para realizar la inspección preliminar. En una ocasión, este procedimiento se repitió cuarenta veces. Define características y acciones para diseñar un agente reactivo que se corresponda con el comportamiento de la avispa.

1. DATOS

- * avispa Sphex -> deja sus huevos dentro de un grillo que ha paralizado y llevado a su nido. -> larvas salen y se alimentan del grillo.
- * Comportamiento avispa -
 - lleva grillo paralizado nido
 - lo deja en el umbral del nido
 - entra dentro del nido -> comprobar
 - arrastra grillo -> interior.
 - grillo se mueve cuando esta la avispa en el interior -> avispa sale del nido -> sale y coloca el grillo en el umbral
 - > 40 veces (se repitió en 1 ocasión)
- + Define características y acciones para diseñar agente reactivo que corresponda con el comportamiento avispa.

Objetivo -> Coger al grill + acciones + dejarlo en el nido

Percepciones ->

- sensor 1 -> la avispa tiene el grill o no
- sensor 2 -> la avispa esta dentro/fuera del umbral
- sensor 3 -> el grill esta o no en el humbral
- sensor 4 -> si el nido -> correcto o no

memoria -> recuerda si el nido esta / o no en lo correcto

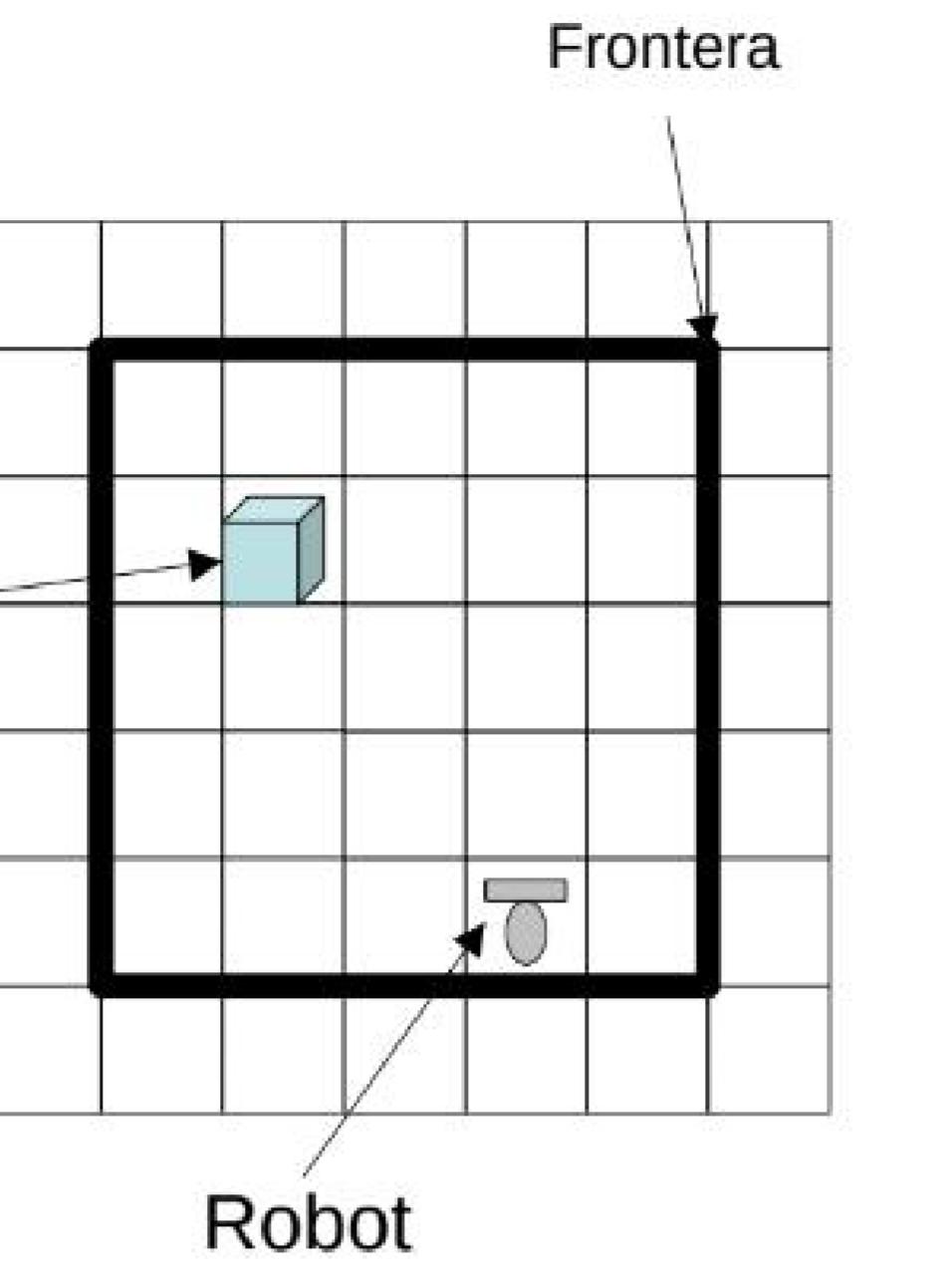
acciones ->

- arrastar -> coger al grill
- dejar -> dejar al grill
- va-umbral -> va al umbral
- va-dentro -> va dentro nido
- va-fuera -> sale nido
- encuentra -> encuentra grill

3. Supongamos que un agente trabaja sobre un tablero formado por NxN casillas. Sobre este tablero se definen dos zonas: una "zona interior" formada por un tablero de $(N-2) \times (N-2)$ casillas inscrito en el tablero general, y una "zona exterior" formada por el resto de las casillas. Separando ambas zona aparece una línea gruesa negra denominada "*Frontera*". En la figura se muestra un ejemplo de la configuración de un tablero 7x7.

El cometido del robot consiste en llevar todos los obstáculos que se encuentren en la zona interior a la zona exterior. El robot siempre se debe encontrar en la zona interior, y no debe nunca traspasar la frontera.

Para realizar esta tarea, el robot dispone de 3 sensores, un sensor de choque "**BUMPER**" que le permite detectar el obstáculo, un sensor de infrarrojos "**CNY70**" que permite ver dónde está la línea de la Frontera, y una brújula digital "**Brújula**" que le indica su orientación en el avance. Los dos primeros sensores se encuentran situados en la parte frontal del robot. La brújula sólo devuelve 4 valores: 0, 1, 2 y 3, representando respectivamente Norte, Este, Sur y Oeste.



Las acciones que puede realizar el robot son las siguientes:

Avanzar: Avanza una casilla en la dirección que marca su brújula siempre que no tenga un obstáculo delante.

Retroceder: Retrocede una casilla en la dirección contraria a la que indica su brújula, siempre que no tenga un obstáculo detrás.

GirarI: Gira sin moverse de la casilla hacia la izquierda.

GirarD: Gira sin moverse de la casilla hacia la derecha.

Nada: No realiza ninguna acción

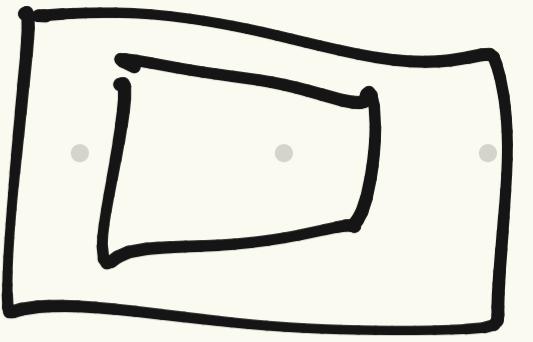
Empujar: Avanza una casilla en la dirección que marca su brújula. Para que esta acción tenga efecto, debe estar activado el sensor de choque.

Se pide:

- Definir las variables de estado (**nombre e descripción**) y las reglas de producción necesarias para diseñar un agente reactivo con memoria que partiendo de una casilla desconocida dentro de la zona interior de un tablero de dimensiones también desconocidas (nunca superiores a 99x99), sea capaz de calcular la dimensión de la zona interior, suponiendo que en el tablero no hay obstáculos.
- Definir las variables de estado y las reglas de producción necesarias que permitan al robot localizar el obstáculo en el tablero.
- Suponiendo que el robot se encuentra orientado hacia el obstáculo en una casilla adyacente (es decir, el sensor BUMPER está activado) y que el obstáculo se encuentra en una casilla interna del tablero que no es adyacente con ninguna casilla pegada a la frontera, definir las variables de estado y las reglas de producción necesarias que permitan al robot expulsar el obstáculo hacia la zona exterior, arrastrándolo por el camino más corto de casillas.

DATOS

- > agente -> tablero $N \times N$ casillas
 - > 2 zonas
 - > interior $(N-2) \times (N-2)$
 - > exterior
- > ambas zonas separadas -> frontera
 - > ej 7×7
- > objetivo: robot -> llevar todos los obstáculos que se encuentren ->
 - > zona interior -> exterior
- > Robot siempre debe encontrarse en la zona interior



3 sensores

- > choque "Bumper"
- > barra "CNY70"
- > brújula "brújula"
- > detecta distancia
- > donde está línea frontal
- > orientación {0, 1, 2, 3.}

{ parte frontal

Acciones

- > FORWARD
- > BACKWARD
- > TURN-L
- > TURN-R
- > IDLE
- > PUS

- a) Define las variable estado (nombre e descripción) y las reglas de producción necesaria → diseñar agente reactivo con memoria
- * parte de una casilla desconocida dentro tabler.
 - * calcule la dimensión zona interior → suponiendo tabler no hay obstáculos.

variable estado → int brújula → 0, 1, 2, 3, = 0 "Norte"
 → char mapa [i][j] [7][7] mapa → del entorno.
 → posición → int fil, col. ← si hay objeto y no esta fuera.

Reglas de Producción
 $\rightarrow \text{BUMPER} \wedge \text{CNY70} \rightarrow \{\text{Push, actualizar}(x,y)\}$.

Si estoy orientado mirando a la casilla que he estado menos n veces, avante, si no miro a donde estoy mirando hago un giro hasta que este orientado a esa casilla

si a CNY70 está activado → casilla a -1.

- b) \rightarrow Para localizar el obstáculo en el tablero \rightarrow BUMPER = true;
- c) BUMPER = true \rightarrow acción = actPUSH
actualizamos mapa (x, y) ;
- * Si estoy orientado mirando a la casilla que he estado menos número de veces, avante, si no miro a donde estoy mirando, hago un giro hasta que este orientado a esa casilla,
si el CNV70 esta activado esa casilla la pongo $\rightarrow -1$;
 - * Contador casillas (que cuente para cada casilla las veces que se ha visitado)
y una priority que ordene los nodos según ese atributo
¿Cómo hacer para que el robot se mueva hacia la casilla estando menor número de veces.

6.

Agent() {

 brujula = 0;

 pasos = 0;

7.

char mapa[7][7];

int brujula

int fil, pos;

int pasos

.CPP

if (CNY7Q == true) {

 accion = actTURN-L;

if (pasos == 3 || CNY7Q == false) {

 pasos = 0

 accion = actTURN-L

8.

if (accion == actFORWARD)

 pasos += 1;

if (BUMPER_).

 accion = actPUSH;

4. Supongamos que tenemos un robot sobre un mapa bidimensional discreto de tamaño NxM. El robot puede realizar las acciones de **Avanzar** y **Girar** en el sentido de las agujas del reloj. El robot posee un sistema de posicionamiento sobre el mapa que le devuelve sus coordenadas absolutas “**(robotX, robotY)**” dentro del mapa.

Suponiendo que en el mapa hay obstáculos fijos (paredes), y que el robot se encuentra ubicado dentro de ese mapa en una posición concreta, definir un comportamiento reactivo para el mismo que le permita desplazarse hasta una coordenada objetivo “**(ObjX, ObjY)**”. Para ello, definir las variables de estado necesarias y el sistema de reglas de producción que reproducen el comportamiento requerido.

DATOS

- robot mapa bidimensional ($N \times M$).
- acciones Avanzar y Girar → sentido aguja reloj.
- Robot → sistema de posicionamiento sobre el mapa
 - devuelve las coordenadas absolutas (robot X, robot Y) dentro mapa.
- mapa → tiene puerta.
- robot → empieza en el centro mapa en una pos concreta.
- Define comportamiento reactivo para el mismo que le permita desplazarse → objetivo (obj X, obj Y).

Objetivo \rightarrow desplazarse hasta una coordenada objetivo (obj_x , obj_y).
Perecer: $0 \leq v$ si hay obstáculo, \neq caso contrario. ¿Cómo rebajar mapa-máscara?

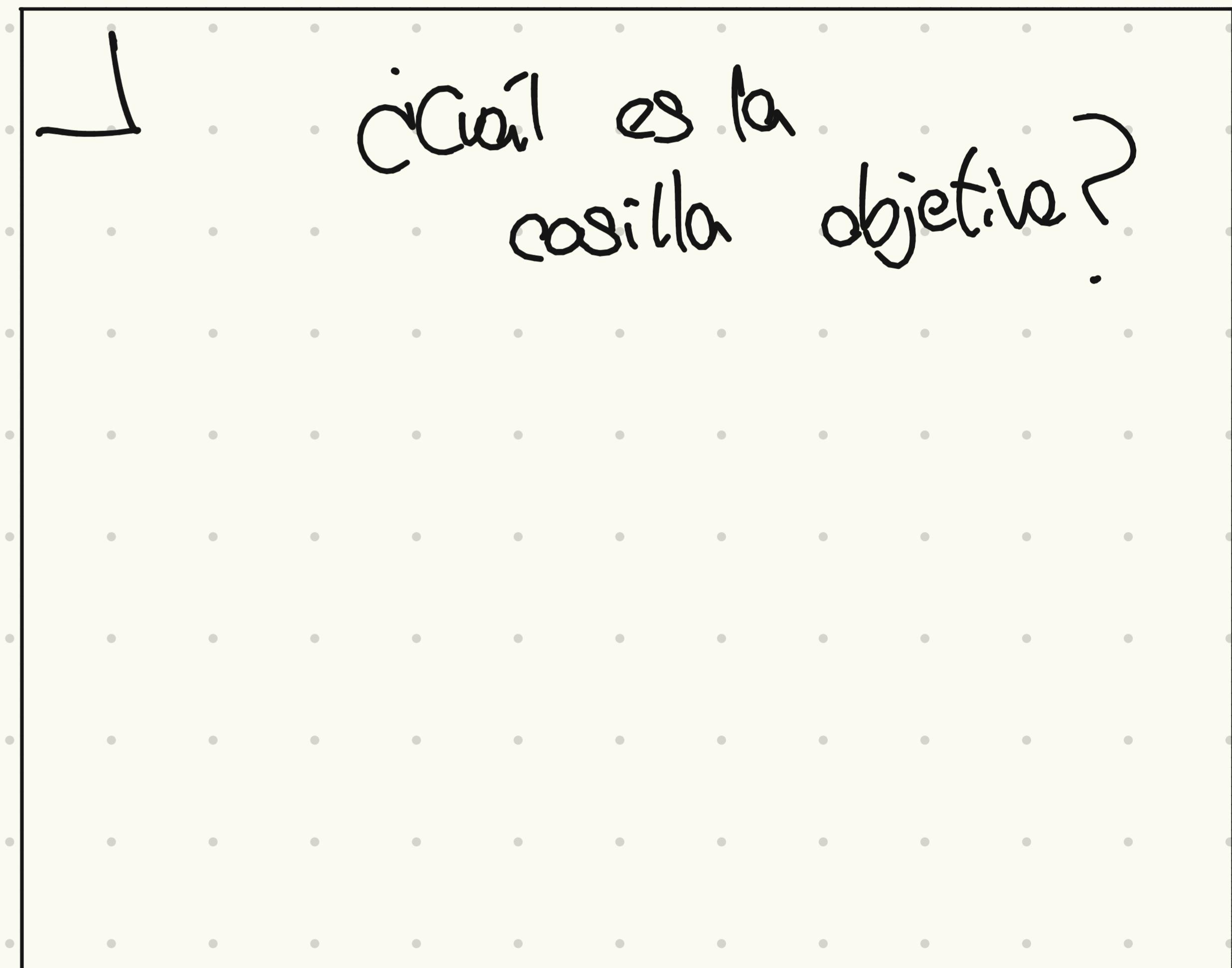
Acciones

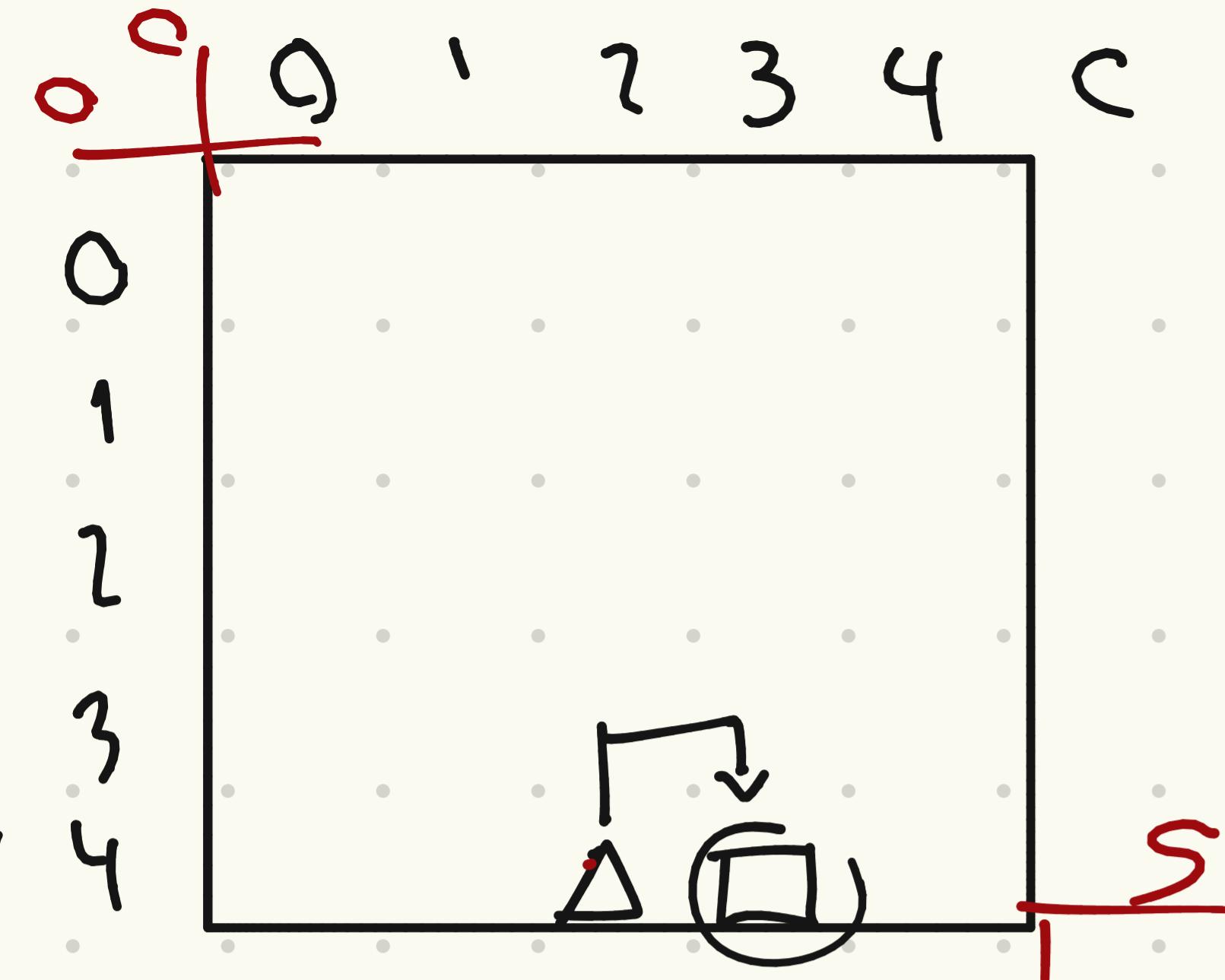
- AV \rightarrow avanzar
- D \rightarrow derecha
- IDLE \rightarrow nada

memoria: no hay

Sistema Producción

- O \rightarrow TURN
- ¬O \rightarrow Av.





1. c. brújula $\begin{matrix} \text{S} \\ \text{S} \end{matrix}$

2 i (origen)

3 (destino) \times

¿Cómo va desde la origen \rightarrow destino?

3-4 \rightarrow 1,4 Como diría que

¿empuje hacia abajo?

IS-41

IC-41

IS-31

IC-31

\times expulsa por la casilla + corta

\rightarrow apartado anterior

recorrido

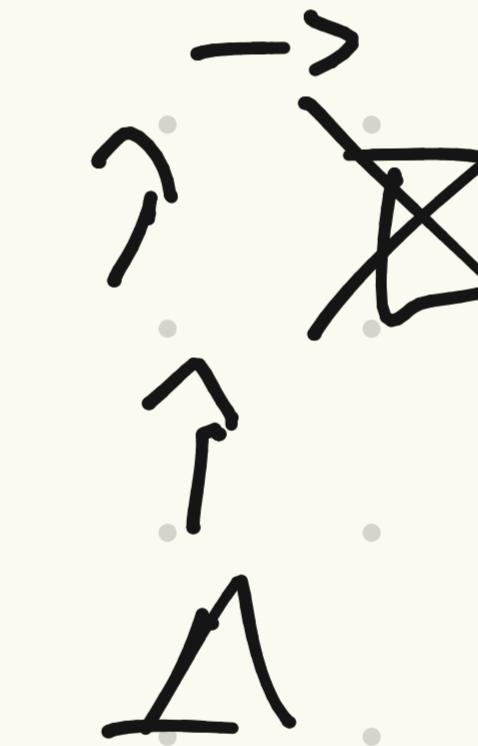


\approx
hasta
encuentre
treute

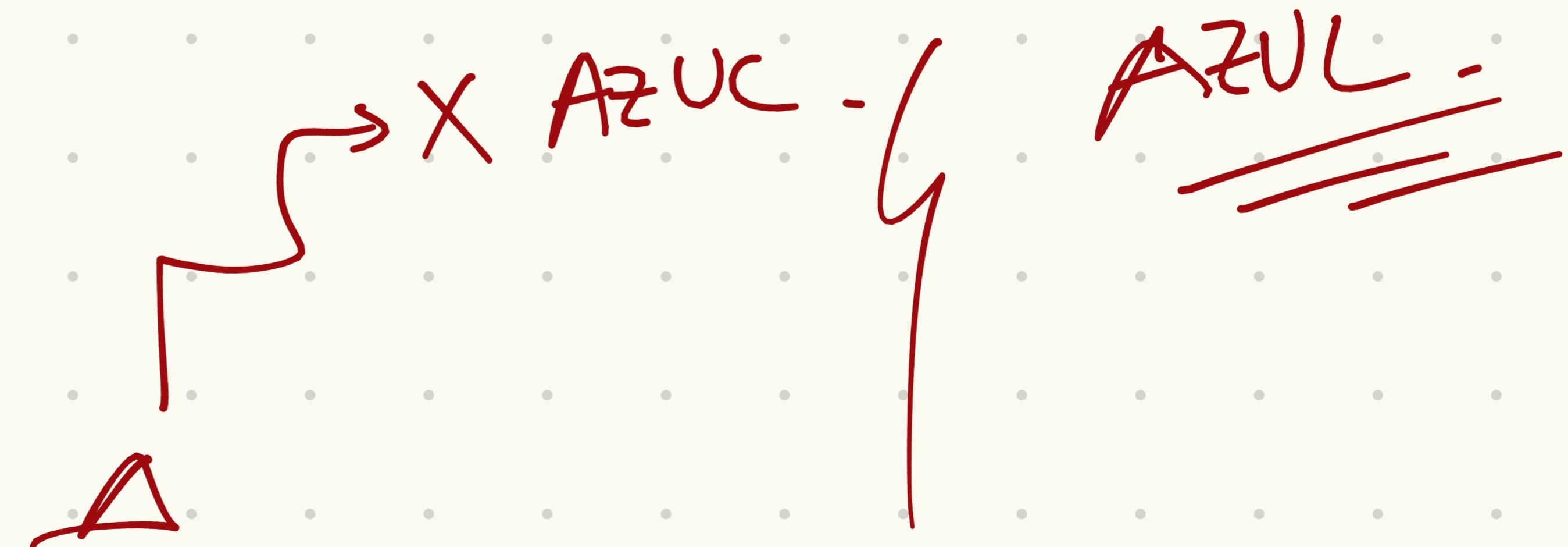
1º Colisión \rightarrow ve donde está el bloque

Avanza hasta encontrar el rango. (sensores)

2º obtenga destino y origen (pes actual).



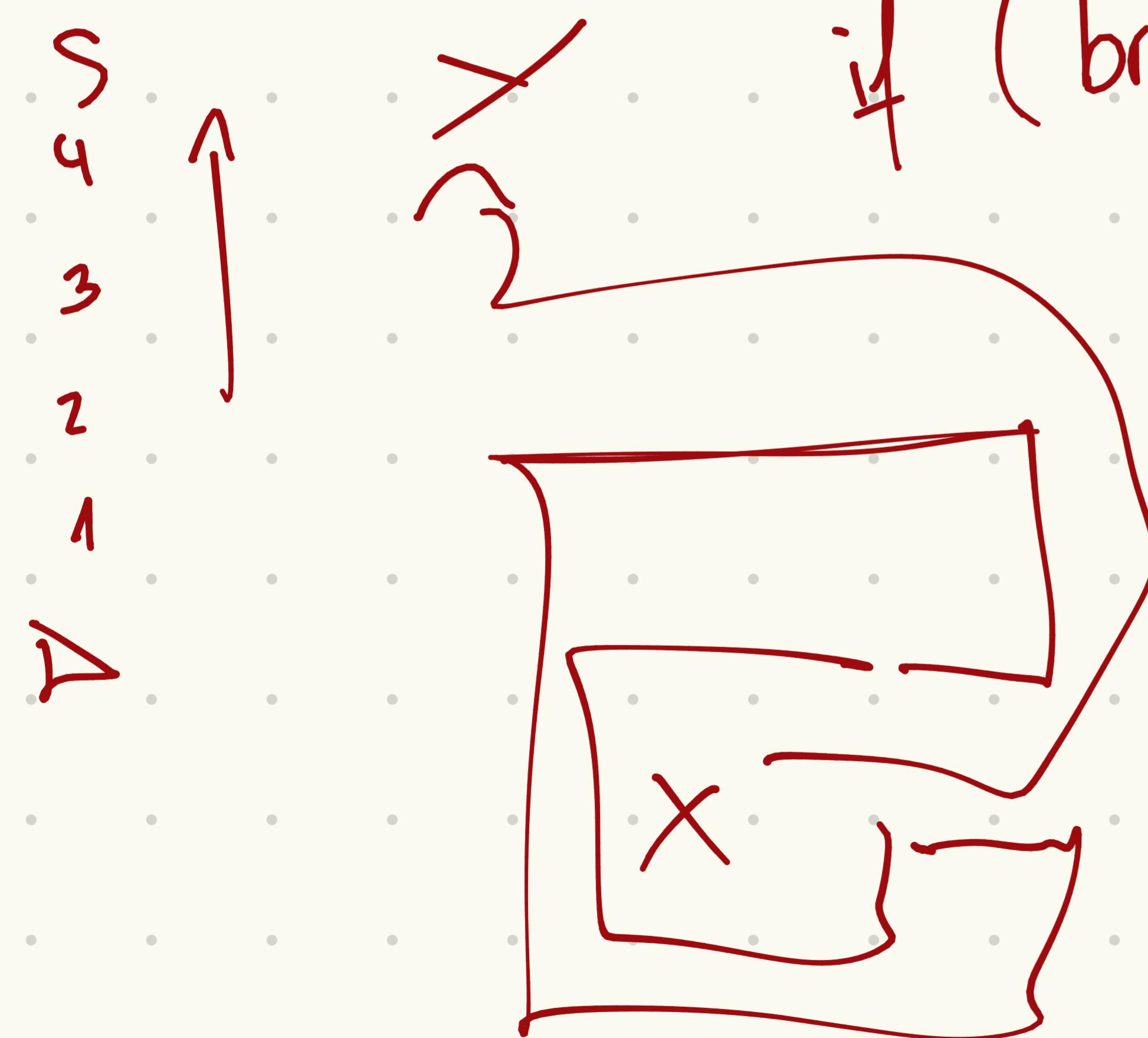
TUTORIAL.



→ Nos queremos mover 5 filas. fila -> arriba, abajo

¿Cómo me muevo para arriba → 1º oriento. → TIENE QUE ESTAR EN EL NORTE.

5 para arriba 1º oriento



if (brujula != norte).

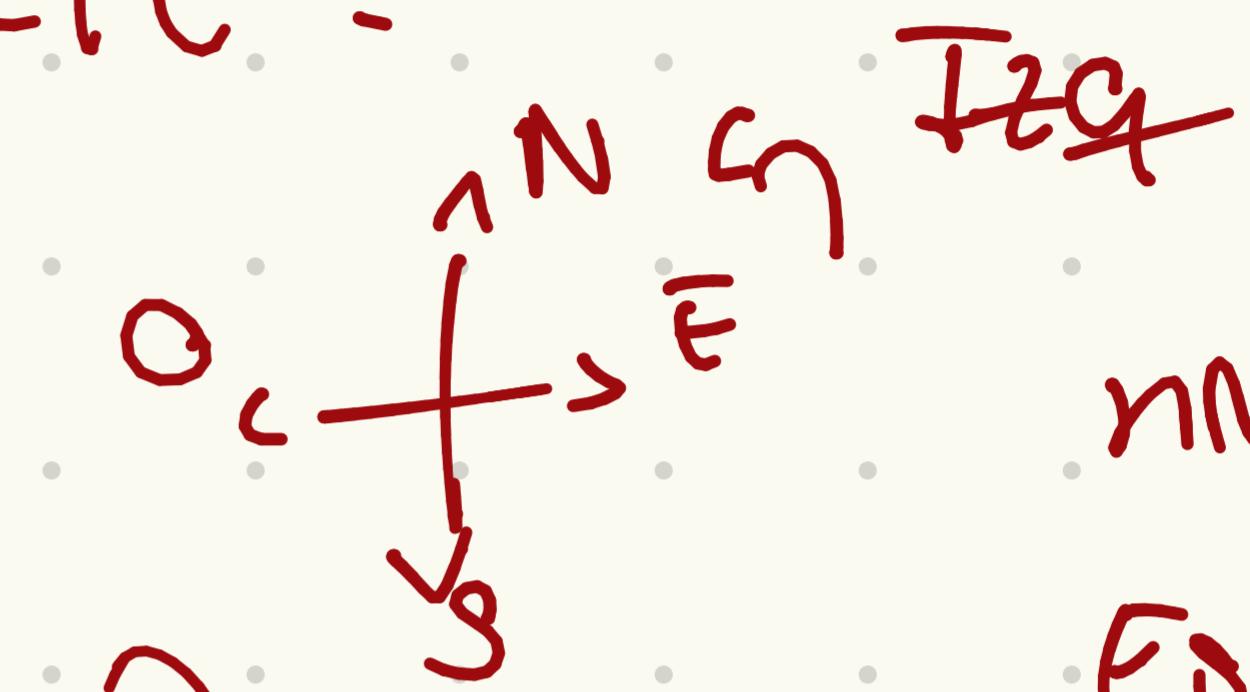
gira → cuántos giros?
brujula_actual = 0

z ← este z - 0 →

z - 0 = 2 izq

1 - 0 = 1 izq

3 - 0 = 3 izq



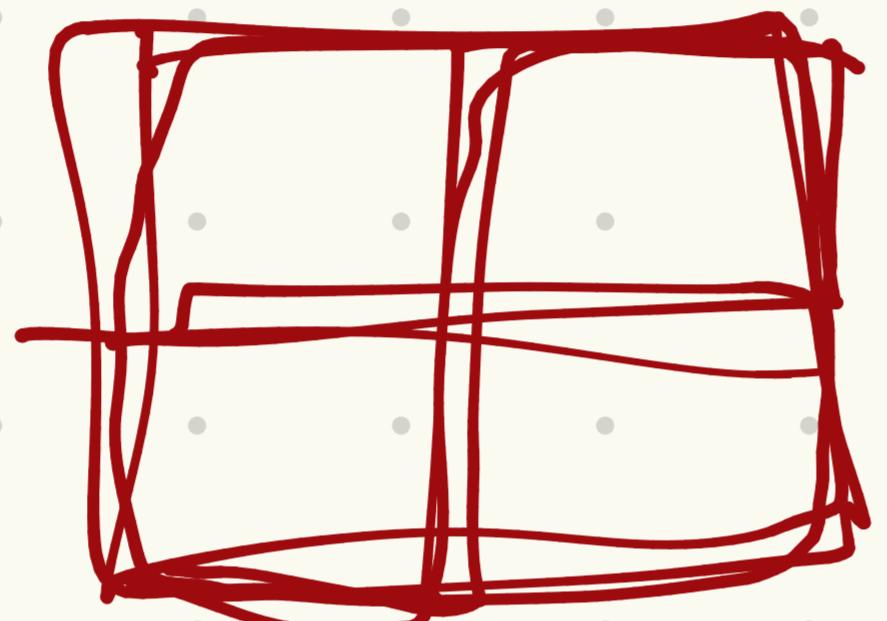
norte 0 -
Este 1 -
sur 2 -
oeste 3 .

este
oeste

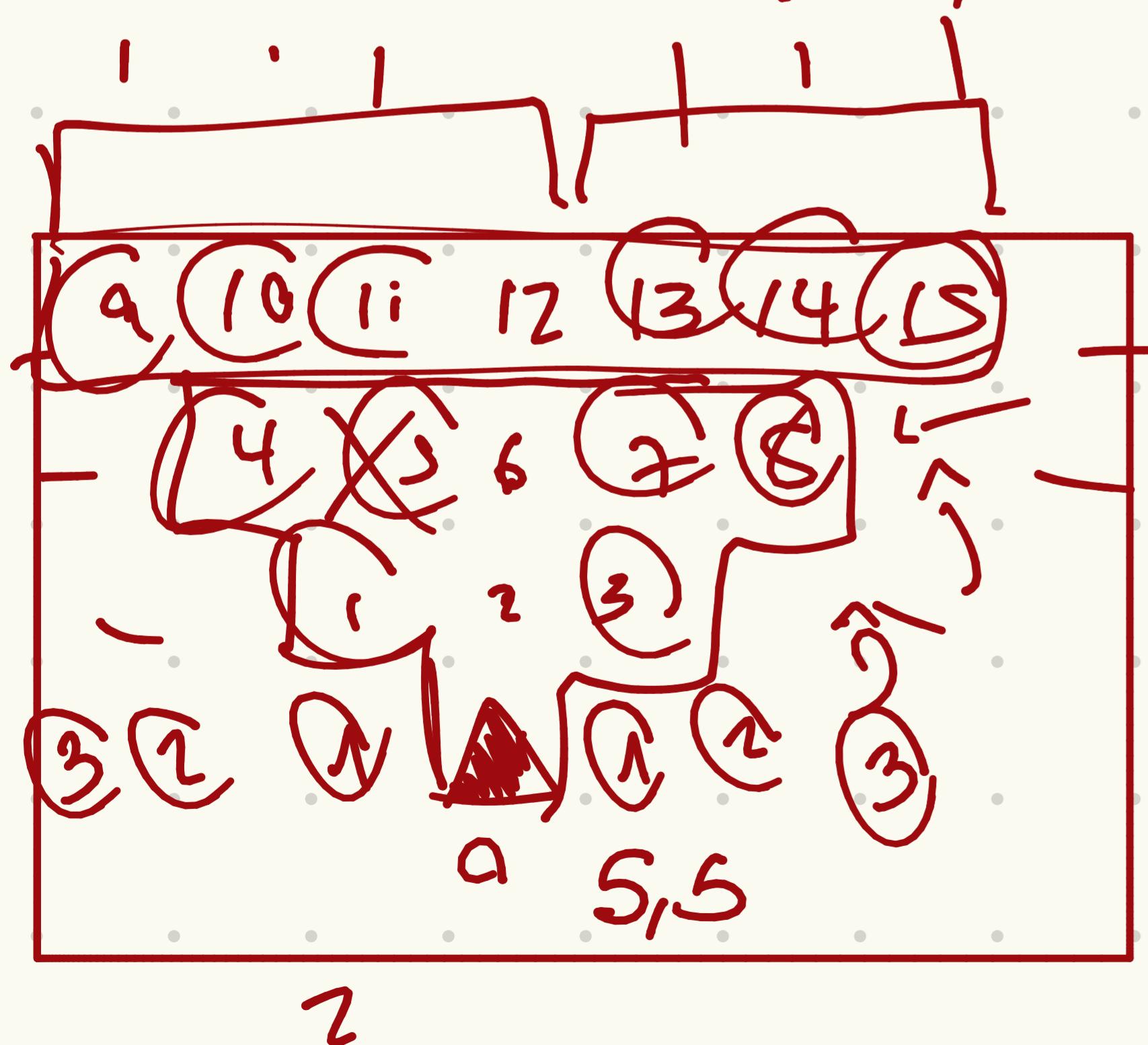
norte

Calculo la posición donde está
tengo la función pero no
considera los

18 12 -' dentro cuadrado.



¿Cómo hago para



reconocer la casilla que busco?

siempre pava arriba

lueser Tzg, der ->. nov 3

88.

6,7

S.S. + X, X
fila col.

hace 1^o columna { Tengo ordenado
luego fila según los datos

- * terminar la de buscar
 - * Luego mov zig zag -
 - * configurar aldeano y lobos.

$\text{PCSC} = \text{posF}$

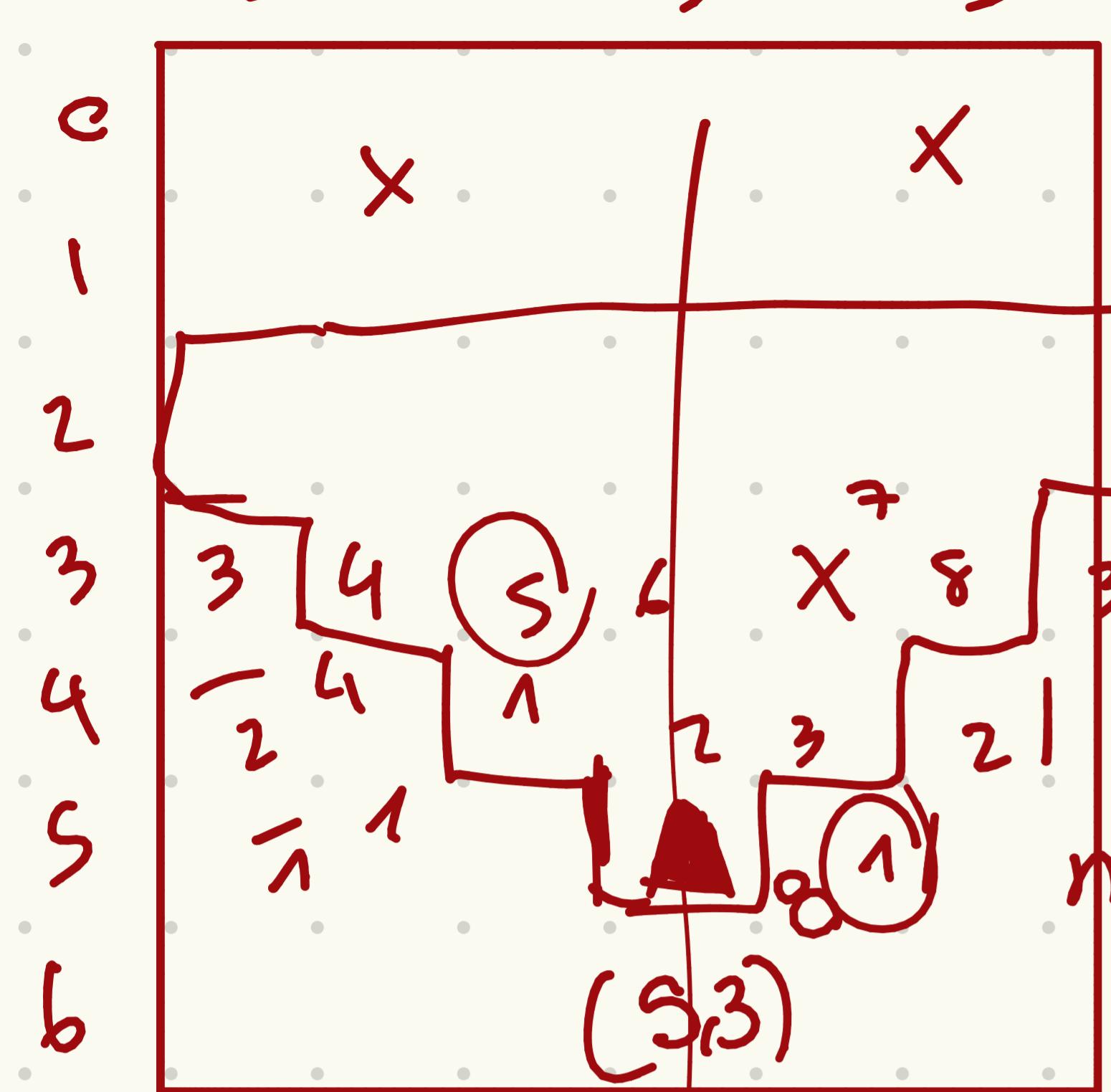
3 niveles

123 (1) condicón

4 S 678(2) condition

9. 10. 11. 12. 13. 14. 15. (3)

3 condizioni



Zarriba
Lizq.

Sensores terrenos [7]

1 colonna
2 fila

Clasame
a resto
al original)

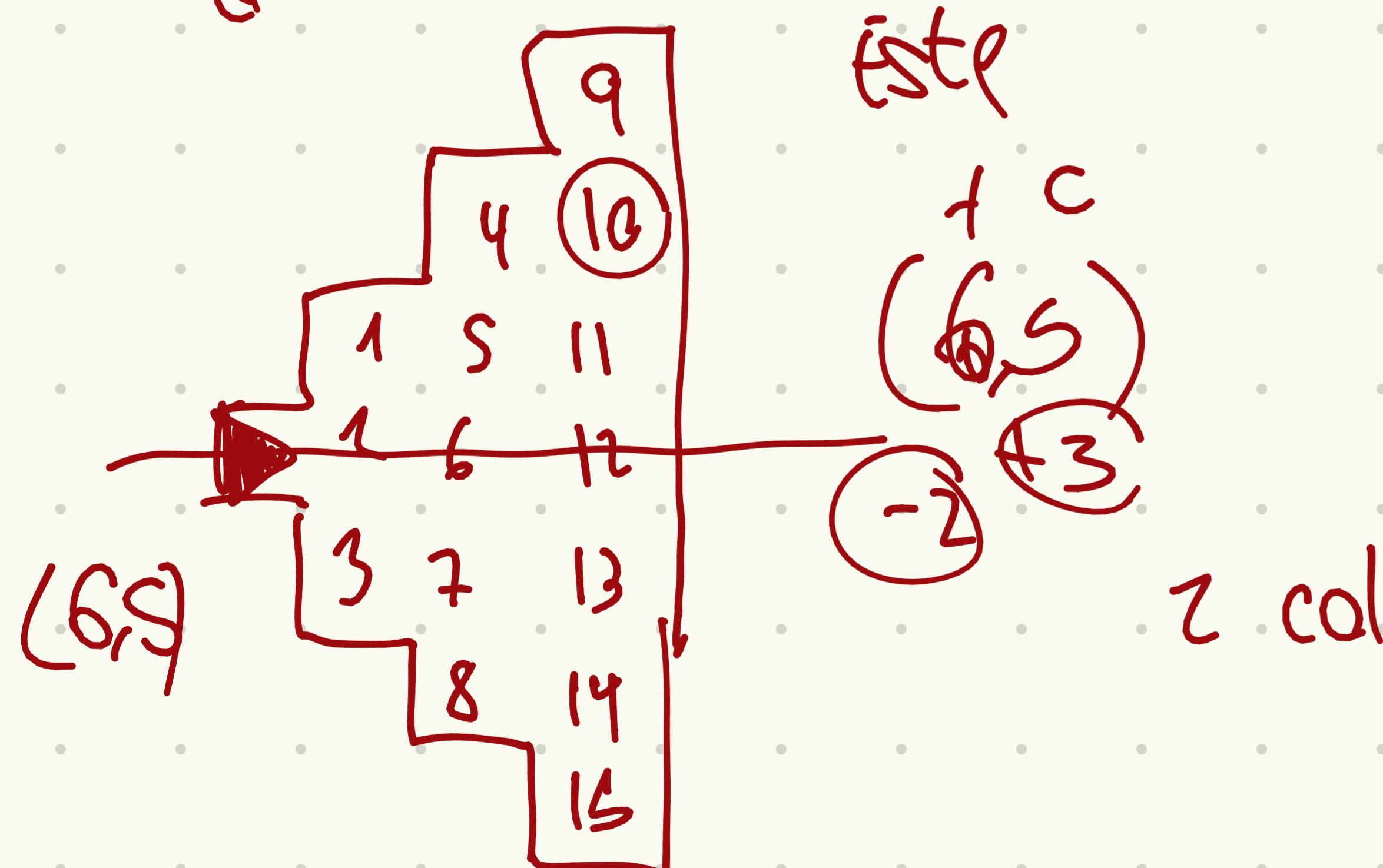
fil col.

$$(5, 3) + (x, y) = (3, 4)$$

fila siempre suma

norte
este
sur
oeste

fib +	cd + -
fila +-	cd f
fila -	cdl + -
fila +-	cd -



(5,3)

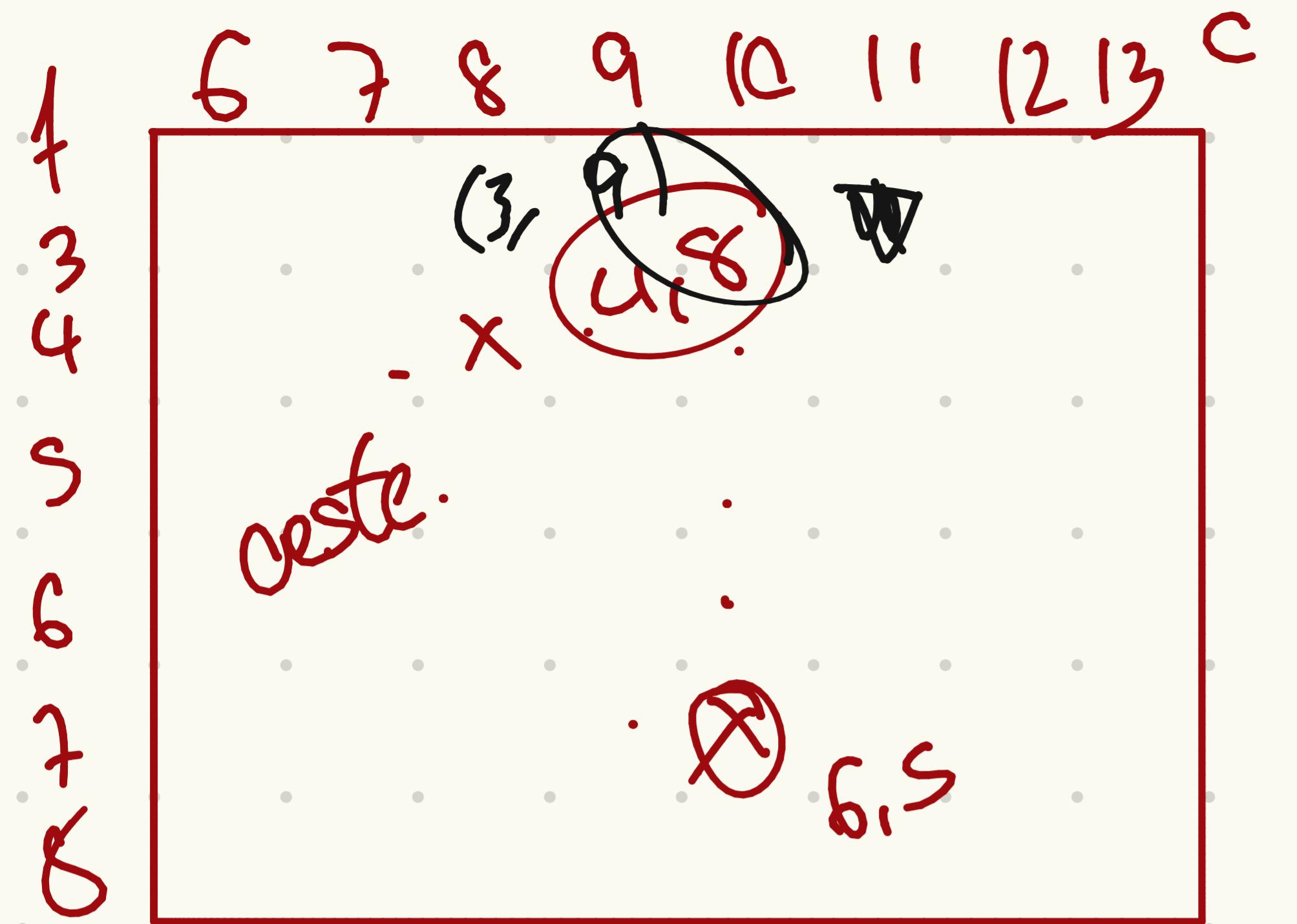
z.col

6, 7
7, 7
8, 7
8, \$

(4, 10)

A (39)

Con una vez me
bastó



8-q neg.

1,2

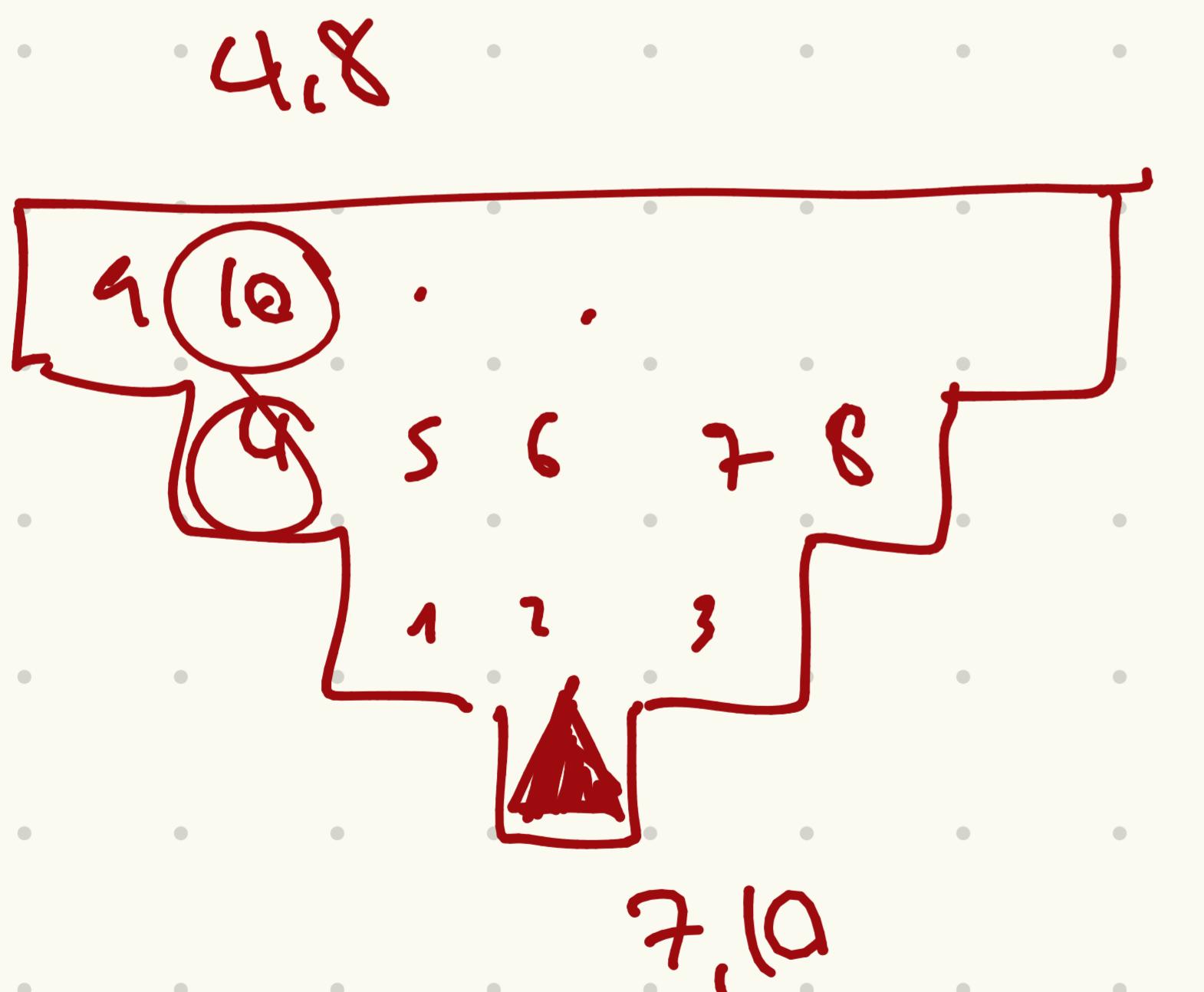
4,7

**PROBLEMA
CON
EL
GIRO!**

7 10

4,8.

7, 10
4, 8

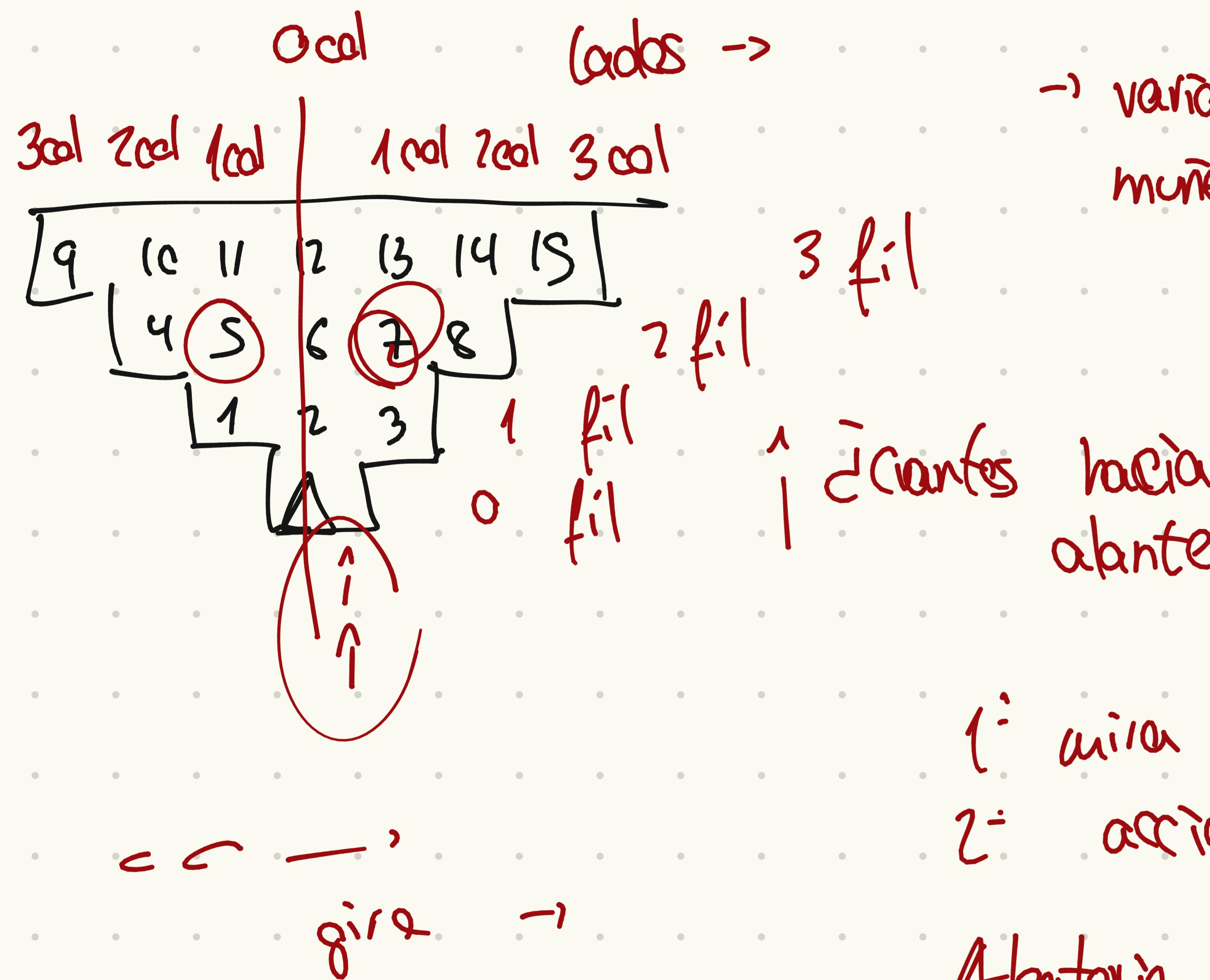


(7, 10)
(4, 8)

orig
dest

arig
dest

Guardar acciones → Lista: X



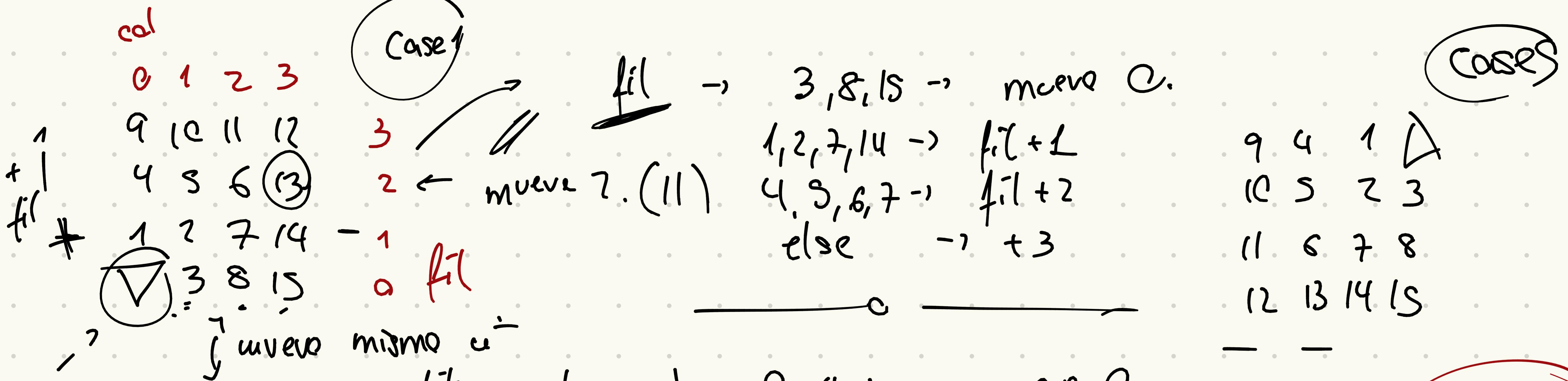
→ varía según sentido donde esté mí muñeca → NO HACE falta sentido.

+ 1 pose por delante

empieza a ver, antes no ha visto → ve en aleatoria.

- 1º mira sensor → modo.
2º acción

Aleatoria → búsqueda → Aleatoria ✓
búsqueda → Aleatoria ✓.
búsqueda → Aleatoria → búsqueda X.



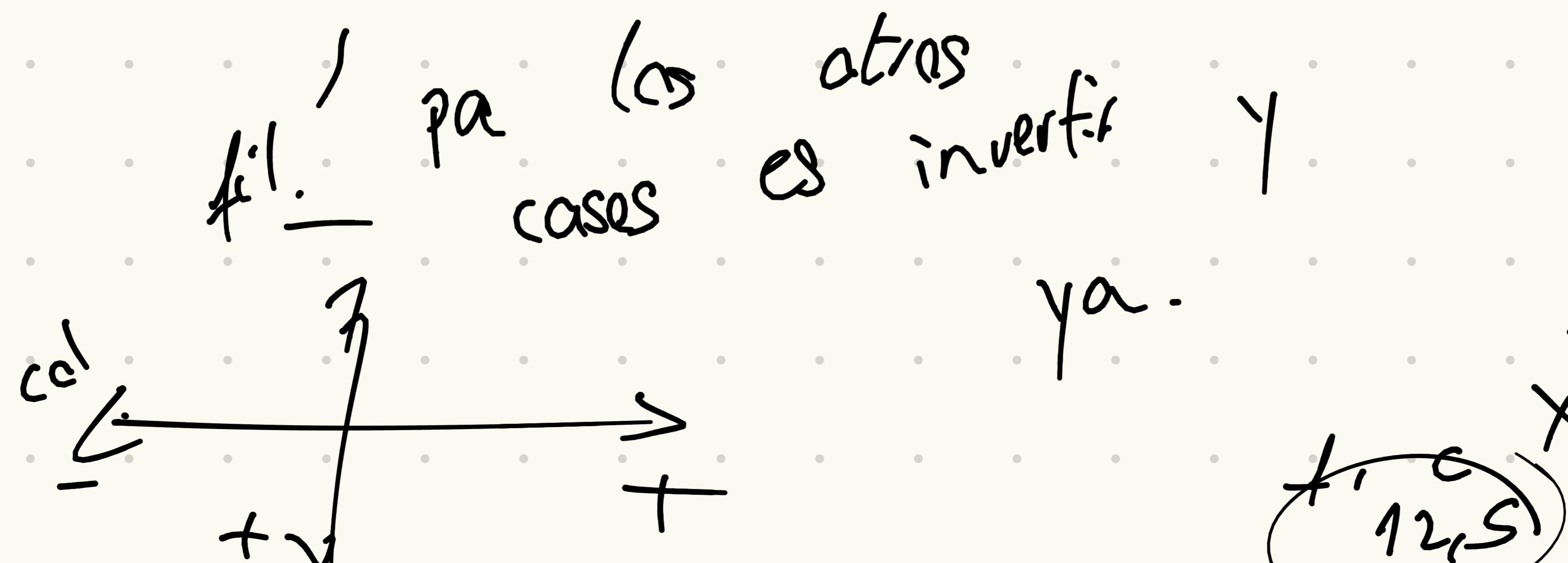
fila x col. col
 (8, 10)

9, 4, 1 → move C
 10, 5, 2, 3 → move 1
 11, 6, 7, 8 → move 2
 else → + 3.

case 7

case 3

- +



0 10, 7
 10

12 13 14 15

11 6 7 8

10 5 2 3

9 4 1

- encontrado zap
encontrado bikini

} INTERVALO

Z⁺

- 3 variable
encontrada

→ busco intervalo

→ ambos → true

false → buscando
true → encontrado

, cada 1 una vez.

Como se ve
Lí actualizó la casilla
solo cuando
busco -
en nov
aleatoria NO

5. Idear una función de potencial artificial (con componentes repulsivos y atractivos) que pueda ser utilizada para guiar un robot desde cualquier casilla del mundo bidimensional cuadriculado de la figura siguiente, a la casilla objetivo que está marcada con una X (suponer que las posibles acciones que puede ejecutar el robot son ir al norte, sur, este y oeste). ¿Tienen las componentes repulsivas y atractivas algún mínimo local? Si es así, ¿dónde?

Necesito controlar \rightarrow muñeca gire más con batería baja.

booleano \neg batería bajo \rightarrow afecte a los sensores avanzar. Si bateria-baja
 \rightarrow pasos-giro.

pasos-giro = 0.

\hookrightarrow giro-rápido = true;

\neg pasos-giro

- que no se recargue 1 vez.

if batería \neg baja

\rightarrow empezamos decrementar

\neg cada 10 pasos \rightarrow decrementamos