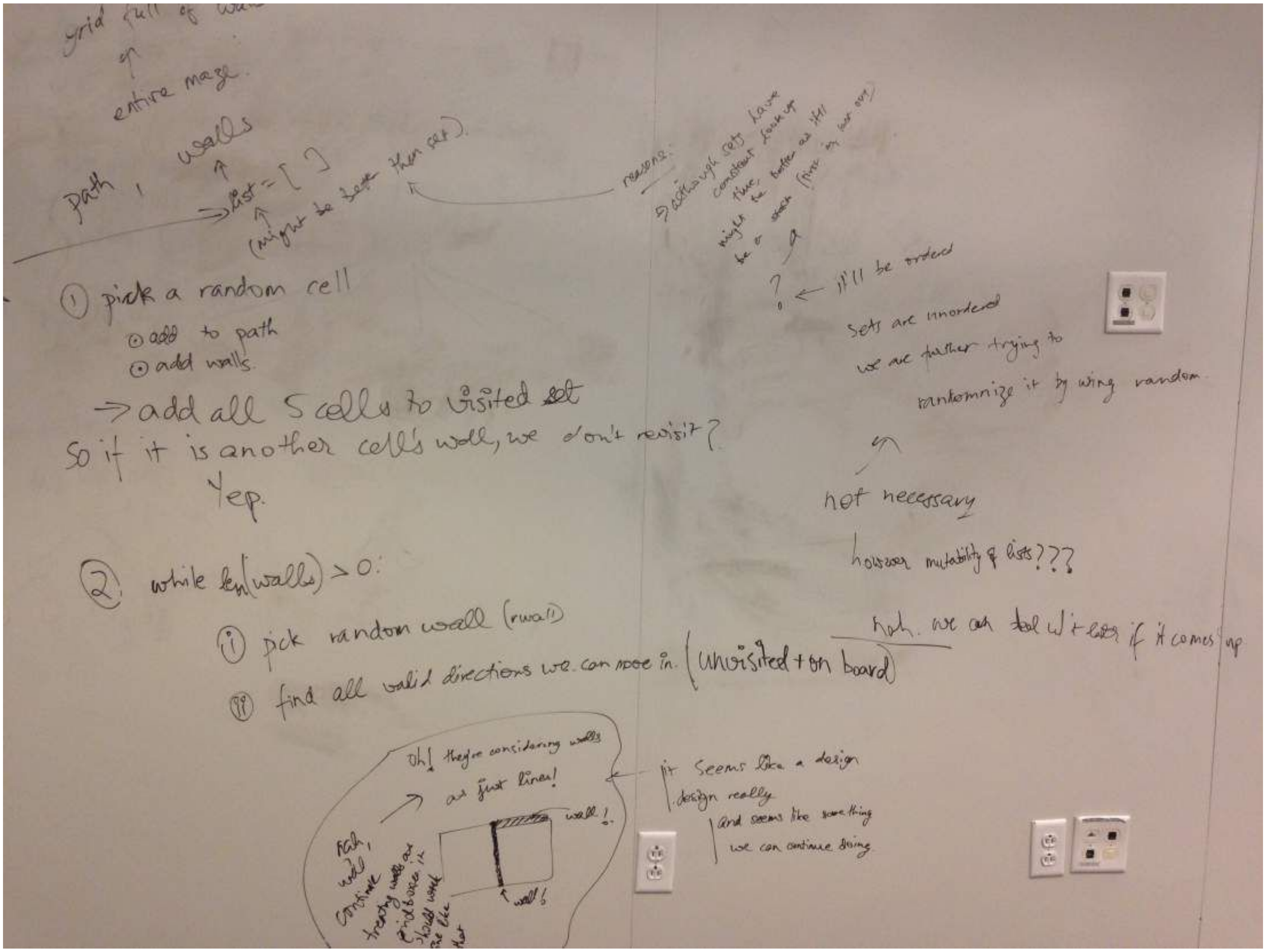
THOUGHT PROCESS



grid full of wa

↑

entire maze.

path , walls

↑

→ list = [ ]

↑

(might be better then set).

reasons:

⇒ although sets have constant look up time, better as it'll might be a stack (first in last out)

? ← it'll be ordered

sets are unordered

we are further trying to randomnize it by using random

↑

not necessary

however mutability of lists???

nah. we can deal w/ it later if it comes up

① pick a random cell

⊙ add to path

⊙ add walls.

→ add all 5 cells to visited ~~set~~

So if it is another cell's wall, we don't revisit?

Yep.

② while len(walls) > 0:

ⓘ pick random wall (rwall)

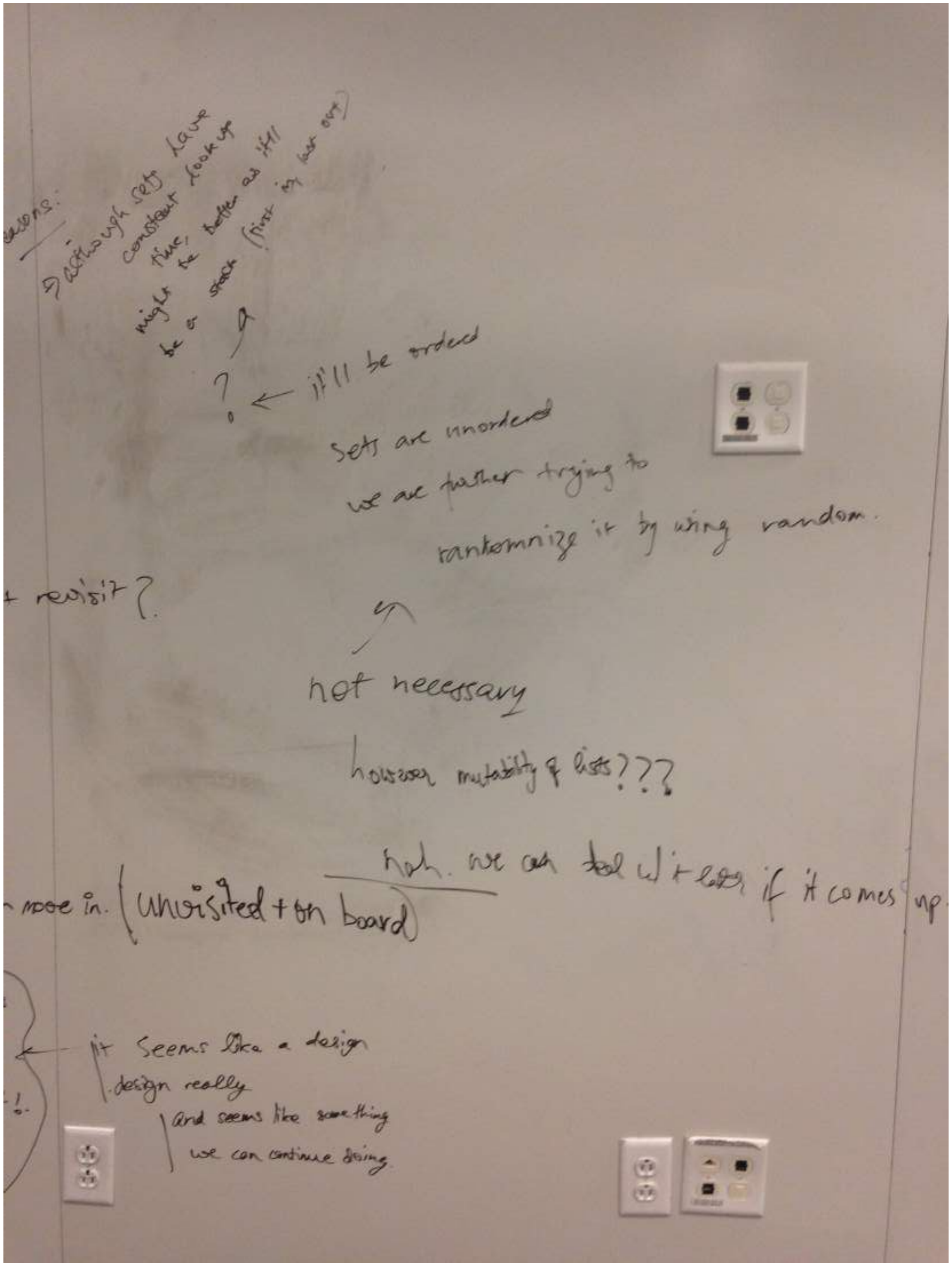
ⓘⓘ find all valid directions we can move in. (unvisited + on board)

Oh! they're considering walls as just lines!

Aah, well continue treating walls as gridboxes it should work like that

← it seems like a design design really

and seems like something we can continue doing



easons:

⇒ although sets have constant look up time, might be better as it'll be a stack (first in last out)

? ← it'll be ordered

sets are unordered

we are further trying to

rankomnize it by using random.

not necessary

however mutability of lists???

nah. we can deal w/ it later if it comes up.

t revisit?

moee in. (unvisited + on board)

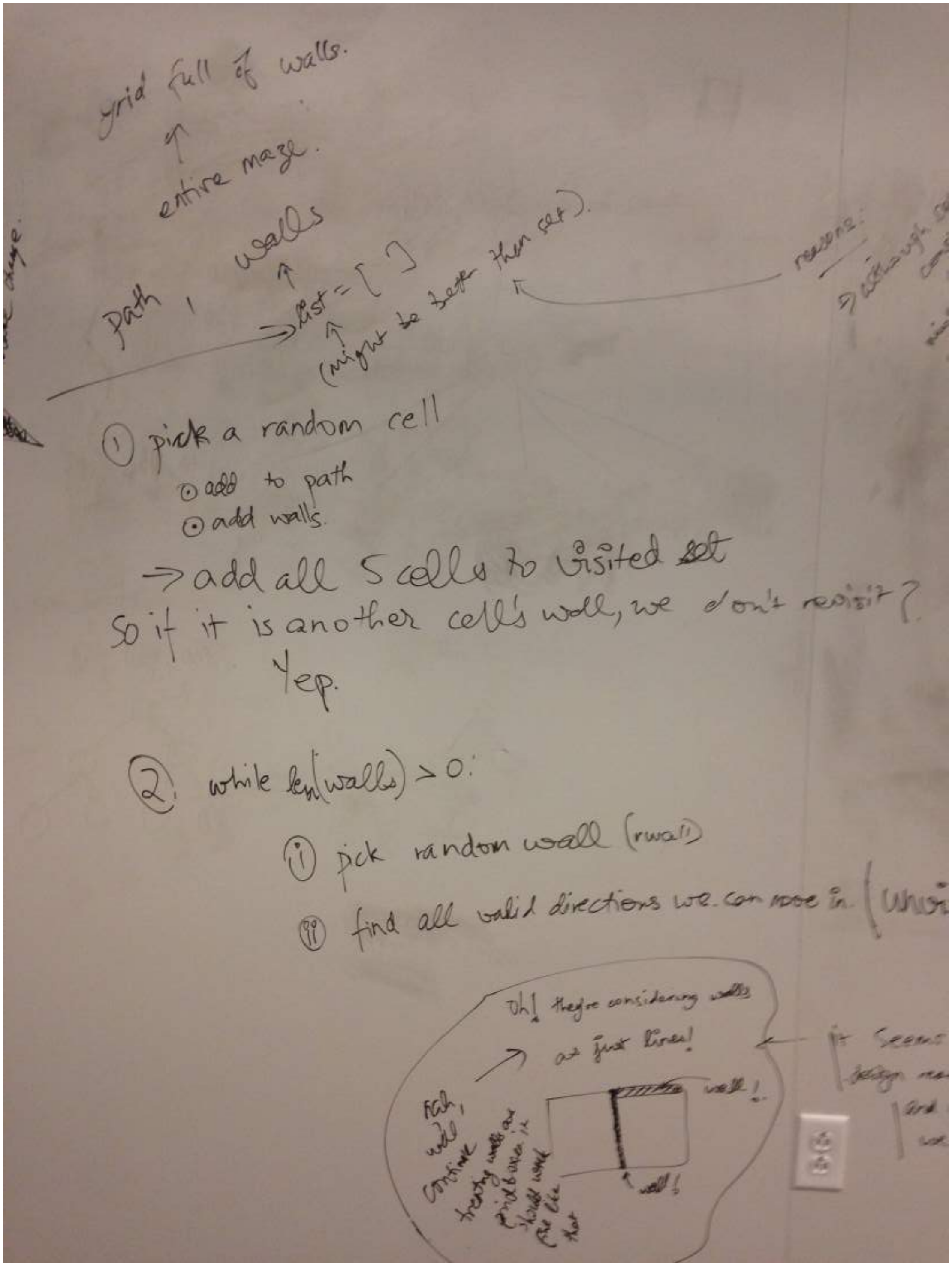
it seems like a design

design really

and seems like something

we can continue doing.

!.



grid full of walls.

↑

entire maze.

path , walls

↑

→ list = [ ]

↑

(might be better then set). ← reasons:

⇒ although se

① pick a random cell

⊙ add to path

⊙ add walls.

→ add all 5 cells to visited ~~set~~

So if it is another cell's wall, we don't revisit?

Yep.

② while len(walls) > 0:

(i) pick random wall (rwall)

(ii) find all valid directions we can move in (whi

Oh! they're considering walls as just lines!

wall!

wall!

Ah, wall, continue treating walls as pixelboxes it should work like that

← it seems

design nea

and

we

conditional

★ (iii) if there is a move we can make from the wall, (to new cell)

```
→ make rwall → path
→ add newcell to
    visited/path.
⇒ add neighboring cells to walls
    (for newcell)
```

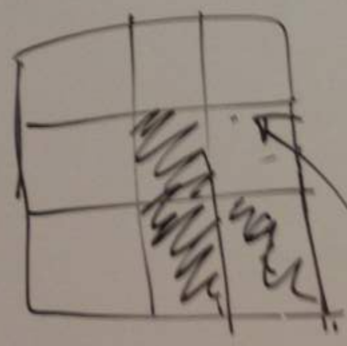
(iv) removerwall from walls list.

make change.

Problems with maze:

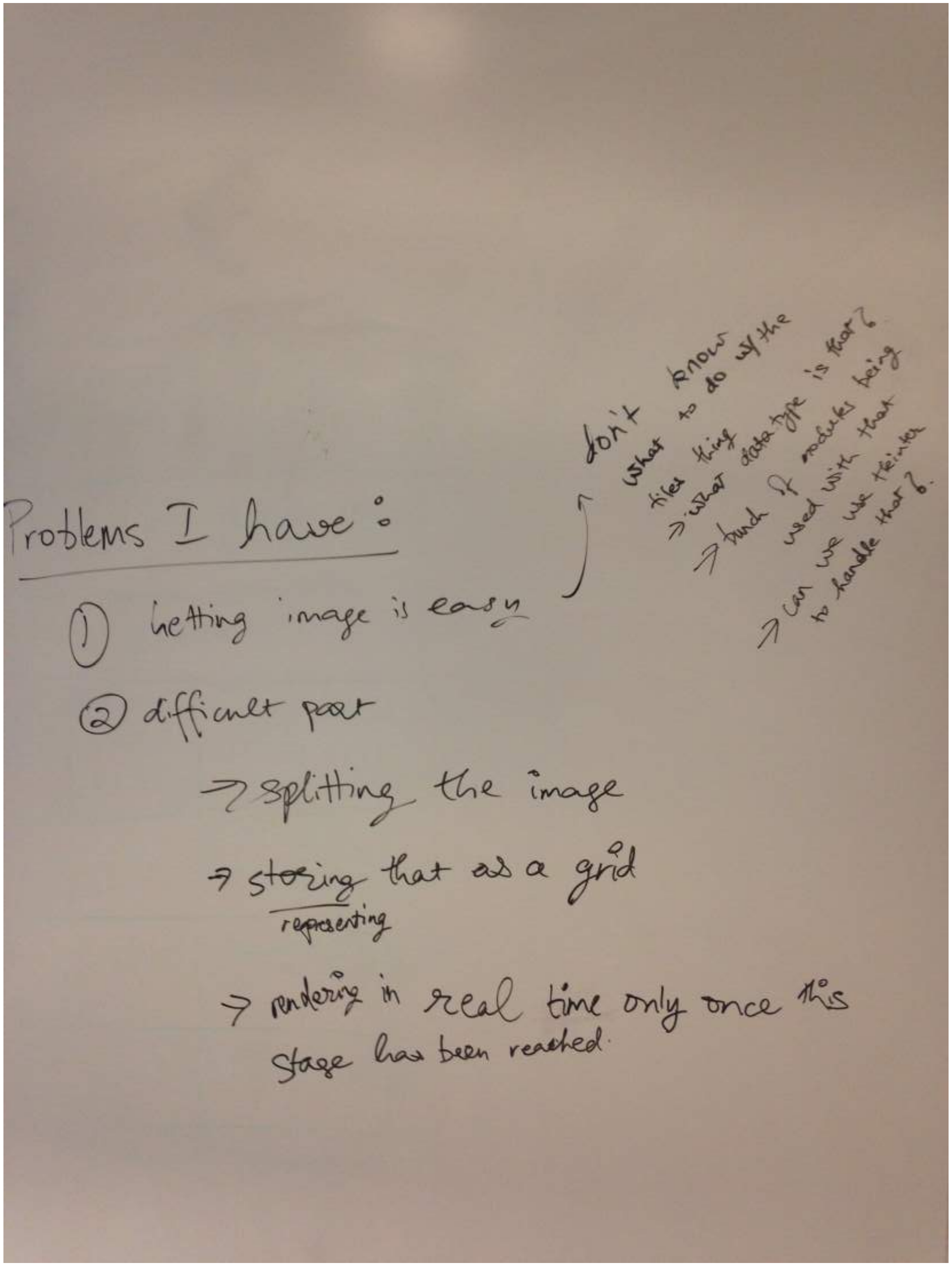
1) Don't know how/why double walling/path is happening.

how to detect it:



this should not be happening

how to detect this?



Problems I have :

① Getting image is easy

② difficult part

→ splitting the image

→ storing that as a grid
representing

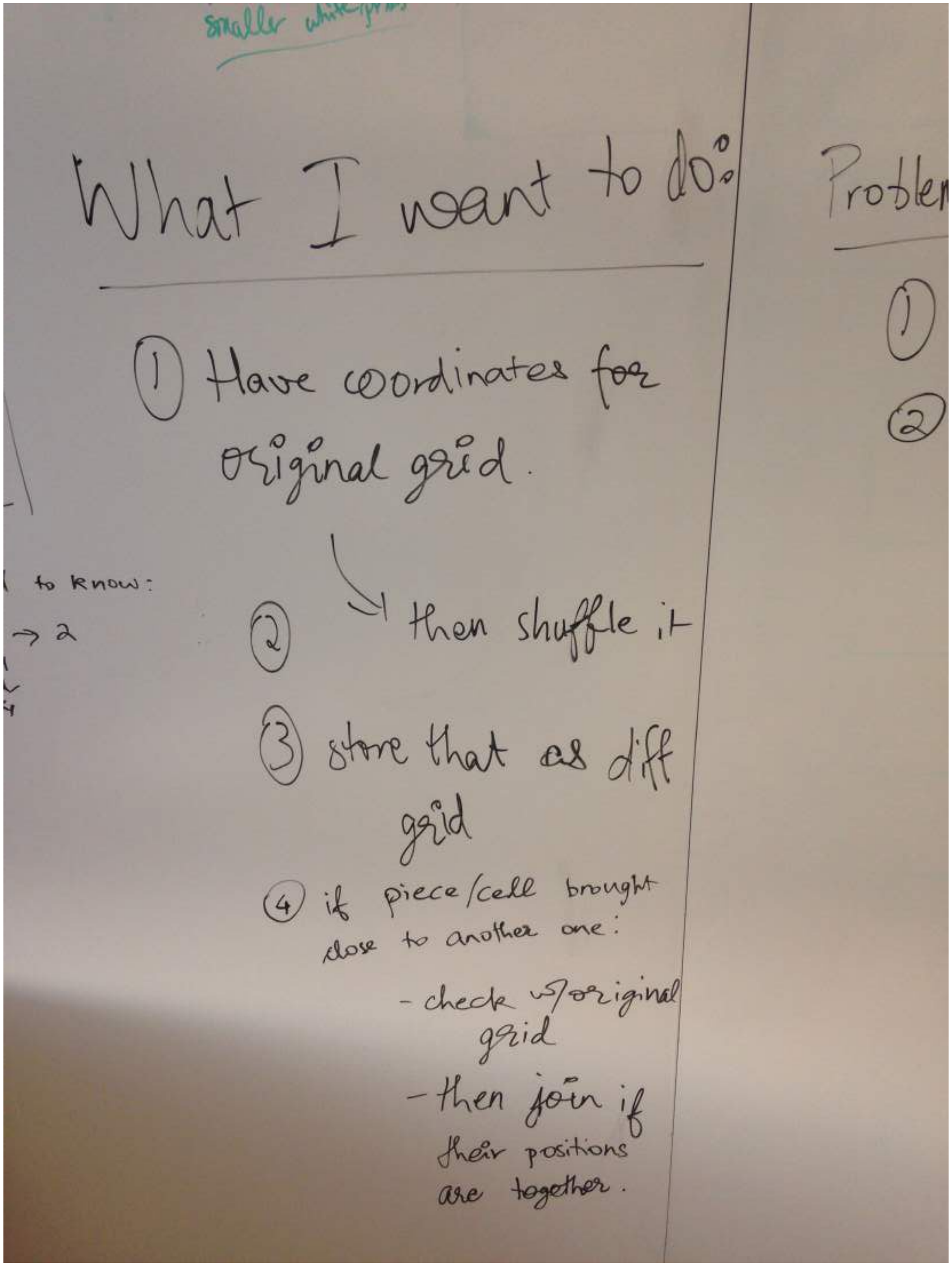
→ rendering in real time only once this stage has been reached.

don't know what to do w/ the tiles thing

→ what data type is that?

→ bunch of modules being used with that

→ can we use tkinter to handle that?



smaller white[illegible]

What I want to do:

1. Have coordinates for original grid.
2. then shuffle it
3. store that as diff grid
4. if piece/cell brought close to another one:
    - check w/original grid
    - then join if their positions are together.

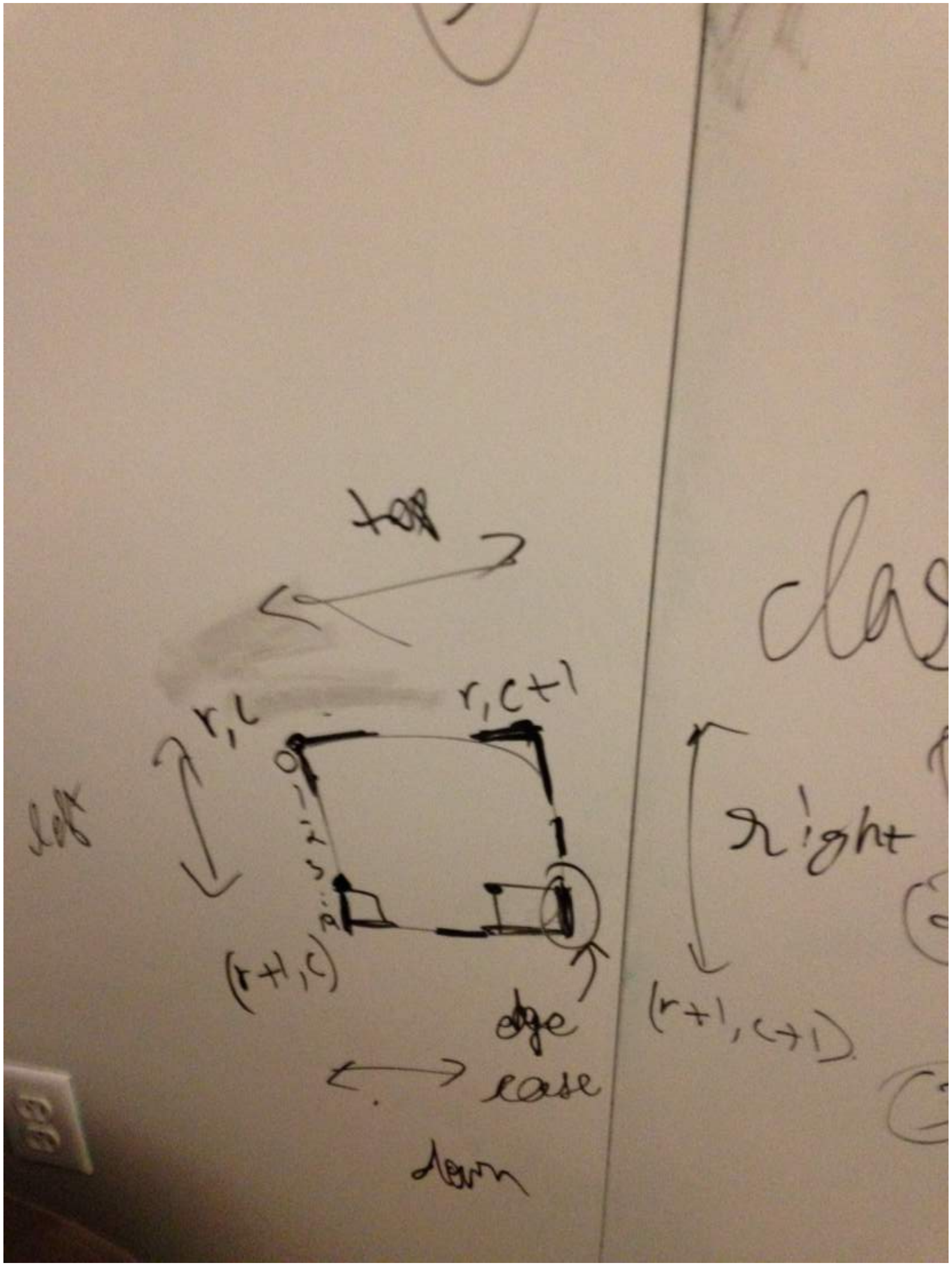
to know:

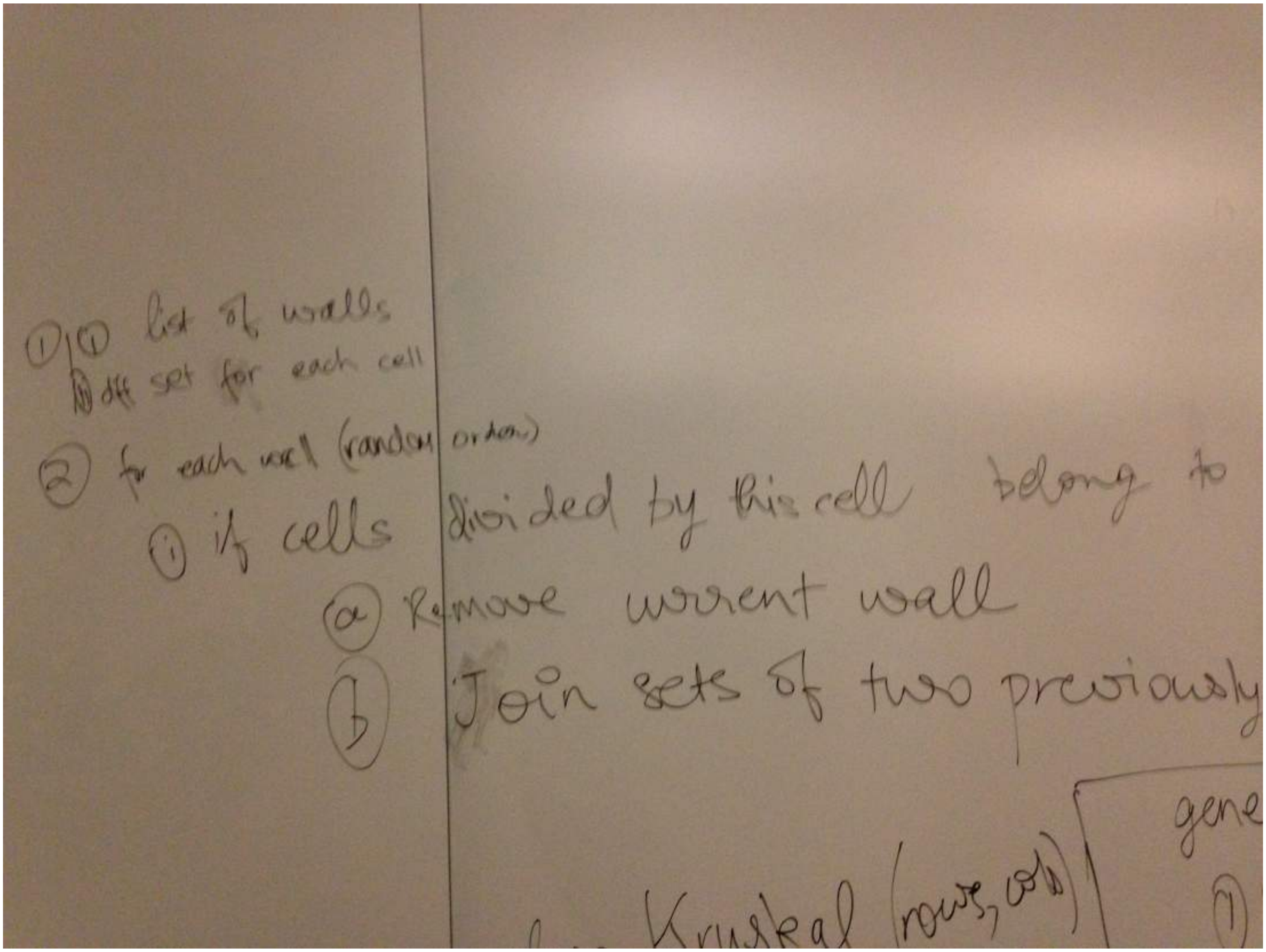
→ 2

Problem

1.

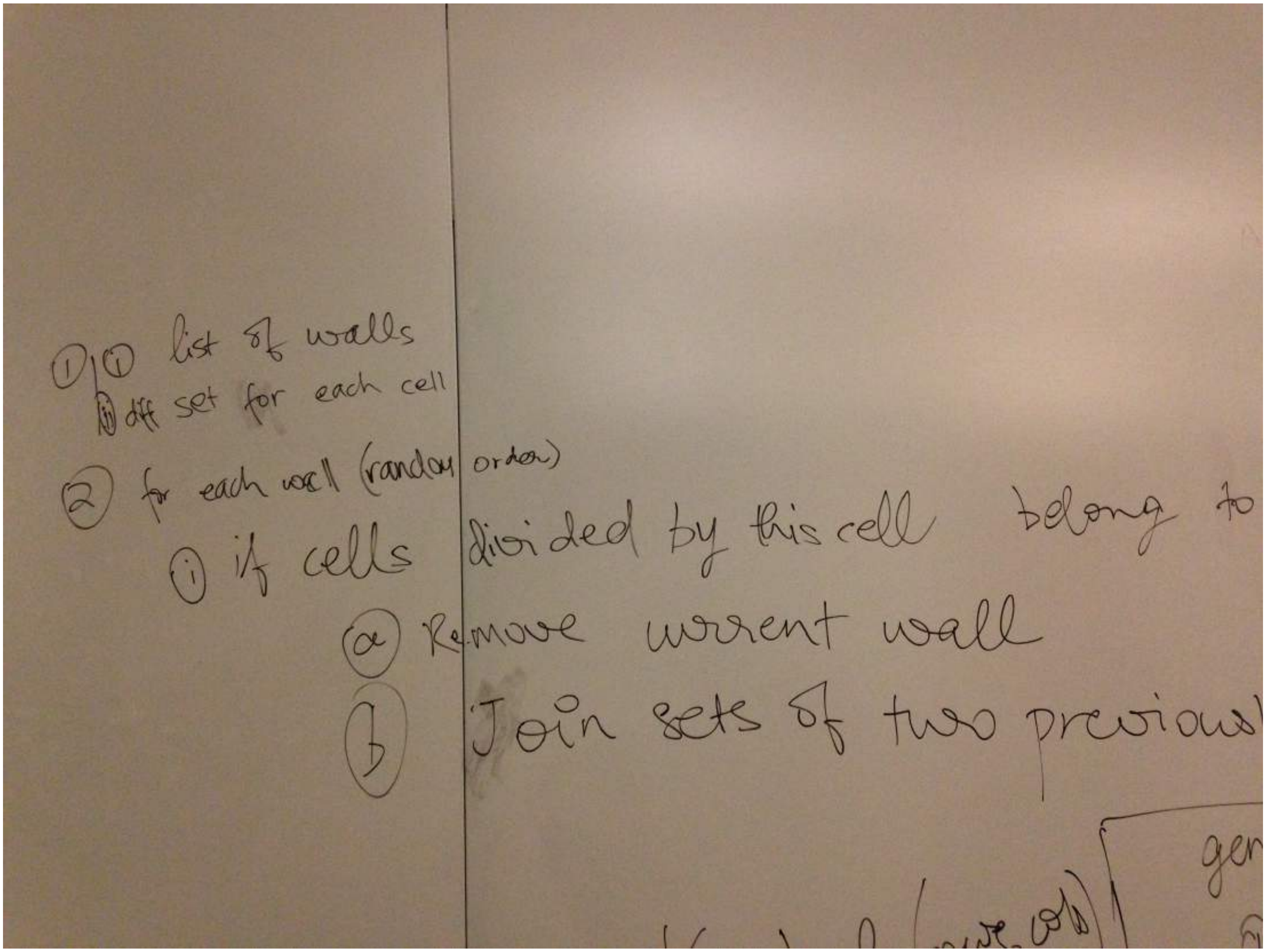
2.


top
r,c
r,c+1
left
0
1
2
3
n
(r+1,c)
edge case
down
right
(r+1,c+1)
clas



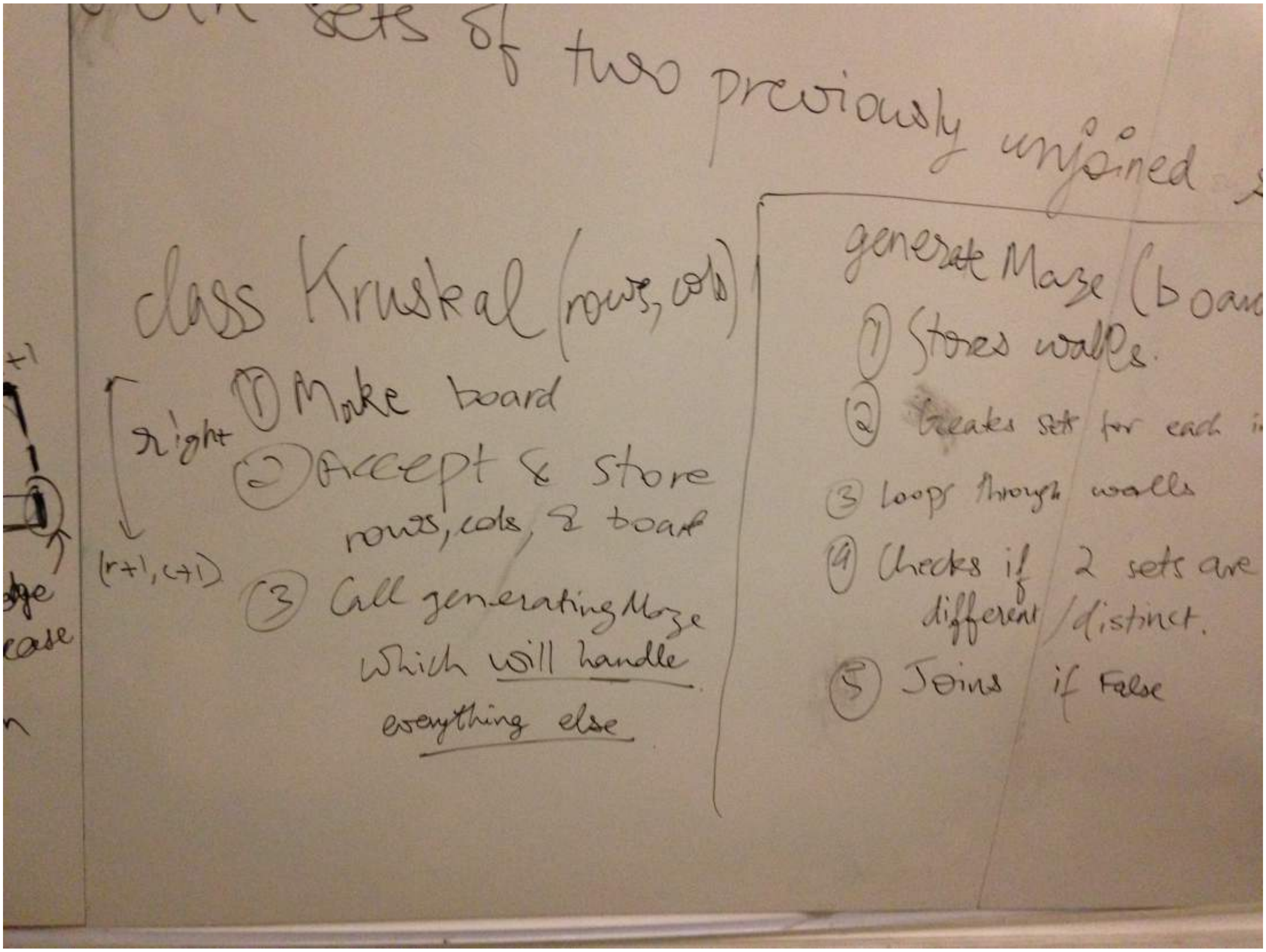
```
(1) (i) list of walls
    (ii) dff set for each cell
(2) for each wall (random order)
    (i) if cells divided by this cell belong to
        (a) Remove current wall
        (b) Join sets of two previously

Kruskal (rows, cols)    gene
                        (1)
```



```
(1) (i) list of walls
    (ii) diff set for each cell
(2) for each wall (random order)
    (i) if cells divided by this cell belong to
        (a) Remove current wall
        (b) Join sets of two previous
```

gen



sets of two previously unjoined

+1

dge
case
n

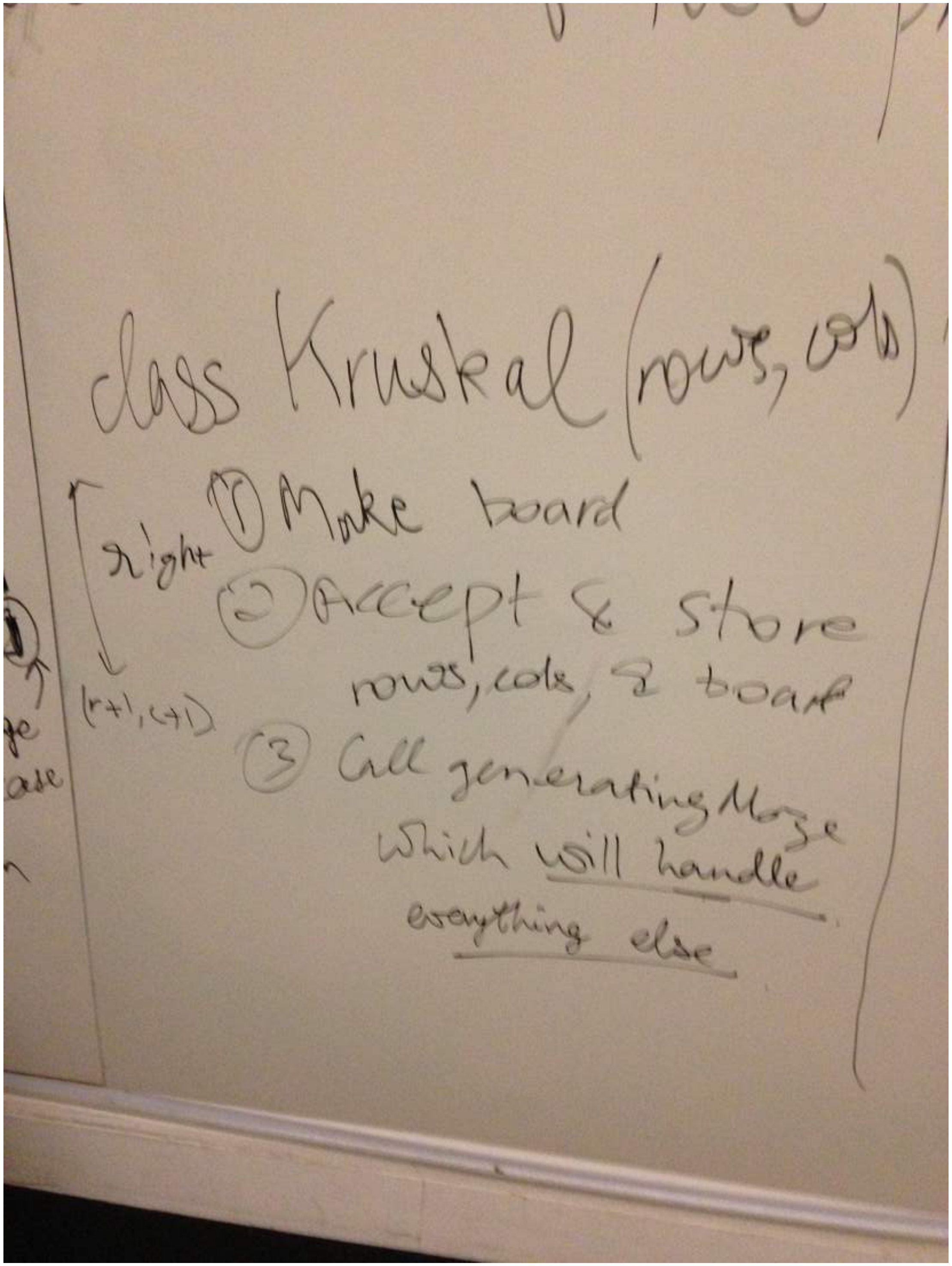
class Kruskal (rows, cols)

right
(r+1, c+1)

1. Make board
2. Accept & store rows, cols, & board
3. Call generatingMaze which will handle everything else

generate Maze (board

1. Stores walls.
2. Creates sets for each i
3. Loops through walls
4. Checks if 2 sets are different/distinct.
5. Joins if False

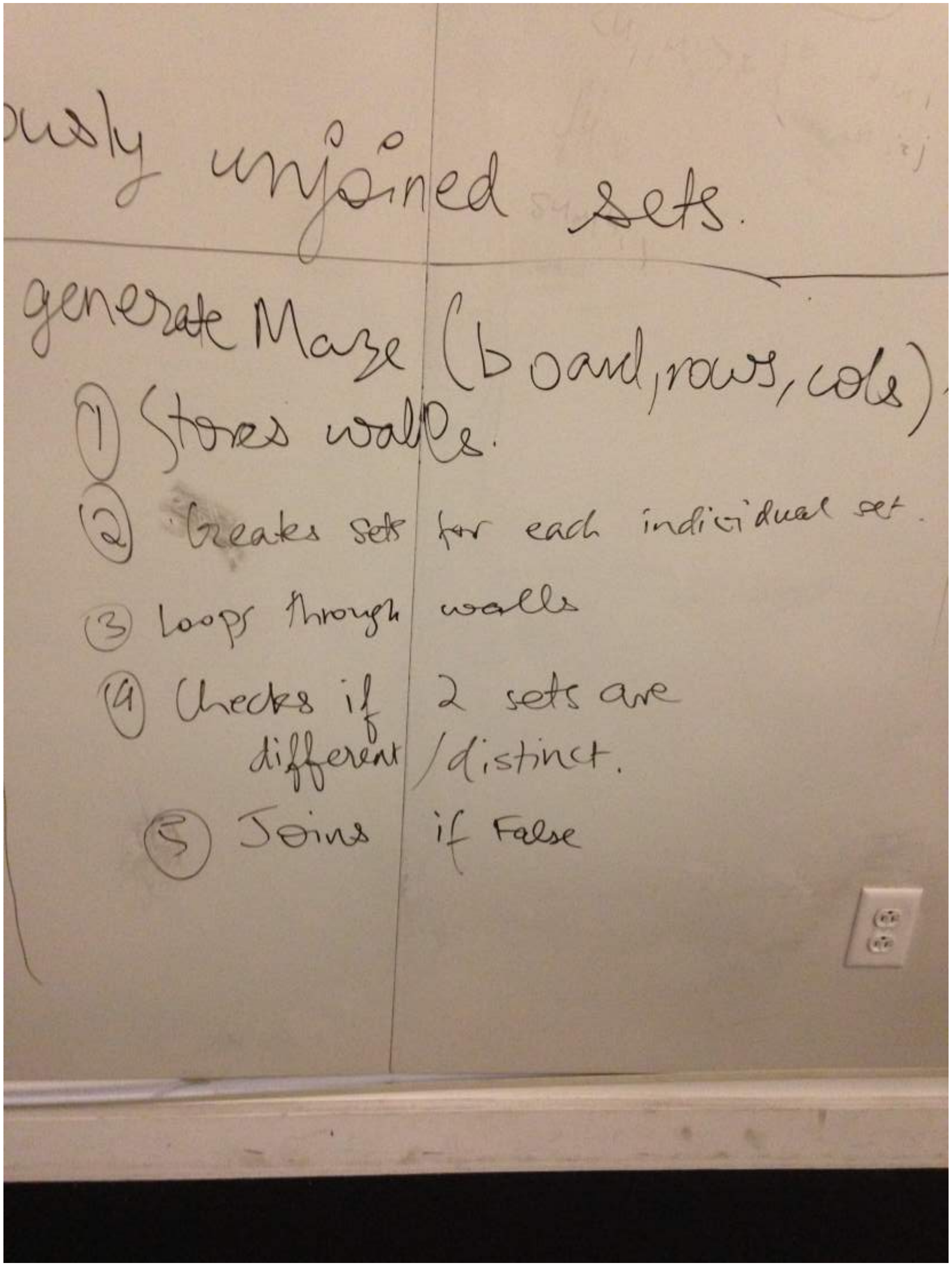


class Kruskal (rows, cols)

right

(r+1, c+1)

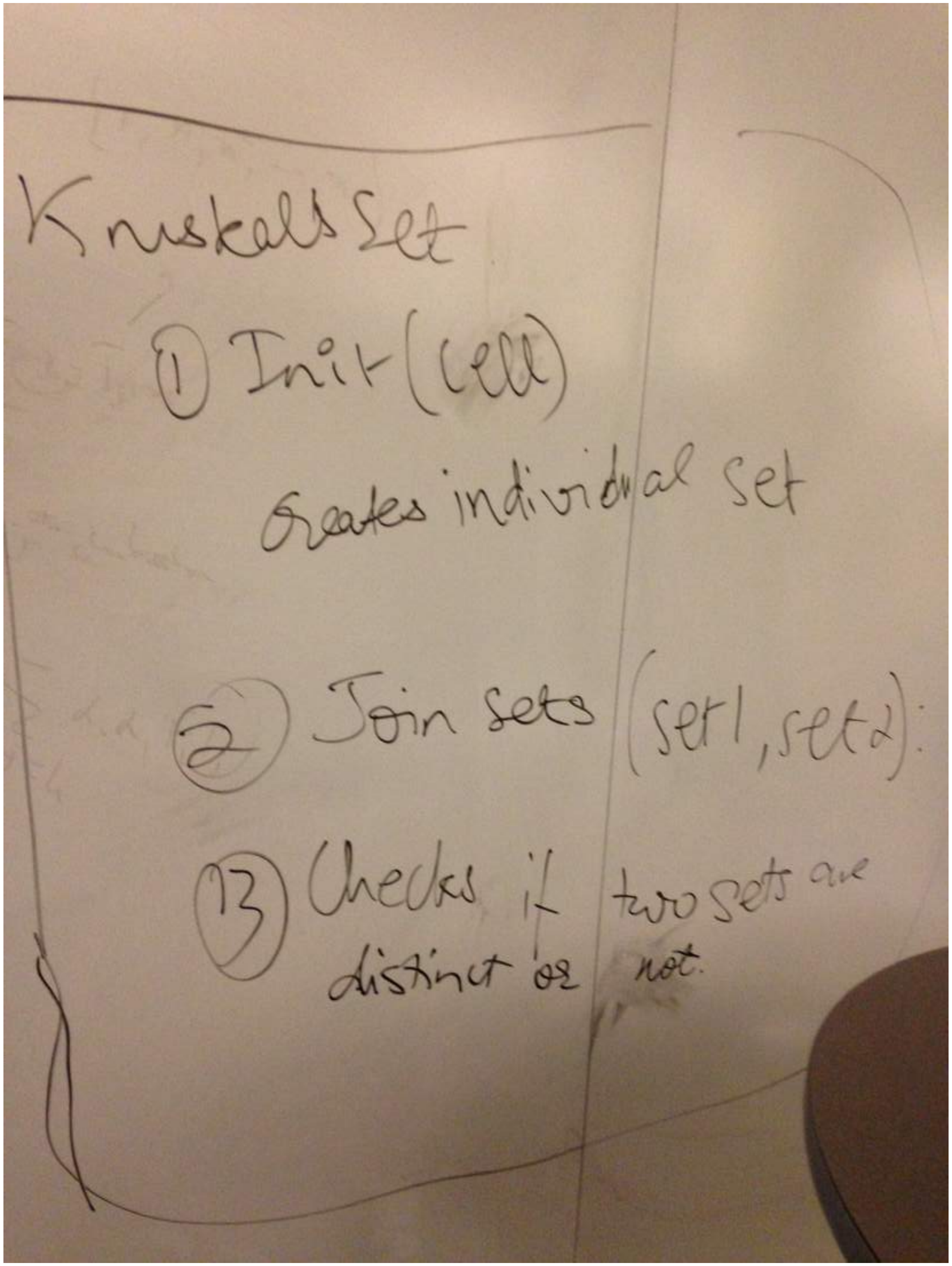
1. Make board
2. Accept & store rows, cols, & board
3. Call generatingMaze which will handle everything else



ously unjoined sets.

generate Maze (board, rows, cols)

1. Stores walls.
2. Creates sets for each individual set.
3. Loops through walls
4. Checks if 2 sets are different / distinct.
5. Joins if False



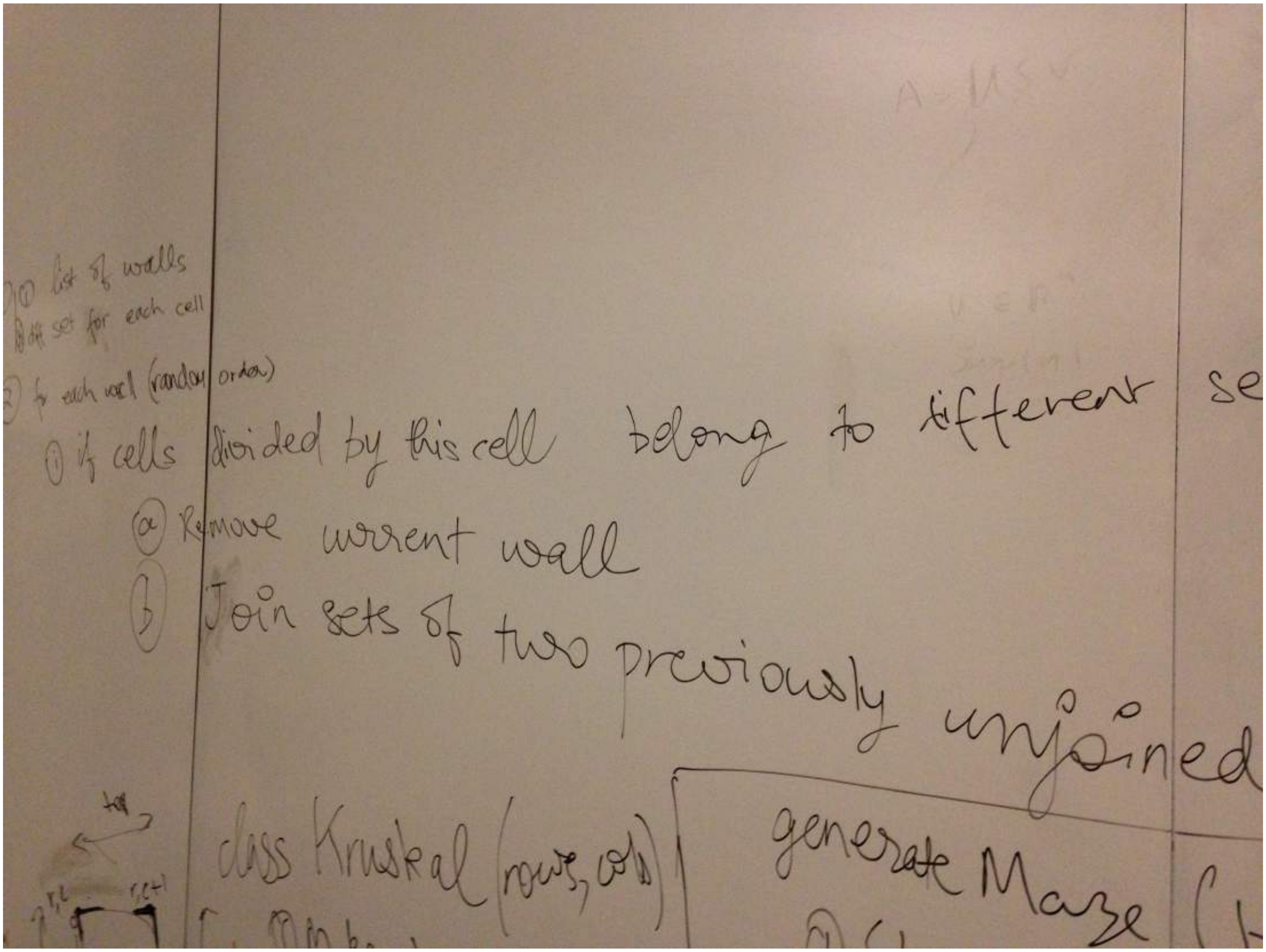
Kruskals Set

① Init(cell)

Creates individual set

② Join sets (set1, set2):

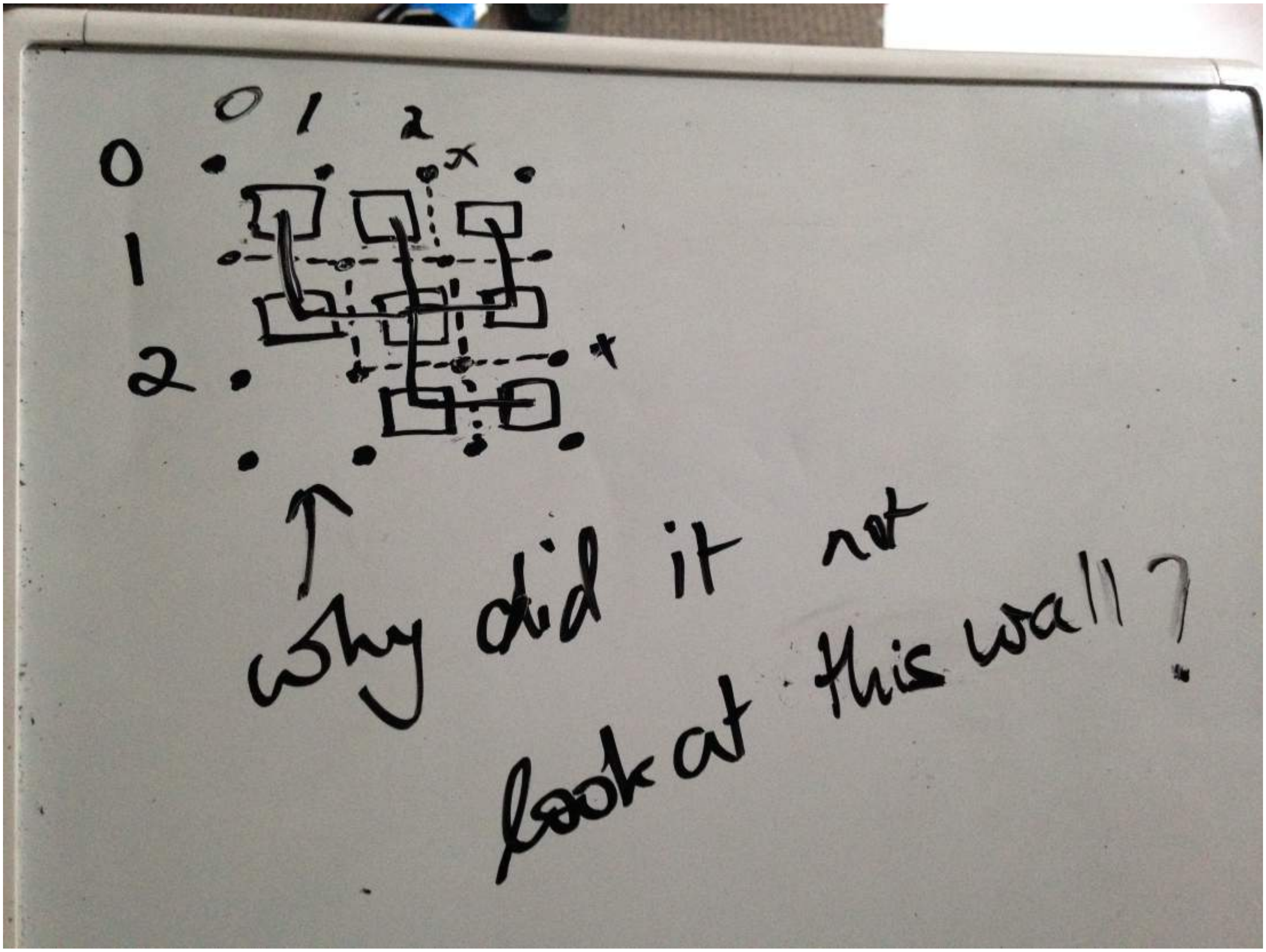
③ Checks if two sets are distinct or not.



```
1) ① list of walls
   ② diff set for each cell
② for each wall (random order)
   ① if cells divided by this cell belong to tifferent se
      a) Remove current wall
      b) Join sets of two previously unjoined
```

class Kruskal (rows, cols)

generate Maze



why did it not look at this wall?