**Step1.**

We create the board using helper function **create\_board()**

The helper function returns the initial positions of plants and zombies randomly.

We represents the board using plant list and zombie list.

def create\_board():

plant\_list = [(0, 0), (2, 0), (4, 0)]

zoom\_list = [(1, 4), (3, 4)]

return plant\_list, zoom\_list

The size of board is 5X5.

**Step2.**

We should search for all possible next states from current state.

At each time, every zombie can move either one of 4 directions. (up, down, left, right).

That’s why total number of possible next states is **4^n**, where n is the number of zombies.

zoom\_n = len ( zoom\_list)

total\_plan = 4\*\*zoom\_n

We should calculate every possible next state.

for i in range ( total\_plan):

p = i

We should calculate next state with next state “p” by moving each zombie in a corresponding way.

next\_zoom\_list = []

for j in range ( zoom\_n):

d = p % 4

p//= 4

r = zoom\_list[j][0] + wy[d]

c = zoom\_list[j][1] + wx[d]

next\_zoom\_list.append ( (r, c))

We should check this next state is valid or not using helper function is\_**valid(zoom\_list)**.

That is, every zombie should not bump each other, and should be placed in a valid board.

if is\_valid(next\_zoom\_list):

For valid next state, we should calculate next distance using helper function **calc\_dis(plant\_list, zoom\_list)**.

next\_dis = calc\_dis ( plant\_list, next\_zoom\_list)

The helper function **calc\_dis(plant\_list, zoom\_list)** calculates the board state using Manhattan distance as following.

The purpose of zombie is to eat all plants, so we match the nearest zombie for every plant and calculate all sum Manhattan distances of every plant and corresponding zombie.

If there is no matched zombie, we sum up 100(very large) into the total distance.

If next state is better (distance is shorter than current state), we select the next state and update the board and exit the next state loop.

If zombie eats a plant, we remove the plant from plant list.

if next\_dis < cur\_dis:

cur\_dis = next\_dis

zoom\_list.clear()

for zoom in next\_zoom\_list:

zoom\_list.append ( zoom)

plant\_n = len ( plant\_list)

for j in range ( plant\_n):

if plant\_list[j] == zoom:

plant\_list.pop(j)

break

**Step3.**

We repeats Step2 until there is no plant left.

while cur\_dis > 0: