**Step 1.**

First we input the current state and final state of the puzzle board.

This is performed by helper function **input\_board(msg)**.

This helper function checks whether the board is valid or not.

**msg** is the string to indicate whether this is initial state or final state.

If the board is valid then the number 1~8 should be used only once.

board = []

This will be filled with 3X3 integer matrix containing from 0 to 8.

0 means empty space.

**Step 2.**

We update current state into new state by moving neighbor cells around empty space.

There are four possible 4 next states.

r0, c0 = find\_pos ( cur\_board, 0)

cur\_dis = calc\_dis ( cur\_board, final\_board)

for i in range ( 4):

r1 = r0 + wy[i]

c1 = c0 + wx[i]

Here (r0, c0) is the position of empty space.

We calculate the current Manhattan distance from current state and final state.

The helper function **find\_pos(board, num)** finds the position (row, column) of the ‘num’ on the ‘board’.

The helper function **calc\_dis(cur\_board, final\_board)** calculates the Manhattan distance between current board and final board.

(r1, c1) is the position of empty space for the next state.

If (r1, c1) is valid position, we calculate Manhattan distance for the next state.

If distance for next state is better (shorter) than current state, we update the board into next state.

if r1 >=0 and r1 < 3 and c1 >=0 and c1 < 3:

cur\_board[r1][c1], cur\_board[r0][c0] = cur\_board[r0][c0], cur\_board[r1][c1]

next\_dis = calc\_dis ( cur\_board, final\_board)

if next\_dis < cur\_dis:

is\_updated = True

break

At every step2, we show the board by helper function **show\_board(board)**

**Step 3.**

If the board is updated into next state, we repeat Step2.

If the board is not updated with any of next states, we stop the program.

while is\_updated: