```
#####################################DFS
ALGORITHM#########################################################################
import time
import sys

def ValidMove(board, x, y):
    """
    :param board: NxN board
    :param x: x position
    :param y: y position
    :return: return 2d board
    """
    return 0 <= x < len(board) and 0 <= y < len(board[0])


def ShowBoard(board):
    """
    :param board: NxN board
    :return: Print board on console
    """
    for row in board:
        print("".join(row))
    print()


def Neighbors(board, pos):
    """
    :param board: NxN board
    :param pos: New position of robot
    :return: List of its neighbor
    """
    neighbors = []
    x, y = pos
    for dx, dy in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
        new_x, new_y = x + dx, y + dy
        if ValidMove(board, new_x, new_y) and board[new_x][new_y] != 'O':
            neighbors.append((new_x, new_y))
    return neighbors


def Successor(board, robot, boxes, storage, depth, actions=[]):
    """
    :param board: NxN board
    :param robot: Robot position
    :param boxes: Boxes Position
    :param storage: Storage Position
    :param depth: Depth
    :param actions: List of actions performed
    :return:Show next board state
    """
    if sorted(boxes) == sorted(storage):
        return actions

    if depth == 0:
        return None

    for neighbor in Neighbors(board, robot):
        new_robot = neighbor
        new_boxes = list(boxes)
        for i, box in enumerate(new_boxes):
            if box == new_robot:
                new_box = (box[0] + (box[0] - robot[0]), box[1] + (box[1] - robot[1]))
                if ValidMove(board, *new_box) and board[new_box[0]][new_box[1]] != 'O':
                    new_boxes[i] = new_box
                    break

        new_board = [list(row) for row in board]
```

```
        for box in new_boxes:
            x, y = box
            new_board[x][y] = 'B'
        new_board[new_robot[0]][new_robot[1]] = 'R'
        ShowBoard(new_board)

        result = Successor(new_board, new_robot, tuple(new_boxes), storage, depth - 1, actions
+ [new_robot])

        if result:
            return result

    return None


def Initstate():
    """
    :return: Initial state of the board
    """
    print('Initial Board\n')
    file_obj = open(sys.argv[1])
    rows = file_obj.readlines()
    for line in rows:
        if line != "":
            board.append(line.strip())
        else:
            break
    return board


def CallSuccessorWithDepth(board, robot, boxes, storage):
    """
    :param board: NxN board
    :param robot: Position of Robot
    :param boxes: Position of box
    :param storage: Position of storage
    :return: Count of the depths
    """
    depth = 0
    while True:
        print(f"Depth: {depth}")
        result = Successor(board, robot, boxes, storage, depth)
        if result:
            return result
        depth += 1


def GetStoragePos(board):
    """
    :param board: NxN board
    :return: Positions of the Storages
    """
    return {(x, y) for x in range(len(board)) for y in range(len(board[0])) if board[x][y] ==
'S'}


def GetRobotPos(board):
    """
    :param board: NxN board
    :return: Positions of the Robot
    """
    return [(x, y) for x in range(len(board)) for y in range(len(board[0])) if board[x][y] ==
'R'][0]


def GetBoxPos(board):
    """
```

```
    :param board:NxN board
    :return: Positions of the boxes
    """
    return tuple((x, y) for x in range(len(board)) for y in range(len(board[0])) if board[x]
[y] == 'B')


# Driver Code
board = []
board = Initstate()

storage = GetStoragePos(board)
robot = GetRobotPos(board)
boxes = GetBoxPos(board)

start_time = time.time()

actions = CallSuccessorWithDepth(board, robot, boxes, storage)

if actions:
    print("End state can be found")
    solution_board = [list(row) for row in board]
    for action in actions:
        x, y = action
        solution_board[x][y] = 'R'

    ShowBoard(solution_board)
else:
    print("End state cannot be found")

print(f'Total execution time is {time.time() - start_time}')
```