```
#######################################BFS
ALGORITHM#####################################################################

from collections import deque
import sys
import time

board = []


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 Neighbours(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 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 Successor(board, robot, boxes, storage):
    """
    :param board: NxN board
    :param robot: Location of robot
    :param boxes: Location of boxes
    :param storage: Location of the storage
    :return:Show next board state
```

```python
    """
    start_state = (robot, boxes)
    visited = set()
    queue = deque([(start_state, [])])

    while queue:
        (robot, boxes), actions = queue.popleft()

        if sorted(boxes) == sorted(storage):
            return actions

        if (robot, boxes) in visited:
            continue

        visited.add((robot, boxes))

        solution_board = [list(row) for row in board]
        for box in boxes:
            x, y = box
            solution_board[x][y] = 'B'
        solution_board[robot[0]][robot[1]] = 'R'
        ShowBoard(solution_board)

        for neighbor in Neighbours(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_state = (new_robot, tuple(new_boxes))
            new_actions = actions + [neighbor]
            queue.append((new_state, new_actions))

    return None


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')
```

```python
# Driver Code
board = Initstate()

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

start_time = time.time()
actions = Successor(board, robot, boxes, storage)

if actions:
    print("End  solution  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}')
```

```
#####################################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]
```

```python
        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):
    """
```

```python
    :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}')
```

```
#######################################GBS
ALGORITHM############################################################
from heapq import heappop, heappush
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 ManhattanDistance(a, b):
    """
    :param a: Any position
    :param b: Any position
    :return: Absolute distance between 2 co-ordinates
    """
    return abs(a[0] - b[0]) + abs(a[1] - b[1])


def Heuristic(board, robot, boxes, targets):
    """
    :param board: NxN Board
    :param robot: Robot position
    :param boxes: Boxes position
    :param targets: Target locations
    :return: total distance
    """
    total_dist = 0
    for box in boxes:
        min_dist = min(ManhattanDistance(box, target) for target in targets)
        total_dist += min_dist
    return total_dist


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


def Successor(board, robot, boxes, targets):
    """
    :param board: NxN board
    :param robot: Location of robot
    :param boxes: Location of boxes
    :param targets: Location of the targets
    :return:Show next board state
    """
    directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]

    def is_goal(boxes):
        return sorted(boxes) == sorted(targets)

    def apply_move(entity, direction):
        return (entity[0] + direction[0], entity[1] + direction[1])

    start_state = (robot, tuple(sorted(boxes)))
    visited = set()
    priority_queue = [(Heuristic(board, robot, boxes, targets), start_state)]  # (heuristic,
(robot, boxes))

    while priority_queue:
```

```python
        _, (robot, boxes) = heappop(priority_queue)

        if is_goal(boxes):
            return boxes

        if (robot, boxes) in visited:
            continue

        visited.add((robot, boxes))

        solution_board = [list(row) for row in board]
        for box in boxes:
            x, y = box
            solution_board[x][y] = 'B'
        solution_board[robot[0]][robot[1]] = 'R'
        ShowBoard(solution_board)

        for direction in directions:
            new_robot = apply_move(robot, direction)
            if not ValidMove(board, *new_robot) or board[new_robot[0]][new_robot[1]] == 'O':
                continue

            if (new_robot, boxes) not in visited:
                heappush(priority_queue, (Heuristic(board, new_robot, boxes, targets),
(new_robot, boxes)))

    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 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]
```

```python
        [y] == 'B')


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

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

start_time = time.time()
result = Successor(board, robot, boxes, storage)

if result:
    print("End  state can be found:")
    solution_board = [list(row) for row in board]
    for box in result:
        x, y = box
        solution_board[x][y] = 'B'
    print("\n".join(["".join(row) for row in solution_board]))
    print()

else:
    print("End state  cannot  be  found.")

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

```python
#########################################ASTAR
ALGORITHM#######################################################
import heapq
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 ManhattanDistance(pos1, pos2):
    """
    :param a: Any position
    :param b: Any position
    :return: Absolute distance between 2 co-ordinates
    """
    return abs(pos1[0] - pos2[0]) + abs(pos1[1] - pos2[1])


def Heuristic(board,robot, boxes, storagepace):
    """
     :param board: NxN Board
     :param robot: Robot position
     :param boxes: Boxes position
     :param storagepace: storagspace  positions
     :return: total distance
     """
    total_dist = 0
    for box in boxes:
        min_dist = min(ManhattanDistance(box, target) for target in storagepace)
        total_dist += min_dist
    return total_dist + ManhattanDistance(robot, boxes[0])


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


def Successor(board, robot, boxes, storage):
    """
     :param board: NxN board
     :param robot: Location of robot
     :param boxes: Location of boxes
     :param storage: Location of the storage
     :return:Show next board state
     """
    print("Pukoban Using A Star Algorithm")
    directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]

    def is_goal(boxes):
        return sorted(boxes) == sorted(storage)

    def apply_move(entity, direction):
        return (entity[0] + direction[0], entity[1] + direction[1])
```

```python
    def valid_move(entity, direction):
        new_entity = apply_move(entity, direction)
        if not ValidMove(board, *new_entity) or board[new_entity[0]][new_entity[1]] == 'O':
            return False
        return True

    start_state = (robot, tuple(sorted(boxes)))
    visited = set()
    priority_queue = [(0, start_state)]

    while priority_queue:
        f_cost, (robot, boxes) = heapq.heappop(priority_queue)

        if is_goal(boxes):
            return boxes

        if (robot, boxes) in visited:
            continue

        visited.add((robot, boxes))

        solution_board = [list(row) for row in board]
        for box in boxes:
            x, y = box
            solution_board[x][y] = 'B'
        solution_board[robot[0]][robot[1]] = 'R'
        ShowBoard(solution_board)

        for direction in directions:
            new_robot = apply_move(robot, direction)
            if not ValidMove(board, *new_robot) or board[new_robot[0]][new_robot[1]] == 'O':
                continue

            if (new_robot, boxes) not in visited:
                heapq.heappush(priority_queue, (f_cost + 1 + Heuristic(board, new_robot,
boxes, storage),
                                                (new_robot, boxes)))

            for box_index, box in enumerate(boxes):
                if box == new_robot:
                    new_box = apply_move(box, direction)
                    if valid_move(box, direction) and (
                    new_box, boxes[:box_index] + (new_box,) + boxes[box_index + 1:]) not in
visited:
                        heapq.heappush(priority_queue, (f_cost + 1 + Heuristic(board,
new_robot, boxes, storage),
                                                        (new_robot,
                                                         boxes[:box_index] + (new_box,) +
boxes[box_index + 1:])))
    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
```

```python
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()


result = Successor(board, robot, boxes, storage)

if result:
    print("End state can be found:")
    solution_board = [list(row) for row in board]
    for box in result:
        x, y = box
        solution_board[x][y] = 'B'
    print("\n".join(["".join(row) for row in solution_board]))

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

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

BFS States

Initial Board

```
OOOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBRR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O RO O
OOBO O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO ORO
OOSS O
OOOOOO
```

```
OOOOO
OBBR O
O BO O
OORO O
OOBS O
OOOOO

OOOOO
OBRR O
O BO O
OOBO O
OOSS O
OOOOO

OOOOO
OBBR O
ORBO O
OOBO O
OOSS O
OOOOO

OOOOO
OBBRRO
O BO O
OO O O
OOSS O
OOOOO

OOOOO
OBBR O
ORBO O
OO O O
OOSS O
OOOOO

OOOOO
O BR O
O BO O
OO O O
OOSSRO
OOOOO

OOOOO
OBBR O
O BO O
OO O O
OORS O
OOOOO

OOOOO
OBBR O
O RO O
OO O O
OOBS O
OOOOO

OOOOO
OBBR O
O BO O
OOBO O
```

```
OOSS O
OOOOOO

OOOOOO
ORBR O
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOBR O
OOOOOO

OOOOOO
OBRR O
O BO O
OO O O
OOBS O
OOOOOO

OOOOOO
OBBR O
ORBO O
OO O O
OOBS O
OOOOOO

OOOOOO
OBBRRO
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
```

```
OBBR O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOBSRO
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOBS O
OOOOOO

OOOOOO
ORBR O
O BO O
OO O O
OOBS O
OOOOOO

OOOOOO
OBBR O
O BORO
OOBO O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OORS O
```

```
OOOOO

OOOOO
OBBR O
O RO O
OO O O
OOSS O
OOOOO

OOOOO
O BR O
O BO O
OORO O
OOSS O
OOOOO

OOOOO
OBBR O
O BO O
OO ORO
OOBS O
OOOOO

OOOOO
OBBRRO
O BO O
OO O O
OOBS O
OOOOO

OOOOO
OBBR O
O BO O
OOBORO
OOSS O
OOOOO

OOOOO
OBBR O
O BO O
OO O O
OOSR O
OOOOO

OOOOO
OBBR O
O BO O
OORO O
OOSS O
OOOOO

OOOOO
OBRR O
O BO O
OO O O
OOSS O
OOOOO

OOOOO
OBBR O
```

```
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BORO
OO O O
OOBS O
OOOOOO

OOOOOO
OBBR O
O BO O
OOBO O
OOSSRO
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O RR O
O BO O
OO O O
OOSS O
OOOOOO
```

```
OOOOO
O BR O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OOBO O
OOSR O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
OBBRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBRR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBRB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBR O
O BO O
```

```
OO O O
OOSS O
OOOOOO


OOOOOO
OBBR O
O BO O
OOBO O
OORS O
OOOOOO


OOOOOO
OBBR O
O BO O
OO O O
OOSSRO
OOOOOO


OOOOOO
OBBR O
O BORO
OO O O
OOSS O
OOOOOO


OOOOOO
OBBR O
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBR O
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBB O
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBRBO
O BO O
OO O O
OOSS O
OOOOOO
```

```
OOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O RB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
OBBR O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBB O
O BO O
OORO O
```

```
OOSS O
OOOOOO


OOOOOO
OBBB O
ORBO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBRRO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBRRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BR O
O BO O
OO O O
OOSSRO
OOOOOO


OOOOOO
O BR O
O BORO
OO O O
OOSS O
OOOOOO


OOOOOO
O BB O
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
```

```
OBBR O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
OBBR O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
OBBB O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
OBBRBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
OBBRBO
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BB O
ORBO O
OO O O
OOSS O
```

```
OOOOO

OOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
OBRRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
O RB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
OBBR O
O BO O
OO O O
OOSR O
OOOOOO

OOOOO
OBBR O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOO
OBBB O
O BO O
OO O O
OOSR O
OOOOOO

OOOOO
OBBRBO
O BO O
OO ORO
OOSS O
OOOOOO

OOOOO
OBBRBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOO
OBBRBO
```

```
ORBO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BB O
O BO O
OO O O
OORS O
OOOOOO


OOOOOO
O BRBO
O BORO
OO O O
OOSS O
OOOOOO


OOOOOO
OBBRBO
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
ORBRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BB O
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOOO
```

```
OOOOO
OBBR O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
OBBB O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BRBO
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBRBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
OBBRBO
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBRRO
O BO O
```

```
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BB O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O RBBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBB O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSSRO
OOOOOO
```

```
OOOOOO
OBBRBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
OBBRBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BBBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BBBO
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBBBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBB O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO ORO
```

```
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
OBBRBO
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BBBO
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
O BBBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BBBO
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
```

```
O RRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBBRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BBBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BBBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BRBO
O BORO
OO O O
OOSS O
```

```
OOOOO

OOOOO
O BRBO
O RO O
OO O O
OOSS O
OOOOOO

OOOOO
ORBRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
O BBRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
O BRBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOO
O BB O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOO
O BBBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOO
O BRBO
O BO O
OO ORO
OOSS O
OOOOOO

OOOOO
O BRBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOO
O BRBO
```

```
ORBO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BB O
O BORO
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O BO O
OO O O
OOSSRO
OOOOOO


OOOOOO
O BRBO
O BO O
OO O O
OORS O
OOOOOO


OOOOOO
O RRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
ORBO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BBRO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O BO O
OO O O
OOSR O
OOOOOO
```

```
OOOOOO
ORBRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BBRO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BB O
O BORO
OO O O
OOSS O
OOOOOO


OOOOOO
O BB O
O BO O
OO ORO
OOSS O
OOOOOO


OOOOOO
O BB O
O BO O
OO O O
OOSSRO
OOOOOO


OOOOOO
O BB O
O BO O
OO O O
OOSR O
OOOOOO


OOOOOO
O BB O
O BO O
OO O O
OORS O
OOOOOO


OOOOOO
O BB O
O BO O
```

```
OORO O
OOSS O
OOOOOO

OOOOOO
O BB O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O RB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO ORO
OOSS O
OOOOOO
```

```
OOOOO
O BRBO
O BO O
OO O O
OOSSRO
OOOOO


OOOOO
O BRBO
O BO O
OO O O
OOSR O
OOOOO


OOOOO
O BRBO
O BO O
OO O O
OORS O
OOOOO


OOOOO
O BRBO
O BO O
OORO O
OOSS O
OOOOO


OOOOO
O BRBO
O RO O
OO O O
OOSS O
OOOOO


OOOOO
O RRBO
O BO O
OO O O
OOSS O
OOOOO


OOOOO
O BRBO
ORBO O
OO O O
OOSS O
OOOOO


OOOOO
ORBRBO
O BO O
OO O O
OOSS O
OOOOO


OOOOO
O RBBO
O BO O
OO O O
```

```
OOSS O
OOOOOO

OOOOOO
O BBBO
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBBBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BBBO
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BBBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BBBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BBBO
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
```

```
O BBBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BBRO
O BO O
OO O O
OOSS O
OOOOOO
```

End State cannot be found.
Total execution time is 0.0020017623901367188 secs

GBS States
Initial Board

```
OOOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O RR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
```

```
O BR O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSSRO
OOOOOO
```

End state cannot be found.
Total execution time is 0.0 secs.

AStar States

Initial Board

Pukoban Using A Star Algorithm
OOOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBRR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O RR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O RO O
OOBO O
OOSS O
OOOOOO

```
OOOOOO
OBBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O RO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O BR O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O RR O
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O RB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
ORBO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
```

```
OORO O
OOBS O
OOOOOO

OOOOOO
OBBR O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
OBRR O
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
OBBR O
ORBO O
OOBO O
OOSS O
OOOOOO

OOOOOO
ORBR O
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OORO O
OOSS O
OOOOOO
```

```
OOOOO
OBBR O
O BO O
OORO O
OOBS O
OOOOOO

OOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
OBBRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
O BR O
O RO O
OO O O
OOBS O
OOOOOO

OOOOO
OBBR O
O BO O
OORO O
OOSS O
OOOOOO

OOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
O BB O
O RO O
OO O O
OOSS O
OOOOOO

OOOOO
O BB O
O RO O
OOBO O
```

```
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
ORBR O
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O BR O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OOBO O
OORS O
OOOOOO

OOOOOO
O RR O
O BO O
OO O O
OOBS O
OOOOOO

OOOOOO
O RB O
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
```

```
O BR  O
O BO  O
OO O  O
OORS  O
OOOOOO

OOOOOO
O BRRO
O BO  O
OOBO  O
OOSS  O
OOOOOO

OOOOOO
O BR  O
ORBO  O
OO O  O
OOBS  O
OOOOOO

OOOOOO
OBBR  O
O RO  O
OO O  O
OOBS  O
OOOOOO

OOOOOO
O BBRO
O BO  O
OO O  O
OOSS  O
OOOOOO

OOOOOO
O BRRO
O BO  O
OO O  O
OOSS  O
OOOOOO

OOOOOO
OBBR  O
O BO  O
OO O  O
OORS  O
OOOOOO

OOOOOO
ORBR  O
O BO  O
OO O  O
OOBS  O
OOOOOO

OOOOOO
O RRBO
O BO  O
OO O  O
OOSS  O
```

```
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOBS O
OOOOOO

OOOOOO
O BR O
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
OBBR O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OOBO O
OORS O
OOOOOO

OOOOOO
O RR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
ORBO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BB O
```

```
O BO  O
OORO  O
OOBS  O
OOOOOO

OOOOOO
O BR  O
O BO  O
OO O  O
OOBR  O
OOOOOO

OOOOOO
ORBB  O
O BO  O
OOBO  O
OOSS  O
OOOOOO

OOOOOO
O BB  O
ORBO  O
OO O  O
OOSS  O
OOOOOO

OOOOOO
OBBR  O
O BORO
OO O  O
OOSS  O
OOOOOO

OOOOOO
O BB  O
O BO  O
OORO  O
OOSS  O
OOOOOO

OOOOOO
OBBR  O
O BO  O
OO O  O
OORS  O
OOOOOO

OOOOOO
OBRR  O
O BO  O
OO O  O
OOBS  O
OOOOOO

OOOOOO
OBBR  O
ORBO  O
OO O  O
OOBS  O
OOOOOO
```

```
OOOOOO
O BR O
ORBO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BR O
O BO O
OORO O
OOSS O
OOOOOO


OOOOOO
O BR O
O BO O
OOBO O
OOSR O
OOOOOO


OOOOOO
O RB O
O BO O
OO O O
OOBS O
OOOOOO


OOOOOO
O BR O
O BO O
OO O O
OOSSRO
OOOOOO


OOOOOO
OBBRRO
O BO O
OOBO O
OOSS O
OOOOOO


OOOOOO
O BR O
O BORO
OOBO O
OOSS O
OOOOOO


OOOOOO
O BRRO
O BO O
OO O O
OOBS O
OOOOOO


OOOOOO
O BBRO
O BO O
```

```
OOBO O
OOSS O
OOOOOO

OOOOOO
O BRRO
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O BB O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BB O
O RO O
OO O O
OOBS O
OOOOOO

OOOOOO
ORBR O
O BO O
OO O O
OOBS O
OOOOOO

OOOOOO
O RRBO
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOBS O
OOOOOO
```

```
OOOOO
O BRBO
O BO O
OO O O
OOBS O
OOOOOO

OOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOO
OBRR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOBR O
OOOOOO

OOOOOO
ORBB O
O BO O
OO O O
OOBS O
OOOOOO

OOOOOO
ORBRBO
O BO O
OO O O
```

```
OOSS O
OOOOOO


OOOOOO
OBBR O
O BO O
OO O O
OOBS O
OOOOOO


OOOOOO
O BB O
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O RO O
OOBO O
OOSS O
OOOOOO


OOOOOO
O BB O
O BO O
OOBO O
OORS O
OOOOOO


OOOOOO
O BR O
O BO O
OO O O
OOBSRO
OOOOOO


OOOOOO
OBBR O
O BO O
OORO O
OOSS O
OOOOOO


OOOOOO
OBBR O
O BO O
OOBO O
OOSR O
OOOOOO


OOOOOO
```

```
O BB  O
O BO  O
OO O  O
OORS  O
OOOOOO

OOOOOO
O BBRO
O BO  O
OO O  O
OOBS  O
OOOOOO

OOOOOO
O BRRO
O BO  O
OO O  O
OOBS  O
OOOOOO

OOOOOO
O BR  O
O BORO
OO O  O
OOBS  O
OOOOOO

OOOOOO
O BB  O
O BORO
OOBO  O
OOSS  O
OOOOOO

OOOOOO
O BRBO
O BORO
OOBO  O
OOSS  O
OOOOOO

OOOOOO
O BR  O
O BO  O
OO O  O
OORS  O
OOOOOO

OOOOOO
O BR  O
O BO  O
OOBO  O
OOSSRO
OOOOOO

OOOOOO
O BB  O
ORBO  O
OO O  O
OOBS  O
```

```
OOOOO

OOOOO
OBBR O
O BO O
OO ORO
OOSS O
OOOOO

OOOOO
O BR O
O BO O
OOBORO
OOSS O
OOOOO

OOOOO
OBBR O
O BO O
OO O O
OOSR O
OOOOO

OOOOO
O RRBO
O BO O
OO O O
OOBS O
OOOOO

OOOOO
OBBR O
O BORO
OOBO O
OOSS O
OOOOO

OOOOO
O BB O
O BO O
OO ORO
OOSS O
OOOOO

OOOOO
O BRBO
O BO O
OO ORO
OOSS O
OOOOO

OOOOO
OBRR O
O BO O
OO O O
OOSS O
OOOOO

OOOOO
O RB O
```

```
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O RB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOBR O
OOOOOO

OOOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBRBO
O BO O
OOBO O
OOSS O
OOOOOO

OOOOOO
OBBRRO
O BO O
OO O O
OOBS O
OOOOOO

OOOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO ORO
OOBS O
OOOOOO

OOOOOO
O BB O
O BO O
OOBO O
OOSR O
OOOOOO
```

```
OOOOO
OBBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BORO
OO O O
OOBS O
OOOOOO

OOOOOO
O BRBO
O BORO
OO O O
OOBS O
OOOOOO

OOOOOO
O BB O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
ORBO O
OOBO O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OORO O
OOBS O
OOOOOO

OOOOOO
OBBR O
O BO O
```

```
OO O O
OOBSRO
OOOOOO

OOOOOO
O BRBO
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OOBORO
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OOBORO
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
ORBRBO
O BO O
OO O O
OOBS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO
```

```
OOOOOO
O BRBO
O RO O
OO O O
OOBS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
OBBR O
O BO O
OOBO O
OOSSRO
OOOOOO

OOOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
```

```
OOBSRO
OOOOOO

OOOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
OBRB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OOBORO
OOSS O
OOOOOO

OOOOOO
```

```
O BB O
O BO O
OO ORO
OOBS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO ORO
OOBS O
OOOOOO

OOOOOO
OBBR O
O BORO
OO O O
OOBS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BB O
O BO O
OOBO O
OOSSRO
OOOOOO

OOOOOO
OBRR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
OBBRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO ORO
OOBS O
```

```
OOOOO

OOOOO
O BRBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BRBO
O BO O
OOBO O
OOSSRO
OOOOOO

OOOOOO
OBRB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O RB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BRBO
```

```
O BO O
OOBO O
OORS O
OOOOOO

OOOOOO
OBRRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O RRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
ORBO O
OO O O
OOBS O
OOOOOO

OOOOOO
OBRRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O RRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OORO O
OOSS O
OOOOOO
```

```
OOOOOO
O BRBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOBSRO
OOOOOO

OOOOOO
OBBRRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBRBO
O BO O
```

```
OO O O
OOSS O
OOOOOO


OOOOOO
OBBB O
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBR O
O BO O
OO O O
OOSR O
OOOOOO


OOOOOO
O BRBO
O BO O
OO O O
OOSR O
OOOOOO


OOOOOO
O BR O
O BO O
OO O O
OOSSRO
OOOOOO


OOOOOO
ORBR O
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BR O
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO
```

```
OOOOOO
OBBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BR O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOBR O
OOOOOO

OOOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BORO
OO O O
```

```
OOSS O
OOOOOO


OOOOOO
O BRBO
O BORO
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O BO O
OOBO O
OOSR O
OOOOOO


OOOOOO
ORBRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
ORBRBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
ORBB O
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBB O
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BB O
O BO O
OO O O
OOSR O
OOOOOO


OOOOOO
```

```
OBBR O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
OBBB O
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBRBO
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
ORBRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
ORBRBO
O BO O
OO O O
OOSS O
```

```
OOOOO

OOOOO
OBBRBO
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BBRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O RBBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBBRO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBRO
```

```
O BO  O
OO O  O
OOSS  O
OOOOOO


OOOOOO
OBBR  O
ORBO  O
OO O  O
OOSS  O
OOOOOO


OOOOOO
OBBB  O
O BO  O
OORO  O
OOSS  O
OOOOOO


OOOOOO
O RBBO
O BO  O
OO O  O
OOSS  O
OOOOOO


OOOOOO
OBBRRO
O BO  O
OO O  O
OOSS  O
OOOOOO


OOOOOO
OBBBRO
O BO  O
OO O  O
OOSS  O
OOOOOO


OOOOOO
OBBRRO
O BO  O
OO O  O
OOSS  O
OOOOOO


OOOOOO
O BB  O
ORBO  O
OO O  O
OOSS  O
OOOOOO


OOOOOO
O BB  O
O BO  O
OORO  O
OOSS  O
OOOOOO
```

```
OOOOO
O BB O
O BO O
OO ORO
OOSS O
OOOOOO


OOOOOO
O BRBO
O BO O
OO ORO
OOSS O
OOOOOO


OOOOOO
OBBR O
O BO O
OO O O
OOSSRO
OOOOOO


OOOOOO
OBBRRO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBRBO
ORBO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BRBO
ORBO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBR O
O BO O
OORO O
OOSS O
OOOOOO


OOOOOO
O BRBO
O BO O
OORO O
OOSS O
OOOOOO


OOOOOO
OBBB O
ORBO O
```

```
OO O O
OOSS O
OOOOOO

OOOOOO
OBBB O
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BB O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BBBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
OBBRBO
O BO O
OORO O
OOSS O
OOOOOO
```

```
OOOOOO
OBBR O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
ORBBBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBBO
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBRBO
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBB O
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
OBBRBO
O BORO
OO O O
```

```
OOSS O
OOOOOO


OOOOOO
OBBRBO
O BO O
OORO O
OOSS O
OOOOOO


OOOOOO
OBBR O
O BO O
OO ORO
OOSS O
OOOOOO


OOOOOO
OBBR O
O BO O
OO O O
OOSR O
OOOOOO


OOOOOO
ORBBBO
O BO O
OO O O
OOSS O
OOOOOO


OOOOOO
O BBBO
O RO O
OO O O
OOSS O
OOOOOO


OOOOOO
OBBB O
O BORO
OO O O
OOSS O
OOOOOO


OOOOOO
OBBB O
O BO O
OO O O
OORS O
OOOOOO


OOOOOO
OBBRBO
O BORO
OO O O
OOSS O
OOOOOO


OOOOOO
```

```
O BB  O
O BO  O
OO  O O
OORS  O
OOOOOO

OOOOOO
O BRBO
O BO  O
OO  O O
OOSSRO
OOOOOO

OOOOOO
O BRBO
O BO  O
OO  O O
OORS  O
OOOOOO

OOOOOO
O BRRO
O BO  O
OO  O O
OOSS  O
OOOOOO

OOOOOO
OBBR  O
O BORO
OO  O O
OOSS  O
OOOOOO

OOOOOO
OBBR  O
O BO  O
OO  O O
OORS  O
OOOOOO

OOOOOO
OBBB  O
O BO  O
OO  O O
OORS  O
OOOOOO

OOOOOO
O BB  O
O BO  O
OO ORO
OOSS  O
OOOOOO

OOOOOO
O BBBO
O BO  O
OO ORO
OOSS  O
```

```
OOOOO

OOOOO
O RRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BBBO
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BBBO
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
OBBB O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBRBO
```

```
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BBBO
ORBO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BBBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
OBBB O
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
OBBB O
O BO O
OO O O
OOSR O
OOOOOO
```

```
OOOOO
O BRBO
O BORO
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
OBBRBO
O BO O
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBB O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BB O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
O BBBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
ORBRBO
O BO O
OO O O
OOSS O
OOOOOO

OOOOOO
O BRBO
O RO O
OO O O
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
```

```
OO ORO
OOSS O
OOOOOO

OOOOOO
OBBR O
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BBBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BBBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
OBBB O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOOO
OBBRBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOOO
O BBBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO ORO
OOSS O
OOOOOO
```

```
OOOOO
OBBRBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOO
OBBB O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOO
O BBBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOO
OBBRBO
O BO O
OO O O
OOSSRO
OOOOOO

OOOOO
O BRBO
ORBO O
OO O O
OOSS O
OOOOOO

OOOOO
O BRBO
O BO O
OORO O
OOSS O
OOOOOO

OOOOO
O BBBO
O BO O
OO O O
OOSR O
OOOOOO

OOOOO
OBBR O
O BO O
OO O O
OOSSRO
OOOOOO

OOOOO
O BRBO
O BO O
OO O O
```

```
OOSSRO
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OORS O
OOOOOO

OOOOOO
O BRBO
O BO O
OO O O
OOSR O
OOOOOO

End state cannot be found:
Total execution time is 0.003007173538208008 secs
```