1. Start

/ \

(3,3) (0,0)

/ \ / \

(2,3) (3,2) (0,1) (1,0)

/ \ / \ / \ / \

(2,2) (3,1)(1,2)(2,1)(1,1)(0,2)

| | | | | |

End End End End End

2. Menggunakan depth-first search (DFS)

class Node:

def \_\_init\_\_(self, state, parent, boat\_pos, depth, cost):

self.state = state

self.parent = parent

self.boat\_pos = boat\_pos

self.depth = depth

self.cost = cost

def is\_valid\_state(state):

m, c = state

if m < 0 or c < 0 or m > 3 or c > 3:

return False

if m > 0 and m < c:

return False

if m < 3 and m > c:

return False

return True

def goal\_test(state):

return state == (0, 0)

def successors(state, boat\_pos):

successors = []

if boat\_pos == 'left':

new\_state = (state[0] - 2, state[1])

if is\_valid\_state(new\_state):

successors.append(Node(new\_state, state, 'right', state.depth + 1, 2))

new\_state = (state[0] - 1, state[1] - 1)

if is\_valid\_state(new\_state):

successors.append(Node(new\_state, state, 'right', state.depth + 1, 2))

new\_state = (state[0] - 1, state[1])

if is\_valid\_state(new\_state):

successors.append(Node(new\_state, state, 'right', state.depth + 1, 1))

new\_state = (state[0], state[1] - 2)

if is\_valid\_state(new\_state):

successors.append(Node(new\_state, state, 'right', state.depth + 1, 2))

new\_state = (state[0], state[1] - 1)

if is\_valid\_state(new\_state):

successors.append(Node(new\_state, state, 'right', state.depth + 1, 1))

else:

new\_state = (state[0] + 2, state[1])

if is\_valid\_state(new\_state):

successors.append(Node(new\_state, state, 'left', state.depth + 1, 2))

new\_state = (state[0] + 1, state[1] + 1)

if is\_valid\_state(new\_state):

successors.append(Node(new\_state, state, 'left', state.depth + 1, 2))

new\_state = (state[0] + 1, state[1])

if is\_valid\_state(new\_state):

successors.append(Node(new\_state, state, 'left', state.depth + 1, 1))

new\_state = (state[0], state[1] + 2)

if is\_valid\_state(new\_state):

successors.append(Node(new\_state, state, 'left', state.depth + 1, 2))

new\_state = (state[0], state[1] + 1)

if is\_valid\_state(new\_state):

successors.

3. Tidak seperti teka-teki lainnya, tidak ada strategi jelas yang berfungsi di semua situasi, sehingga menyulitkan pemain untuk menemukan pendekatan yang tepat. Pemain mungkin harus mencoba pendekatan yang berbeda dan membalikkan gerakan mereka beberapa kali sebelum menemukan solusi yang tepat.