**MACHINARIES AND CANNIBALS**

from collections import deque

def is\_valid(state, total\_missionaries, total\_cannibals):

lM, lC, rM, rC = state

if lM < 0 or lC < 0 or rM < 0 or rC < 0:

return False

if lM > total\_missionaries or lC > total\_cannibals or rM > total\_missionaries or rC > total\_cannibals:

return False

if (lM > 0 and lC > lM) or (rM > 0 and rC > rM):

return False

return True

def is\_goal(state, total\_missionaries, total\_cannibals):

\_, \_, rM, rC = state

return rM == total\_missionaries and rC == total\_cannibals

def get\_next\_states(state, boat\_on\_left, total\_missionaries, total\_cannibals, boat\_capacity):

lM, lC, rM, rC = state

next\_states = []

for m in range(0, boat\_capacity + 1):

for c in range(0, boat\_capacity + 1):

if m + c > 0 and m + c <= boat\_capacity:

if boat\_on\_left:

if lM >= m and lC >= c:

next\_states.append((lM - m, lC - c, rM + m, rC + c))

else:

if rM >= m and rC >= c:

next\_states.append((lM + m, lC + c, rM - m, rC - c))

return [state for state in next\_states if is\_valid(state, total\_missionaries, total\_cannibals)]

def solve\_missionaries\_and\_cannibals(total\_missionaries, total\_cannibals, boat\_capacity):

initial\_state = (total\_missionaries, total\_cannibals, 0, 0)

boat\_on\_left = True

queue = deque([(initial\_state, boat\_on\_left, [])])

visited = set()

while queue:

current\_state, boat\_on\_left, path = queue.popleft()

if is\_goal(current\_state, total\_missionaries, total\_cannibals):

return path + [(current\_state, boat\_on\_left)]

if (current\_state, boat\_on\_left) in visited:

continue

visited.add((current\_state, boat\_on\_left))

next\_states = get\_next\_states(current\_state, boat\_on\_left, total\_missionaries, total\_cannibals, boat\_capacity)

for next\_state in next\_states:

queue.append((next\_state, not boat\_on\_left, path + [(current\_state, boat\_on\_left)]))

return None

def describe\_step(current\_state, next\_state, boat\_on\_left):

cM, cC, \_, \_ = current\_state

nM, nC, \_, \_ = next\_state

missionaries\_moved = abs(cM - nM)

cannibals\_moved = abs(cC - nC)

if boat\_on\_left:

direction = "from left to right"

else:

direction = "from right to left"

description = f"Moved {missionaries\_moved} missionary(ies) and {cannibals\_moved} cannibal(s) {direction}."

return description

def print\_solution(total\_missionaries, total\_cannibals, boat\_capacity):

solution = solve\_missionaries\_and\_cannibals(total\_missionaries, total\_cannibals, boat\_capacity)

if solution:

print(f"Initial state:\nLeft: M:{total\_missionaries}, C:{total\_cannibals}, B:1 | Right: M:0, C:0, B:0")

print(f"Solution path for {total\_missionaries} Missionaries, {total\_cannibals} Cannibals, and Boat capacity {boat\_capacity}:\n")

for i in range(len(solution) - 1):

current\_state, boat\_on\_left = solution[i]

next\_state, \_ = solution[i + 1]

lM, lC, rM, rC = current\_state

boat\_status\_left = "B:1" if boat\_on\_left else "B:0"

boat\_status\_right = "B:0" if boat\_on\_left else "B:1"

print(f"Left: M:{lM}, C:{lC}, {boat\_status\_left} | Right: M:{rM}, C:{rC}, {boat\_status\_right}")

description = describe\_step(current\_state, next\_state, boat\_on\_left)

print(f"Step {i + 1}: {description}\n")

final\_state, boat\_on\_left = solution[-1]

lM, lC, rM, rC = final\_state

boat\_status\_left = "B:1" if boat\_on\_left else "B:0"

boat\_status\_right = "B:0" if boat\_on\_left else "B:1"

print(f"Left: M:{lM}, C:{lC}, {boat\_status\_left} | Right: M:{rM}, C:{rC}, {boat\_status\_right}")

print("\nSolution found")

else:

print("No solution found.")

def get\_positive\_integer(prompt):

while True:

try:

value = int(input(prompt))

if value <= 0:

print("Invalid input. Please enter a positive integer.")

else:

return value

except ValueError:

print("Invalid input. Please enter a valid number.")

if \_\_name\_\_ == "\_\_main\_\_":

missionaries = get\_positive\_integer("Enter the number of missionaries: ")

cannibals = get\_positive\_integer("Enter the number of cannibals: ")

boat\_capacity = get\_positive\_integer("Enter the boat capacity: ")

print\_solution(missionaries, cannibals, boat\_capacity)