NANCY T THOMAS
1947244

PROBLEM STATEMENT:

Write a program to implement BFS (for 8 puzzle problem or Water Jug problem or any AI search problem

EXPLANATION:

Here the initial state is (0, 0). The goal state is (2, n) for any value of n.To solve this we have to make some assumptions not mentioned in the problem. They are:

- We can fill a jug from the pump.
- We can pour water out of a jug to the ground.
- We can pour water from one jug to another.
- There is no measuring device available.

WATER JUG PROBLEM

```
from collections import deque

def BFS(a, b, target):

  # Map is used to store the states, every
  # state is hashed to binary value to
  # indicate either that state is visited
  # before or not
  m = {}
  isSolvable = False
```

```python
path = []

# Queue to maintain states
q = deque()

# Initialing with initial state
q.append((0, 0))

while (len(q) > 0):

  # Current state
  u = q.popleft()

  #q.pop() #pop off used state

  # If this state is already visited
  if ((u[0], u[1]) in m):
    continue

  # Doesn't met jug constraints
  if ((u[0] > a or u[1] > b or
    u[0] < 0 or u[1] < 0)):
    continue

  # Filling the vector for constructing
  # the solution path
  path.append([u[0], u[1]])

  # Marking current state as visited
  m[(u[0], u[1])] = 1

  # If we reach solution state, put ans=1
  if (u[0] == target or u[1] == target):
    isSolvable = True

    if (u[0] == target):
      if (u[1] != 0):

        # Fill final state
        path.append([u[0], 0])
```

```python
    else:
      if (u[0] != 0):

        # Fill final state
        path.append([0, u[1]])

    # Print the solution path
    sz = len(path)
    for i in range(sz):
      print("(", path[i][0], ",",
          path[i][1], ")")
    break

  # If we have not reached final state
  # then, start developing intermediate
  # states to reach solution state
  q.append([u[0], b]) # Fill Jug2
  q.append([a, u[1]]) # Fill Jug1

  for ap in range(max(a, b) + 1):

    # Pour amount ap from Jug2 to Jug1
    c = u[0] + ap
    d = u[1] - ap

    # Check if this state is possible or not
    if (c == a or (d == 0 and d >= 0)):
      q.append([c, d])

    # Pour amount ap from Jug 1 to Jug2
    c = u[0] - ap
    d = u[1] + ap

    # Check if this state is possible or not
    if ((c == 0 and c >= 0) or d == b):
      q.append([c, d])

  # Empty Jug2
  q.append([a, 0])
```

```
    # Empty Jug1
    q.append([0, b])

  # No, solution exists if ans=0
  if (not isSolvable):
    print ("No solution")

# Driver code
if __name__ == '__main__':

  Jug1=input("Enter Jug1 capacity\t");
  Jug1 = int(Jug1);
  Jug2=input("Enter Jug2 capacity\t");
  Jug2 = int(Jug2);

  target=input("Enter target jug capacity\t");
  target = int(target);

  print("Path from initial state to solution state ::")

  BFS(Jug1, Jug2, target)
```