```
from collections import deque
def Solution(a, b, target):
 m = {}
 isSolvable = False
 path = []

 q = deque()

 #Initializing with jugs being empty
 q.append((0, 0))

 while (len(q) > 0):

  # Current state
  u = q.popleft()
  if ((u[0], u[1]) in m):
   continue
  if ((u[0] > a or u[1] > b or
   u[0] < 0 or u[1] < 0)):
   continue
  path.append([u[0], u[1]])

  m[(u[0], u[1])] = 1

  if (u[0] == target or u[1] == target):
   isSolvable = True

   if (u[0] == target):
    if (u[1] != 0):
     path.append([u[0], 0])
   else:
    if (u[0] != 0):

     path.append([0, u[1]])

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

  q.append([u[0], b]) # Fill Jug2
  q.append([a, u[1]]) # Fill Jug1

  for ap in range(max(a, b) + 1):
   c = u[0] + ap
   d = u[1] - ap

   if (c == a or (d == 0 and d >= 0)):
    q.append([c, d])

   c = u[0] - ap
   d = u[1] + ap

   if ((c == 0 and c >= 0) or d == b):
```

```
    q.append([c, d])

   q.append([a, 0])

   q.append([0, b])

  if (not isSolvable):
   print("Solution not possible")

if __name__ == '__main__':

  Jug1, Jug2, target = 4, 3, 2
  print("Path from initial state "
   "to solution state ::")

  Solution(Jug1, Jug2, target)
```