Date : 13-11-2020

Name : Rudramuni G.m
USN : 1BM18CS085

AI LABTEST-1

(2)

```
 def main():
   """
 starting_node=[[0,0]]
  jugs=get_jugs()
  goal_amount=get_goal(jugs)
  check_dict={}
  is_depth = get_search_type()
search(starting_node, jugs, goal_amount, check_dict, is_depth)

def get_index(node):
  """
 return pow(7, node[0], + pow(5, node[1]))

def get_search_type():
   """
  s=input("Enter 'b' for BFS, 'd' for DFS: ")
  s=s[0].lower()
 while s!= 'd' and s!='b':
  s = input ("The input is not valid. Enter 'b' for BFS
                d for DFS: ")
   s = s[0].lower()
  return s=='d'

def get_jugs():
   """
 print(" Reciving volume of jugs:")
    jugs=[]
  temp= int(input("Enter first jug volume(>1): "))
  while temp<1:
    temp=int(input("Enter valid amount(>1): "))
     jugs.append(temp)
```

```
temb = int(input("Enter Second volume(>1):"))
 while temb <1:
   temb= int (intput ("Enter valid amount(>1):"))
    jugs.append(temb)
    return jugs

def get_goal(jugs):
 print("Receiving desired amt of water")
 max_amount = max(jugs[0], jugs[1])
 S = "Enter desired amt of water(1 - {0}):".
      format(max_amount)
 goal_amount = int(input("Enter valid amount
          (1-{0}): ".format(max_amount)))
  return goal_amount

 def is_goal(path, goal_amount):
   print("Checking if goal achieved")
return path[-1][0] == goal_amount or path[-1][1]==goal_amt

def been_there(node, check_dict):
  print("Checking if {0} is visited befor".formate(node))
   return check_dict.get(get_index(node), False)

def next_transitions(jugs, path, check_dict):
print("Finding Next transitions & cheking for loops")
  result = []
   next_nodes = []
   node = []
   a_max = jugs[0]
   b_max = jugs[1]
   a = path[-1][0]
   b = path[-1][1]
```

```
node.append (a)
 node.append (b-max)
 it not been-there(node, check-dict):
    next-nodes.append (node)
   node=[]

node.append (min (amax, a+b)
node.append (b -(node[0]-a)) # b-(a'-a)
 it nod beethere (node, check-dict);
   next-nods-append (node)
    node=[]
  same too 4,5,6 jugs.
 too i in range (0, len(next-nodes)):
     temp= list(Path)
     temp.append(next-nodes[i])
      result.append(temb).

 it len (next-nodes) ==0
   Point ("No more unvisited nodes ")
 else
   point (" possible transitions:")
    too node in next-nodes:
      Point (node)
    return result

 det transition (old, new, jugs):
   a= old[0]
   b = old[1]
   a-Prime= new[0]
   b-Prime= new[1]
   amax= jugs[0]
   b-max = jags[1]
```

```
    if a>a_prime:
     if b==b_prime:
      return Clear {0}. format(a_max)
    else:
   return "pour {0} litl jug into {1}-jug:
  else
  if b>b_prime:
   return "Clear {0}-liter jug: . format(a_max)
   elif a==a_prime:
        return "Clear {0}\t\t\t", format(b_max).
    else:
     return (pour {0}_litter jug into{1}. format
                          (b_max, a_max).
   else
   if a==a_prime:
     return " Fill {0} -litterjug:\t\t\t", format(b_ma
   else:
 return "Fill {0}-liter jug: \t\t\t". format(a_max,
def search (startingnode, jugs, goal amount, checkdi
               is_depth).
  if is_depth:
   print("Implementing DFS...")
   else
   print (" Impu. BFS---")
   goal =[]
   accomplished = false.
   q = collection.deque()
    q. appendleft (starting_node)
```

07

Ru.

1BM18CS085

```
    while (len(q))! =0,
      Path= q.popleft()
       check-dict[get-index(path[-1])]=True.
     if len(path) >=2:
  Print(transition(path[-2], path[-1], jugs] path[-1]))
      accomplished = True.
       goal = path
        break
  next_moves = next_transitions (jugs, path, checkdict)
  for i   in next_moves:
    if is_depth:
       q.appendleft(i)
      else.
       q.append(i)
    if accomplished:
     Print ("achieved)
      Print_path (goal, jugs)
     else
       print(Prob  cant  solved")
     if _name_ == '_main_'.
       main()
```