13-11-2020

AI Lab-1

Yashasvi.R
1BM18CS154
5th Sem, B sec

Question-2

```
import collections
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):
    s = input("
    return pow(7, node[0]) * pow(5, node[1])

def get_search_type():
    return s == 'd'

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


Yashasvi Ravikumar

Yashaswi. R
1BM18CS154
5th Sem, B sec

```
def get_goal(jugs):
    print("Receiving desired amount of water")
    max_amount = max(jugs[0], jugs[1])
    s = "Enter the desired amount of water (1-{0}):".format
        (max_amount)
    return goal_amount.

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

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

def next_transitions(jugs, path, check_dict):
    print("Finding next transitions and checking for loops")
    result=[]
    next_nodes = []
    node = []

    a_max = jugs[0]
    b_max = jugs[1]

    a = path[-1][0]
    b = path[-1][1]
```


Yashaswi.Rajkumar

Yashaswi.R
1BM18CS154
5th Sem, B Sec

```
node.append(a_max)
node.append(b)
if not been_there(node, check_dict):
    next_nodes.append(node)
node = []

node.append(a)
node.append(b_max)
if not been_there(node, check_dict):
    next_nodes.append(node)
node = []

node.append(min(a_max, a+b))
node.appened(b-(node[0]-a))
if not been_there(node, check_dict):
    next_nodes.append(node)
node = []

node.append(min(a+b, b_max))
node.insert(0, a-(node[0]-b))
if not been_there(node, check_dict):
    next_nodes.append(node)
node = []

node.append(0)
node.append(b)
if not been_there(node, check_dict):
    next_nodes.append(node)
node = []
```


Yashaswi.Ravikumar

Yashaswi.R
1BM18CS154
5th Sem, B sec

```
node.append(a)
node.append(0)
if(not been_there(node, check_dict)):
    next_nodes.append(node)
for i in range(0, len(next_nodes)):
    temp = list(path)
    temp.append(next_nodes[i])
    result.append(temp)
if len(next_nodes)==0:
    print("No more unvisited nodes")
else
    print("Possible")
    for node in next_nodes:
        print(node)
return result

def transaction(old, new, jugs):
    a = old[0]
    b = old[1]
    a_prime = new[0]
    b_prime = new[1]
    a_max = jugs[0]
    b_max = jugs[1]
```


Yashaswi.Pavikumar

```
if a > a-prime:
    if b == b-prime:
        return " Clear {0}-liters jug ".format(a_max)
    else return " Pour {0}-liter jug into {1}-liters jug ".
                                                format
                                                (a-max, b-max
else
    if b > b-prime
        if a == a-prime:
            return' Clear {0}L jug ".format(b-max)
        else return " Clear {0}L to {1}L jug ".format(b-max, a-max)
    else
        if a == a-prime
            return "Fill {0}L jug :".format(b-max)
        else return 'Fill {0}-liters jug ".format(a-max)

def print_path(path, jugs):
    print(" Start from", path[0]
    for i in range (0, len(path)-1)
        print(i+1,":", transition(path[i], path[i+1], jugs)
                                                path[i+1]

def search (starting_node, jugs, goal_amount, check_dict, is_depth):
    if is== print("DFS")
        goal = []
        accomplished = False
```

AI lab-1

Yashaswi R
1BM18CS154
5th Sem, B sec

```
q = collections.deque()
q.appendleft(starting_node)
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])
    if is_goal(path, goal_amount):
        accomplished = True
        goal = path
        break
    next_moves = next_transitions(jugs, path, check_dict)
    for i in next_moves:
        if is_depth:
            q.appendleft(i)
        else
            q.append(i)

    if accomplished:
        print("The goal is achived , moves in")
        print_path(goal, jugs)
    else:
        print("Cannot be Solved")

if __name__ == '__main__':
    main()
```


Yashaswi.Ravikumar