PROJECT REPORT

SOLVING MAP COLORING PROBLEM USING CONSTRAINT SATISFACTION PROBLEM

INTELLIGENT SYSTEMS
PROJECT- 3

SUBMITTED TO: Prof. Dewan T Ahmed

Done by:

1. **Ramanathan Sivaramakrishnan(801243565)**
2. **Siddharth Jayachandra Babu(801252829)**
3. **Sneha Srikanth(801252592)**

Map Coloring using Constraint Satisfaction Problem:

Problem Statement:

The maps of Australia and The United States of America are respectively taken and the constraints are:

1. Only 5 colors are to be used and they are Red,Green,Blue, Black respectively.
2. No adjacent regions can be colored using the same color.

We have to solve this map coloring problem using the constraint satisfaction algorithm.

Introduction:

What is the Constraint Satisfaction Problem?
When we should take many decision simultaneously, we should model those decision using variable. So, in a Constraint Satisfaction Problem, there are many set of variables such as X1, X2,.....,Xn and all of these variables have a number of choices to make and they use the non-empty domain.

For example,
X1 = { 1,2,3,4}
So, this means X1 has values of 1,2,3,4 and this is called the domain.
Constraints may also be rules that does not allow you to assign other values.
Eg: X3 < X4
Solution: We should find values for all variables such that the constraints are satisfied. Or in other words, we say the constraints should not be broken.
{D1,D2…..Dn} - This is the set of domains for each X, where X is a variable.
C is the set of constraints
Ci= <scope, relation>
Here, the “scope” tells what variables are involved and “relation” is the relationship between those values.

GRAPH COLORING:

Let us assume a graph G, where G=(V,E) where V is the vertices and E is the edge and has an integer k. We should make sure that none of the adjacent vertices must have the same color. So a k-coloring of G which is an one to one mapping is used. To find the least value of k that is the chromatic number of G which can be also denoted by X(G) or X. We should the Minimum Graph Coloring Problem algorithm. The size of maximum value of G has this chromatic number within it, where the chromatic number of least number of color needed to satisfy the constraint of nearby region

MAP COLORING:

The map of Australia and USA is used for this problem and our main goal is to satisfy the constraints that is, 4 colors are only supposed to be used and they are Blue, Green, Red and black. The adjacent regions should also not be the same. If this happens, the constraint will be broken.

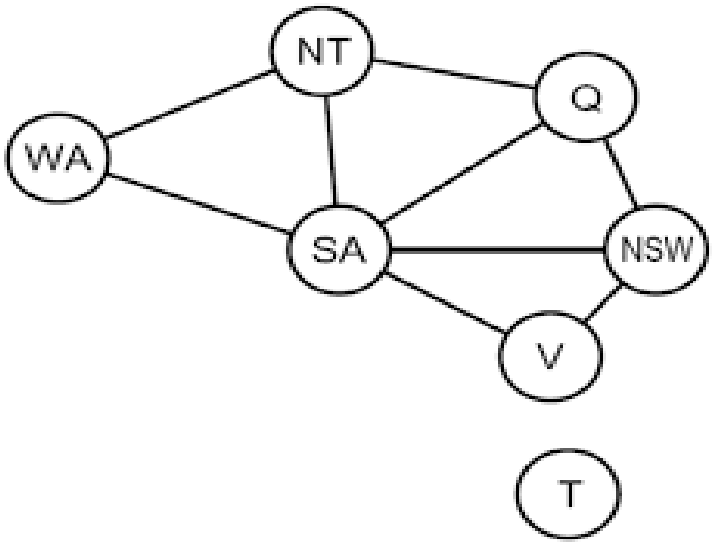
In previous different algorithms, such as the A* algorithm,we should use the heuristic value or the heuristic function. The states are evaluated using these heuristic value and the goal is attained.

But, in constraint satisfaction problem, it is the representation of states which has variable value. Since we have many and different constraints, this algorithm is more efficient algorithm.

We can find any solution from this algorithm, but the most important question would be, whether it is optimal. In other words, the solution may or maynot be optimal, either they have a solution or they do not.

AUSTRALIA PROBLEM:

The Australian map is divided into different regions that is the states. We should color the Australian map such that we should only use three colors and the adjacent region should not be colored same.
Firstly, put it into a graph(tree). Use the search which helps us to guide through.



X={SA,NSW,NT,Q,WA,V,T}
D={red,blue,green} for all Xi belongs to X
C={(for all Xi, Xj,such that Xi touches Xj},
{color(Xi) not equal to color(Xj)}

DFS-BACKTRACKING SEARCH:

If a single variable assignment is used in Depth First Search for a constraint satisfaction problem, then that approach is called the backtracking search. This method is an uninformed algorithm.
The variable assignment can be as follows,
If WA=R then NT=G. This is similar to NT=G then WA=R
This property is called the commutative law.
In simple words, backtracking algorithm is one if the current state break the constraint or does not satisfies the constraint, then it back tracks the previous states one or many times until the goal is achieved and satisfying the constraint.

MINIMUM REMAINING VALUES:

This chooses the variable of minimum legal values.It assigns the best value to its sooner not the later as it can be left unassigned when the constraint variable will run out of values. This then results in backtracking.

DEGREE HEURISTIC:

The Degree Heuristic or the DH chooses the variable with most constraint on the remaining variable.

LEAST CONSTRAINING VALUE:

If the variable has the variable that rules out the least value in the rest of the variable.

CONSTRAINT PROPAGATION:

This is a technique similar to forward checking, the future states of the unassigned variable is searched. To improve the efficiency of searching, we should try to detect the failures that had occurred.. Even if are not able to detect the failures completely, we will be able to detect the obvious defects, that needed to be fixed to efficiently search in the future searches.
It should be applied to every level of the searching process. We should better off if the propagation slows down. If the propagation slows down, this may affect the searching process as well.

NODE CONSISTENCY:

Every unary constraint in a variable is to be satisfied by every values in a domain of these variable. And the vise-versa should also be true.

ARC CONSISTENCY:

If the value of a variable in its domain satisfies the variable's binary constraint, then that variable is said to be arc consistent.

FORWARD CHECKING:

The forward checking approach is similar to backtracking search method with the propagation approach. Like in propagation, even forward checking looks the future steps(look ahead).

Let us assume a variable X, in forward checking process, it first establishes an arc consistency. Let Y be an uninformed variable which is connected to X(constraint).

Additionally, delete Y's any value from the domain which is inconsistent with X's value.

If arc consistency is done as a pre-processing step, then there is no purpose in doing the arc consistency.

SOURCE CODE:

1)heuristic_included_csp.py

```
from input_files import
heuristic_america_graph,heuristic_america_states,heuristic_australia_graph,heuristic_a
ustralia_state
import time
import random
from Node import Node


"""
Intialising the default values so that they can be used for finding
the answer for the various functions.
"""
colorlist = []
all_states = []
states = []
statedict={}
backtracks = 0
"""
THIS FUNCTION IS USED TO SELECT THE NEIGHBOURING
STATES IN GRAPH
"""
def pick_big_state(legal_colors):
   for k in sorted(legal_colors, key=lambda k: len(legal_colors[k]), reverse=True):
       #print k
       return k

def update_neighbors(statechanged,legal_colors,statedict,color_assigned,states):
   for state in statedict[statechanged]:
           #print(state)
           if color_assigned in list(filter(lambda x:x[0]==state,legal_colors))[0][1]:
               list(filter(lambda
x:x[0]==state,legal_colors))[0][1].remove(color_assigned)


def init_colors(n):
```

```
    for i in range(n):
        colorlist.append(random.randint(0,255))

def random_color(n):
    return tuple(colorlist)
"""
Building graph for the states
which can be traversed for assigning the values with
"""
def constructing_state_color_graph(num,numstates,states):
    colors = random_color(3)
    a = Node(colors[0],states[0])
    root = a
    mystates = {}
    for i in range(1,numstates,1):
        w = Node(colors[0],states[i])
        a.put_child(w)
        a.nextnode = w
        #a = w
        mystates[states[i]] = a

        for j in range(1,num,1):
            b = Node(colors[j])
            a.put_child(b)
        a = w
    print(root)
    print(root.next)
    print(root.next[1].next)

"""
This is the getter function for getting the corresponding
colors of the states.
"""
def colors_getter_function(states,mystatedict):
    listcol = []
    for i in states:
        listcol.append(mystatedict.get(i,""))

    return listcol

"""
This function is used to assign the
```

```
colors to the states so that we can generate
colors for the states

"""
def assigning_colors_state(states,i,numcolors,colors):
   tlist = []
   #colors = random_color(numcolors)
   for j in range(numcolors):
       tlist.append(Node(colors[j],states[i]))

   return tlist

"""
This is used to calculate the heuristic value for
assigning the colors to the states.
"""
def
heuristic_function_colors(dict_of_state1,dict_of_state,number_of_colors,present_con_st
ates,objects,count,colors_origi):
   global backtracks
   all_states.append(dict_of_state1.copy())
   for _ in range(len(present_con_states.next)):
       dict_of_state1[present_con_states.next[0].myname] =
present_con_states.next[_].mycolor
       temp_colors_origi = colors_origi.copy()
       if dict_of_state1.get(present_con_states.next[0].myname) in
colors_getter_function(dict_of_state[present_con_states.next[0].myname],dict_of_state1
):
         continue
       if count == len(objects) - 1:
           return 1,dict_of_state1
       backtracks +=1

update_neighbors(present_con_states.next[0].myname,temp_colors_origi,statedict,present
_con_states.next[_].mycolor,objects)
       temp_colors_origi = sorted(temp_colors_origi,key=lambda k:len(k[1]))
       temp_colors_for_states = colorlist.copy()
       present_con_states.next[_].next =
assigning_colors_state(states,count+1,number_of_colors,temp_colors_for_states)
       result =
heuristic_function_colors(dict_of_state1,dict_of_state,number_of_colors,present_con_st
ates.next[_],objects,count+1,temp_colors_origi)
```

```
        if result[0] == 1:
            return 1,dict_of_state
        continue
    return 0,dict_of_state

"""
Reading the nodes and initialising
with the colors from the list.
"""
def init(states,statedict,numcolors):
    colors = colorlist
    root = Node(colors[0],states[0])
    for j in range(numcolors):
        root.put_child(Node(colors[j],states[0]))

    return root
"""
This is the main driver function where the user gives the input '
"""
if __name__ == "__main__":
    numcolors = 4
    init_colors(numcolors)
    n=int(input("ENTER THE STATES THAT YOU WANT TO CONSIDER \n1.America State
\n2.Australia State"))
    colorlist = ["red","blue","green","black"]
    if(n==1):
        statedict =heuristic_america_graph

        states = heuristic_america_states
    elif(n==2):
        statedict = heuristic_australia_graph
        states= heuristic_australia_state

    legal_colors = []

    for state in states:
        legal_colors.append([state,colorlist.copy()])
    mystatedict = {}
    root = init(states,statedict,numcolors)
    start_time = time.time()
    answer =
heuristic_function_colors(mystatedict,statedict,numcolors,root,states,0,legal_colors)
```

```
end_time = time.time()
count = 0
for key in answer[1]:
    count+=1
    if answer[1][key] in colors_getter_function(statedict[key],mystatedict):
        print("We are not able to assign the colors for the states in the graph ")
print("THE COLORS ASSIGNED FOR THE STATES OF THE GRAPH ARE: ")
print(answer)
print("THE NUMBER OF EXECUTION THAT HAPPENED IN THE PROGRAM "+ str(backtracks))
print("THE TOTAL RUNTIME OF THE PROGRAM" + str(end_time - start_time) + " seconds")
```

2) dfs_csp.py

```
import time
from input_files import
dfs_australia_states,dfs_australia_graph,dfs_america_states,dfs_america_graph
import random
from Node import Node



"""
Intialising the default values so that they can be used for finding
the answer for the various functions.
"""
colors_for_states = []
states_present = []
states = []
statedict={}
back_propagation = 0
"""
This function is used for initialisation of colors.
"""
def initialisation_of_colors(number_of_colors):
   for _ in range(number_of_colors):
       colors_for_states.append(random.randint(0,255))



"""
This function is used for the heuristic calculation of the
states whether the current color can be used to assign to them or not.
"""
def
heuristic_calc_dfs(dict_of_state1,dict_of_state,number_of_colors,present_con_states,ob
jects,count,colors_origi):
   global back_propagation
   states_present.append(dict_of_state1.copy())
   for _ in range(len(present_con_states.next)):
       dict_of_state1[present_con_states.next[0].myname] =
present_con_states.next[_].mycolor
       temp_colors_origi = colors_origi.copy()
       if dict_of_state1.get(present_con_states.next[0].myname) in
getcolors(dict_of_state[present_con_states.next[0].myname],dict_of_state1):
```

```
            continue
        if count == len(objects) - 1:
            return 1,dict_of_state1
        back_propagation +=1

adjust_adjacent_states(present_con_states.next[0].myname,temp_colors_origi,statedict,p
resent_con_states.next[_].mycolor,objects)
        temp_colors_origi = sorted(temp_colors_origi,key=lambda k:len(k[1]))
        temp_colors_for_states = colors_for_states.copy()
        present_con_states.next[_].next =
assigning_of_colors(states,count+1,number_of_colors,temp_colors_for_states)
        result =
heuristic_calc_dfs(dict_of_state1,dict_of_state,number_of_colors,present_con_states.ne
xt[_],objects,count+1,temp_colors_origi)
        if result[0] == 1:
            return 1,dict_of_state
        continue
    return 0,dict_of_state

"""
This function is used for intialising the states with the
random colors so that we may have a start point for everything.
"""
def init(objects,statedict,number_of_colors):
    colors_used = colors_for_states
    value = Node(colors_used[0],objects[0])
    for j in range(number_of_colors):
        value.put_child(Node(colors_used[j],objects[0]))
    return value

"""
This function is used to adjust the neighbouring
states based on the colors that are available to assign to the states.
"""
def
adjust_adjacent_states(new_state,color_origi,dict_of_state,assigned_colors,state_nodes
):
    for _ in dict_of_state[new_state]:
            if assigned_colors in list(filter(lambda y:y[0]==_,color_origi))[0][1]:
                result=list(filter(lambda y:y[0]==_,color_origi))[0][1]
                result.remove(assigned_colors)
```

```
"""
THIS FUNCTION IS USED TO GET THE STATE THEIR CURRENT
COLOR.
"""
def getcolors(nodes_state,dict_of_state1):
   colors_of_list = []
   for _ in nodes_state:
       colors_of_list.append(dict_of_state1.get(_,""))
   return colors_of_list

"""
This function is used to assign the states with the
random colors in the begning.
"""
def assigning_random_colors(number_of_colors):
   return tuple(colors_for_states)



"""
This function is used to generate the colors for the state and to
create a node and assigning values to it.
"""
def assigning_of_colors(objects,k,number_of_colors,given_colors):
   temp_var = []
   for _ in range(number_of_colors):
       temp_var.append(Node(given_colors[_],objects[k]))

   return temp_var

"""
This function is used to calculate the dfs without heurisitcs
"""
def
calc_dfs_without_heuristic(dict_of_state1,dict_of_state,number_of_colors,present_con_s
tates,objects,count):

   global back_propagation
   for _ in range(len(present_con_states.next)):
       dict_of_state1[present_con_states.next[0].myname] =
present_con_states.next[_].mycolor
```

```
        if dict_of_state1.get(present_con_states.next[0].myname) in
getcolors(dict_of_state[present_con_states.next[0].myname],dict_of_state1):
            continue
        states_present.append(dict_of_state1.copy())
        temp_list_of_colors = colors_for_states.copy()
        if count == len(objects) - 1:
            return 1,dict_of_state1

        present_con_states.next[_].next =
assigning_of_colors(objects,count+1,number_of_colors,temp_list_of_colors)

        ans =
calc_dfs_without_heuristic(dict_of_state1,dict_of_state,number_of_colors,present_con_s
tates.next[_],objects,count+1)
        if ans[0] == 1:
            return 1,dict_of_state1
        continue

    back_propagation = back_propagation + 1
    return 0,dict_of_state1

"""
This is the driver function of the program
where the execution starts in the program.
"""
if __name__ == "__main__":
    number_of_colors = 4
    initialisation_of_colors(number_of_colors)
    colors_for_states = ["red","blue","green","black"]
    n=int(input("Enter the input for America or Australia states to be considered \n
1.America States \n2.Australia States"))
    if(n==1):
        statedict = dfs_america_graph

        states = dfs_america_states
    elif(n==2):
        states=dfs_australia_states
        statedict=dfs_australia_graph
    i=int(input("Enter the method:\n1.DFS with heuristic \n2.DFS without heuristics"))

    colors_original = []
    for _ in states:
```

```
        colors_original.append([_,colors_for_states.copy()])

    head = init(states,statedict,number_of_colors)
    dict_of_states = {}
    now = time.time()
    if(i==1):
        result =
heuristic_calc_dfs(dict_of_states,statedict,number_of_colors,head,states,0,colors_orig
inal)
    elif(i==2):
        result=
calc_dfs_without_heuristic(dict_of_states,statedict,number_of_colors,head,states,0)
    then = time.time()
    iter = 0
    for _ in result[1]:
        iter+=1
        if result[1][_] in getcolors(statedict[_],dict_of_states):
            print("THE COLORS COULD NOT BE ASSIGNED TO ALL THE VARIABLES")
    print(len(states_present))
    print("THE SOLUTION FOR THE SELECTED COUNTRY STATES IS THAT")
    print(result)
    print("NUMBER OF BACK TRACKING HAPPENED FOR FINDING THE ANSWER IS THAT: "+
str(back_propagation))
    print("THE RUNNING TIME OF THE DFS WITH HEURISTIC IS THAT" + str(then - now) +
"seconds")
```

3)forward_checking_csp.py

```
import time
from input_files import
fc_america_states,fc_america_graph,fc_australia_states,fc_australia_graph
import random
from Node import Node

"""
Intialising the default values so that they can be used for finding
the answer for the various functions.
"""
colors_for_states = []
states_present = []
back_propagation = 0
states = []
statedict={}

"""
This function is used for intialising the states with the
random colors so that we may have a start point for everything.
"""
def init(objects,statedict,number_of_colors):
   colors_used = colors_for_states
   value = Node(colors_used[0],objects[0])
   for j in range(number_of_colors):
       value.put_child(Node(colors_used[j],objects[0]))
   return value

"""
This function is used to assign the states with the
random colors in the begning.
"""
def assigning_random_colors(number_of_colors):
   return tuple(colors_for_states)

"""
This function is used for initialisation of colors.
"""
def initialisation_of_colors(number_of_colors):
   for _ in range(number_of_colors):
```

```
        colors_for_states.append(random.randint(0,255))

"""
THIS FUNCTION IS USED TO GET THE STATE THEIR CURRENT
COLOR.
"""
def getcolors(nodes_state,dict_of_state1):
    colors_of_list = []
    for _ in nodes_state:
        colors_of_list.append(dict_of_state1.get(_,""))
    return colors_of_list




"""
This function is used to generate the colors for the state and to
create a node and assigning values to it.
"""
def assigning_of_colors(objects,k,number_of_colors,given_colors):
    temp_var = []
    for _ in range(number_of_colors):
        temp_var.append(Node(given_colors[_],objects[k]))
    return temp_var

"""
This function is used for the forward checking propagation of the
graph.
"""
def
checking_foward_propagation(dict_of_state1,dict_of_state,number_of_colors,present_cond
_states,objects,count):

    global back_propagation
    colors_for_states.append(dict_of_state1.copy())
    for _ in range(len(present_cond_states.next)):
        time.sleep(0.000002)
        dict_of_state1[present_cond_states.next[0].myname] =
present_cond_states.next[_].mycolor
        if dict_of_state1.get(present_cond_states.next[0].myname) in
getcolors(dict_of_state[present_cond_states.next[0].myname],dict_of_state1):
```

```
            continue
        if count == len(objects) - 1:
            return 1,dict_of_state1

        tcolor_var_list = colourlist.copy()
        getting_color = getcolors(objects[count+1],dict_of_state1)
        tcolor_var_list = [x for x in tcolor_var_list if x not in getting_color]
        present_cond_states.next[_].next =
assigning_of_colors(objects,count+1,number_of_colors,tcolor_var_list)

        result =
checking_foward_propagation(dict_of_state1,dict_of_state,number_of_colors,present_cond
_states.next[_],objects,count+1)
        if result[0] == 1:
            return 1,dict_of_state1
        continue
    back_propagation +=1

    return 0,dict_of_state1




"""
This is the driver function of the program
where the execution starts in the program.
"""
if __name__ == "__main__":
    numcolours = 4
    initialisation_of_colors(numcolours)

    colourlist = ["red","blue","green","black"]
    n=int(input("Enter the input for America or Australia states to be considered \n
1.America States \n2.Australia States"))
    if(n==1):
        statedict = fc_america_graph
        states = fc_america_states
    elif(n==2):
        states = fc_australia_states
        statedict  = fc_australia_graph
```

```
    dict_of_states = {}
    print(states)
    head = init(states,statedict,numcolours)
    iter = 0
    now = time.time()
    result =
checking_foward_propagation(dict_of_states,statedict,numcolours,head,states,0)
    then = time.time()

    for _ in result[1]:
        iter+=1
        if result[1][_] in getcolors(statedict[_],dict_of_states):
                print("THE COLORS COULD NOT BE ASSIGNED TO ALL THE VARIABLES")



    print("THE SOLUTION FOR THE SELECTED COUNTRY STATES IS THAT")
    print(result)
    print("NUMBER OF BACK TRACKING HAPPENED FOR FINDING THE ANSWER IS THAT: "+
str(back_propagation))
    print("THE RUNNING TIME OF THE DFS WITH HEURISTIC IS THAT" + str(then - now) +
"seconds")
    print("THE NUMBER OF THE COLORS ASSIGNED FOR THE ALL THE
STATES",len(colors_for_states))
    time.sleep(10)
```

4) input_files.py

```
#THESE ARE THE INPUTS THAT ARE BEING USED FOR VARIOUS FUNCTIONS.
csp_singleton_america_states= ['New Hampshire', 'Oklahoma', 'Tennessee', 'Illinois',
'New Mexico', 'Kentucky', 'West Virginia', 'Maryland', 'Maine', 'Wisconsin',
'Missouri', 'Minnesota', 'Montana', 'Massachusetts', 'South Carolina', 'North Dakota',
'Pennsylvania', 'Arizona', 'South Dakota', 'Ohio', 'Oregon', 'Alabama', 'Indiana',
'Rhode Island', 'Virginia', 'Idaho', 'Nevada', 'Nebraska', 'New York', 'Utah',
'Michigan', 'Kansas', 'Florida', 'Connecticut', 'Iowa', 'Wyoming', 'Louisiana',
'California', 'Vermont', 'Texas', 'Georgia', 'New Jersey', 'North Carolina',
'Washington', 'Delaware', 'colourado', 'Mississippi', 'Arkansas']
csp_singleton_america_graph ={
       'Alabama':['Florida', 'Georgia', 'Mississippi', 'Tennessee'],
       'Arizona':['California', 'colourado', 'Nevada', 'New Mexico', 'Utah'],
       'Arkansas' :['Louisiana', 'Mississippi', 'Missouri', 'Oklahoma', 'Tennessee',
'Texas'],
       'California':['Arizona', 'Nevada', 'Oregon'],
       'colourado':['Arizona', 'Kansas', 'Nebraska', 'New Mexico', 'Oklahoma', 'Utah',
'Wyoming'],
       'Connecticut':['Massachusetts', 'New York', 'Rhode Island'],
       'Delaware':['Maryland', 'New Jersey', 'Pennsylvania'],
       'Florida':['Alabama', 'Georgia'],
       'Georgia':['Alabama', 'Florida', 'North Carolina', 'South Carolina',
'Tennessee'],
       'Idaho':['Montana', 'Nevada', 'Oregon', 'Utah', 'Washington', 'Wyoming'],
       'Illinois':['Indiana','Iowa', 'Michigan', 'Kentucky', 'Missouri', 'Wisconsin'],
       'Indiana':['Illinois', 'Kentucky', 'Michigan', 'Ohio'],
       'Iowa': ['Illinois', 'Minnesota', 'Missouri', 'Nebraska', 'South Dakota',
'Wisconsin'],
       'Kansas' :['colourado', 'Missouri', 'Nebraska', 'Oklahoma'],
       'Kentucky':['Illinois', 'Indiana', 'Missouri', 'Ohio', 'Tennessee', 'Virginia',
'West Virginia'],
       'Louisiana':['Arkansas', 'Mississippi', 'Texas'],
       'Maine':["New Hampshire"],
       "Maryland":['Delaware','Pennsylvania','Virginia', 'West Virginia'],
       'Massachusetts':['Connecticut', 'New Hampshire', 'New York', 'Rhode Island',
'Vermont'],
       'Michigan':['Illinois', 'Indiana', 'Minnesota', 'Ohio', 'Wisconsin'],
       'Minnesota':['Iowa', 'Michigan', 'North Dakota', 'South Dakota', 'Wisconsin'],
       'Mississippi':['Alabama', 'Arkansas', 'Louisiana', 'Tennessee'],
```

```
       'Missouri':['Arkansas', 'Illinois', 'Iowa', 'Kansas', 'Kentucky', 'Nebraska',
'Oklahoma', 'Tennessee'],
       'Montana':['Idaho', 'North Dakota', 'South Dakota', 'Wyoming'],
       'Nebraska' :['colourado', 'Iowa', 'Kansas', 'Missouri', 'South Dakota',
'Wyoming'],
       'Nevada':['Arizona', 'California', 'Idaho', 'Oregon', 'Utah'],
       'New Hampshire': ['Maine', 'Massachusetts', 'Vermont'],
       'New Jersey':["Delaware", "New York", "Pennsylvania"],
       'New Mexico':['Arizona', 'colourado', 'Oklahoma', 'Texas', 'Utah'],
       'New York':['Connecticut', 'Massachusetts', 'New Jersey', 'Pennsylvania',
'Rhode Island', 'Vermont'],
       'North Carolina':['Georgia', 'South Carolina', 'Tennessee', 'Virginia'],
       'North Dakota':['Minnesota', 'Montana', 'South Dakota'],
       'Ohio':['Indiana', 'Kentucky', 'Michigan', 'Pennsylvania', 'West Virginia'],
       'Oklahoma' :['Arkansas', 'colourado', 'Kansas', 'Missouri', 'New Mexico',
'Texas'],
       'Oregon':["California", 'Idaho', 'Nevada', "Washington"],
       'Pennsylvania':['Delaware', 'Maryland', 'New Jersey', 'New York', 'Ohio', 'West
Virginia'],
       'Rhode Island':['Connecticut', 'Massachusetts', 'New York'],
       'South Carolina':['Georgia', 'North Carolina'],
       'South Dakota':['Iowa', 'Minnesota', 'Montana', 'Nebraska', 'North Dakota',
'Wyoming'],
       'Tennessee':['Alabama', 'Arkansas', 'Georgia', 'Kentucky', 'Mississippi',
'Missouri', 'North Carolina', 'Virginia'],
       'Texas':['Arkansas', 'Louisiana', 'New Mexico', 'Oklahoma'],
       'Utah':['Arizona', 'colourado', 'Idaho', 'Nevada', 'New Mexico', 'Wyoming'],
       'Vermont':['Massachusetts', 'New Hampshire', 'New York'],
       'Virginia':['Kentucky', 'Maryland', 'North Carolina', 'Tennessee', 'West
Virginia'],
       'Washington':['Idaho', 'Oregon'],
       'West Virginia':['Kentucky', 'Maryland', 'Ohio', 'Pennsylvania', 'Virginia'],
       'Wisconsin':['Illinois', 'Iowa', 'Michigan', 'Minnesota'],
       'Wyoming':['colourado', 'Idaho', 'Montana', 'Nebraska', 'South Dakota',
'Utah'],
       'Hawaii':[],
       'Alaska':[]}
csp_singleton_australia_states=['wa','nt','q','nsw','v','sa']
csp_singleton_australia_graph={
          'wa':['nt','sa'],
          'nt':['wa','q','sa'],
          'sa':['wa','q','nsw','nt','v'],
```

```
            'q':['nt','sa','nsw'],
            'nsw':['q','v','sa'],
            'v':['sa','nsw']
            }
dfs_america_graph = {
   'Alabama':['Florida', 'Georgia', 'Mississippi', 'Tennessee'],
   'Arizona':['California', 'Colorado', 'Nevada', 'New Mexico', 'Utah'],
   'Arkansas' :['Louisiana', 'Mississippi', 'Missouri', 'Oklahoma', 'Tennessee',
'Texas'],
   'California':['Arizona', 'Nevada', 'Oregon'],
   'Colorado':['Arizona', 'Kansas', 'Nebraska', 'New Mexico', 'Oklahoma', 'Utah',
'Wyoming'],
   'Connecticut':['Massachusetts', 'New York', 'Rhode Island'],
   'Delaware':['Maryland', 'New Jersey', 'Pennsylvania'],
   'Florida':['Alabama', 'Georgia'],
   'Georgia':['Alabama', 'Florida', 'North Carolina', 'South Carolina', 'Tennessee'],
   'Idaho':['Montana', 'Nevada', 'Oregon', 'Utah', 'Washington', 'Wyoming'],
   'Illinois':['Indiana','Iowa', 'Michigan', 'Kentucky', 'Missouri', 'Wisconsin'],
   'Indiana':['Illinois', 'Kentucky', 'Michigan', 'Ohio'],
   'Iowa': ['Illinois', 'Minnesota', 'Missouri', 'Nebraska', 'South Dakota',
'Wisconsin'],
   'Kansas' :['Colorado', 'Missouri', 'Nebraska', 'Oklahoma'],
   'Kentucky':['Illinois', 'Indiana', 'Missouri', 'Ohio', 'Tennessee', 'Virginia',
'West Virginia'],
   'Louisiana':['Arkansas', 'Mississippi', 'Texas'],
   'Maine':["New Hampshire"],
   "Maryland":['Delaware','Pennsylvania','Virginia', 'West Virginia'],
   'Massachusetts':['Connecticut', 'New Hampshire', 'New York', 'Rhode Island',
'Vermont'],
   'Michigan':['Illinois', 'Indiana', 'Minnesota', 'Ohio', 'Wisconsin'],
   'Minnesota':['Iowa', 'Michigan', 'North Dakota', 'South Dakota', 'Wisconsin'],
   'Mississippi':['Alabama', 'Arkansas', 'Louisiana', 'Tennessee'],
   'Missouri':['Arkansas', 'Illinois', 'Iowa', 'Kansas', 'Kentucky', 'Nebraska',
'Oklahoma', 'Tennessee'],
   'Montana':['Idaho', 'North Dakota', 'South Dakota', 'Wyoming'],
   'Nebraska' :['Colorado', 'Iowa', 'Kansas', 'Missouri', 'South Dakota', 'Wyoming'],
   'Nevada':['Arizona', 'California', 'Idaho', 'Oregon', 'Utah'],
   'New Hampshire': ['Maine', 'Massachusetts', 'Vermont'],
   'New Jersey':["Delaware", "New York", "Pennsylvania"],
   'New Mexico':['Arizona', 'Colorado', 'Oklahoma', 'Texas', 'Utah'],
   'New York':['Connecticut', 'Massachusetts', 'New Jersey', 'Pennsylvania', 'Rhode
Island', 'Vermont'],
```

```
   'North Carolina':['Georgia', 'South Carolina', 'Tennessee', 'Virginia'],
   'North Dakota':['Minnesota', 'Montana', 'South Dakota'],
   'Ohio':['Indiana', 'Kentucky', 'Michigan', 'Pennsylvania', 'West Virginia'],
   'Oklahoma' :['Arkansas', 'Colorado', 'Kansas', 'Missouri', 'New Mexico', 'Texas'],
   'Oregon':["California", 'Idaho', 'Nevada', "Washington"],
   'Pennsylvania':['Delaware', 'Maryland', 'New Jersey', 'New York', 'Ohio', 'West
Virginia'],
   'Rhode Island':['Connecticut', 'Massachusetts', 'New York'],
   'South Carolina':['Georgia', 'North Carolina'],
   'South Dakota':['Iowa', 'Minnesota', 'Montana', 'Nebraska', 'North Dakota',
'Wyoming'],
   'Tennessee':['Alabama', 'Arkansas', 'Georgia', 'Kentucky', 'Mississippi',
'Missouri', 'North Carolina', 'Virginia'],
   'Texas':['Arkansas', 'Louisiana', 'New Mexico', 'Oklahoma'],
   'Utah':['Arizona', 'Colorado', 'Idaho', 'Nevada', 'New Mexico', 'Wyoming'],
   'Vermont':['Massachusetts', 'New Hampshire', 'New York'],
   'Virginia':['Kentucky', 'Maryland', 'North Carolina', 'Tennessee', 'West
Virginia'],
   'Washington':['Idaho', 'Oregon'],
   'West Virginia':['Kentucky', 'Maryland', 'Ohio', 'Pennsylvania', 'Virginia'],
   'Wisconsin':['Illinois', 'Iowa', 'Michigan', 'Minnesota'],
   'Wyoming':['Colorado', 'Idaho', 'Montana', 'Nebraska', 'South Dakota', 'Utah']
   }
dfs_america_states = ['Maine', 'Minnesota', 'South Dakota', 'Illinois', 'Utah',
'Wyoming', 'Texas', 'Idaho', 'Wisconsin', 'Connecticut', 'Pennsylvania', 'Kansas',
'West Virginia', 'North Carolina', 'Colorado', 'California', 'Florida', 'Vermont',
'Virginia', 'North Dakota', 'Michigan', 'New Jersey', 'Nevada', 'Arkansas',
'Mississippi', 'Iowa', 'Kentucky', 'Maryland', 'Louisiana', 'Alabama', 'Oklahoma',
'New Mexico', 'Rhode Island', 'Massachusetts', 'South Carolina', 'Indiana',
'Delaware', 'Tennessee', 'Georgia', 'Arizona', 'Nebraska', 'Missouri', 'New
Hampshire', 'Ohio', 'Oregon', 'Washington', 'Montana', 'New York']
dfs_australia_states=['wa','nt','q','nsw','v','sa']
dfs_australia_graph={
       'wa':['nt','sa'],
       'nt':['wa','q','sa'],
       'sa':['wa','q','nsw','nt','v'],
       'q':['nt','sa','nsw'],
       'nsw':['q','v','sa'],
       'v':['sa','nsw']}
fc_america_graph={
   'Alabama':['Florida', 'Georgia', 'Mississippi', 'Tennessee'],
   'Arizona':['California', 'colourado', 'Nevada', 'New Mexico', 'Utah'],
```

```
   'Arkansas' :['Louisiana', 'Mississippi', 'Missouri', 'Oklahoma', 'Tennessee',
'Texas'],
   'California':['Arizona', 'Nevada', 'Oregon'],
   'colourado':['Arizona', 'Kansas', 'Nebraska', 'New Mexico', 'Oklahoma', 'Utah',
'Wyoming'],
   'Connecticut':['Massachusetts', 'New York', 'Rhode Island'],
   'Delaware':['Maryland', 'New Jersey', 'Pennsylvania'],
   'Florida':['Alabama', 'Georgia'],
   'Georgia':['Alabama', 'Florida', 'North Carolina', 'South Carolina', 'Tennessee'],
   'Idaho':['Montana', 'Nevada', 'Oregon', 'Utah', 'Washington', 'Wyoming'],
   'Illinois':['Indiana','Iowa', 'Michigan', 'Kentucky', 'Missouri', 'Wisconsin'],
   'Indiana':['Illinois', 'Kentucky', 'Michigan', 'Ohio'],
   'Iowa': ['Illinois', 'Minnesota', 'Missouri', 'Nebraska', 'South Dakota',
'Wisconsin'],
   'Kansas' :['colourado', 'Missouri', 'Nebraska', 'Oklahoma'],
   'Kentucky':['Illinois', 'Indiana', 'Missouri', 'Ohio', 'Tennessee', 'Virginia',
'West Virginia'],
   'Louisiana':['Arkansas', 'Mississippi', 'Texas'],
   'Maine':["New Hampshire"],
   "Maryland":['Delaware','Pennsylvania','Virginia', 'West Virginia'],
   'Massachusetts':['Connecticut', 'New Hampshire', 'New York', 'Rhode Island',
'Vermont'],
   'Michigan':['Illinois', 'Indiana', 'Minnesota', 'Ohio', 'Wisconsin'],
   'Minnesota':['Iowa', 'Michigan', 'North Dakota', 'South Dakota', 'Wisconsin'],
   'Mississippi':['Alabama', 'Arkanssas', 'Louisiana', 'Tennessee'],
   'Missouri':['Arkansas', 'Illinois', 'Iowa', 'Kansas', 'Kentucky', 'Nebraska',
'Oklahoma', 'Tennessee'],
   'Montana':['Idaho', 'North Dakota', 'South Dakota', 'Wyoming'],
   'Nebraska' :['colourado', 'Iowa', 'Kansas', 'Missouri', 'South Dakota', 'Wyoming'],
   'Nevada':['Arizona', 'California', 'Idaho', 'Oregon', 'Utah'],
   'New Hampshire': ['Maine', 'Massachusetts', 'Vermont'],
   'New Jersey':["Delaware", "New York", "Pennsylvania"],
   'New Mexico':['Arizona', 'colourado', 'Oklahoma', 'Texas', 'Utah'],
   'New York':['Connecticut', 'Massachusetts', 'New Jersey', 'Pennsylvania', 'Rhode
Island', 'Vermont'],
   'North Carolina':['Georgia', 'South Carolina', 'Tennessee', 'Virginia'],
   'North Dakota':['Minnesota', 'Montana', 'South Dakota'],
   'Ohio':['Indiana', 'Kentucky', 'Michigan', 'Pennsylvania', 'West Virginia'],
   'Oklahoma' :['Arkansas', 'colourado', 'Kansas', 'Missouri', 'New Mexico', 'Texas'],
   'Oregon':["California", 'Idaho', 'Nevada', "Washington"],
   'Pennsylvania':['Delaware', 'Maryland', 'New Jersey', 'New York', 'Ohio', 'West
Virginia'],
```

```
   'Rhode Island':['Connecticut', 'Massachusetts', 'New York'],
   'South Carolina':['Georgia', 'North Carolina'],
   'South Dakota':['Iowa', 'Minnesota', 'Montana', 'Nebraska', 'North Dakota',
'Wyoming'],
   'Tennessee':['Alabama', 'Arkansas', 'Georgia', 'Kentucky', 'Mississippi',
'Missouri', 'North Carolina', 'Virginia'],
   'Texas':['Arkansas', 'Louisiana', 'New Mexico', 'Oklahoma'],
   'Utah':['Arizona', 'colourado', 'Idaho', 'Nevada', 'New Mexico', 'Wyoming'],
   'Vermont':['Massachusetts', 'New Hampshire', 'New York'],
   'Virginia':['Kentucky', 'Maryland', 'North Carolina', 'Tennessee', 'West
Virginia'],
   'Washington':['Idaho', 'Oregon'],
   'West Virginia':['Kentucky', 'Maryland', 'Ohio', 'Pennsylvania', 'Virginia'],
   'Wisconsin':['Illinois', 'Iowa', 'Michigan', 'Minnesota'],
   'Wyoming':['colourado', 'Idaho', 'Montana', 'Nebraska', 'South Dakota', 'Utah'],
   "Hawai":[],
   "Alaska":[]
   }
fc_america_states =['New Hampshire', 'Oklahoma', 'Tennessee', 'Illinois', 'New
Mexico', 'Kentucky', 'West Virginia', 'Maryland', 'Maine', 'Wisconsin', 'Missouri',
'Minnesota', 'Montana', 'Massachusetts', 'South Carolina', 'North Dakota',
'Pennsylvania', 'Arizona', 'South Dakota', 'Ohio', 'Oregon', 'Alabama', 'Indiana',
'Rhode Island', 'Virginia', 'Idaho', 'Nevada', 'Nebraska', 'New York', 'Utah',
'Michigan', 'Kansas', 'Florida', 'Connecticut', 'Iowa', 'Wyoming', 'Louisiana',
'California', 'Vermont', 'Texas', 'Georgia', 'New Jersey', 'North Carolina',
'Washington', 'Delaware', 'colourado', 'Mississippi', 'Arkansas']
fc_australia_states=['wa','nt','q','nsw','v','sa']
fc_australia_graph={
      'wa':['nt','sa'],
      'nt':['wa','q','sa'],
      'sa':['wa','q','nsw','nt','v'],
      'q':['nt','sa','nsw'],
      'nsw':['q','v','sa'],
      'v':['sa','nsw']}

heuristic_america_graph={
'Alabama':['Florida', 'Georgia', 'Mississippi', 'Tennessee'],
'Arizona':['California', 'Colorado', 'Nevada', 'New Mexico', 'Utah'],
'Arkansas' :['Louisiana', 'Mississippi', 'Missouri', 'Oklahoma', 'Tennessee',
'Texas'],
'California':['Arizona', 'Nevada', 'Oregon'],
```

```
'Colorado':['Arizona', 'Kansas', 'Nebraska', 'New Mexico', 'Oklahoma', 'Utah',
'Wyoming'],
'Connecticut':['Massachusetts', 'New York', 'Rhode Island'],
'Delaware':['Maryland', 'New Jersey', 'Pennsylvania'],
'Florida':['Alabama', 'Georgia'],
'Georgia':['Alabama', 'Florida', 'North Carolina', 'South Carolina', 'Tennessee'],
'Idaho':['Montana', 'Nevada', 'Oregon', 'Utah', 'Washington', 'Wyoming'],
'Illinois':['Indiana','Iowa', 'Michigan', 'Kentucky', 'Missouri', 'Wisconsin'],
'Indiana':['Illinois', 'Kentucky', 'Michigan', 'Ohio'],
'Iowa': ['Illinois', 'Minnesota', 'Missouri', 'Nebraska', 'South Dakota',
'Wisconsin'],
'Kansas' :['Colorado', 'Missouri', 'Nebraska', 'Oklahoma'],
'Kentucky':['Illinois', 'Indiana', 'Missouri', 'Ohio', 'Tennessee', 'Virginia', 'West
Virginia'],
'Louisiana':['Arkansas', 'Mississippi', 'Texas'],
'Maine':["New Hampshire"],
"Maryland":['Delaware','Pennsylvania','Virginia', 'West Virginia'],
'Massachusetts':['Connecticut', 'New Hampshire', 'New York', 'Rhode Island',
'Vermont'],
'Michigan':['Illinois', 'Indiana', 'Minnesota', 'Ohio', 'Wisconsin'],
'Minnesota':['Iowa', 'Michigan', 'North Dakota', 'South Dakota', 'Wisconsin'],
'Mississippi':['Alabama', 'Arkansas', 'Louisiana', 'Tennessee'],
'Missouri':['Arkansas', 'Illinois', 'Iowa', 'Kansas', 'Kentucky', 'Nebraska',
'Oklahoma', 'Tennessee'],
'Montana':['Idaho', 'North Dakota', 'South Dakota', 'Wyoming'],
'Nebraska' :['Colorado', 'Iowa', 'Kansas', 'Missouri', 'South Dakota', 'Wyoming'],
'Nevada':['Arizona', 'California', 'Idaho', 'Oregon', 'Utah'],
'New Hampshire': ['Maine', 'Massachusetts', 'Vermont'],
'New Jersey':["Delaware", "New York", "Pennsylvania"],
'New Mexico':['Arizona', 'Colorado', 'Oklahoma', 'Texas', 'Utah'],
'New York':['Connecticut', 'Massachusetts', 'New Jersey', 'Pennsylvania', 'Rhode
Island', 'Vermont'],
'North Carolina':['Georgia', 'South Carolina', 'Tennessee', 'Virginia'],
'North Dakota':['Minnesota', 'Montana', 'South Dakota'],
'Ohio':['Indiana', 'Kentucky', 'Michigan', 'Pennsylvania', 'West Virginia'],
'Oklahoma' :['Arkansas', 'Colorado', 'Kansas', 'Missouri', 'New Mexico', 'Texas'],
'Oregon':["California", 'Idaho', 'Nevada', "Washington"],
'Pennsylvania':['Delaware', 'Maryland', 'New Jersey', 'New York', 'Ohio', 'West
Virginia'],
'Rhode Island':['Connecticut', 'Massachusetts', 'New York'],
'South Carolina':['Georgia', 'North Carolina'],
```

```
'South Dakota':['Iowa', 'Minnesota', 'Montana', 'Nebraska', 'North Dakota',
'Wyoming'],
'Tennessee':['Alabama', 'Arkansas', 'Georgia', 'Kentucky', 'Mississippi', 'Missouri',
'North Carolina', 'Virginia'],
'Texas':['Arkansas', 'Louisiana', 'New Mexico', 'Oklahoma'],
'Utah':['Arizona', 'Colorado', 'Idaho', 'Nevada', 'New Mexico', 'Wyoming'],
'Vermont':['Massachusetts', 'New Hampshire', 'New York'],
'Virginia':['Kentucky', 'Maryland', 'North Carolina', 'Tennessee', 'West Virginia'],
'Washington':['Idaho', 'Oregon'],
'West Virginia':['Kentucky', 'Maryland', 'Ohio', 'Pennsylvania', 'Virginia'],
'Wisconsin':['Illinois', 'Iowa', 'Michigan', 'Minnesota'],
'Wyoming':['Colorado', 'Idaho', 'Montana', 'Nebraska', 'South Dakota', 'Utah']
}
heuristic_america_states=['Illinois', 'Oklahoma', 'California', 'Utah', 'Wyoming',
'Missouri', 'Michigan', 'Texas', 'Iowa', 'Delaware', 'Tennessee', 'Maryland',
'Kentucky', 'Montana', 'Minnesota', 'Connecticut', 'Louisiana', 'West Virginia',
'Pennsylvania', 'Nebraska', 'Kansas', 'Indiana', 'Rhode Island', 'Arizona', 'Florida',
'Massachusetts', 'South Dakota', 'Nevada', 'South Carolina', 'Ohio', 'New Hampshire',
'Idaho', 'Washington', 'Colorado', 'Oregon', 'New Jersey', 'Mississippi', 'Arkansas',
'Vermont', 'Wisconsin', 'Alabama', 'Georgia', 'Maine', 'New Mexico', 'North Carolina',
'New York', 'Virginia', 'North Dakota']
heuristic_australia_graph={
       'wa':['nt','sa'],
       'nt':['wa','q','sa'],
       'sa':['wa','q','nsw','nt','v'],
       'q':['nt','sa','nsw'],
       'nsw':['q','v','sa'],
       'v':['sa','nsw']}
heuristic_australia_state=['wa','nt','q','nsw','v','sa']
singleton_america_graph = {
       'Alabama':['Florida', 'Georgia', 'Mississippi', 'Tennessee'],
       'Arizona':['California', 'Colorado', 'Nevada', 'New Mexico', 'Utah'],
       'Arkansas' :['Louisiana', 'Mississippi', 'Missouri', 'Oklahoma', 'Tennessee',
'Texas'],
       'California':['Arizona', 'Nevada', 'Oregon'],
       'Colorado':['Arizona', 'Kansas', 'Nebraska', 'New Mexico', 'Oklahoma', 'Utah',
'Wyoming'],
       'Connecticut':['Massachusetts', 'New York', 'Rhode Island'],
       'Delaware':['Maryland', 'New Jersey', 'Pennsylvania'],
       'Florida':['Alabama', 'Georgia'],
       'Georgia':['Alabama', 'Florida', 'North Carolina', 'South Carolina',
'Tennessee'],
```

```
      'Idaho':['Montana', 'Nevada', 'Oregon', 'Utah', 'Washington', 'Wyoming'],
      'Illinois':['Indiana','Iowa', 'Michigan', 'Kentucky', 'Missouri', 'Wisconsin'],
      'Indiana':['Illinois', 'Kentucky', 'Michigan', 'Ohio'],
      'Iowa': ['Illinois', 'Minnesota', 'Missouri', 'Nebraska', 'South Dakota',
'Wisconsin'],
      'Kansas' :['Colorado', 'Missouri', 'Nebraska', 'Oklahoma'],
      'Kentucky':['Illinois', 'Indiana', 'Missouri', 'Ohio', 'Tennessee', 'Virginia',
'West Virginia'],
      'Louisiana':['Arkansas', 'Mississippi', 'Texas'],
      'Maine':["New Hampshire"],
      "Maryland":['Delaware','Pennsylvania','Virginia', 'West Virginia'],
      'Massachusetts':['Connecticut', 'New Hampshire', 'New York', 'Rhode Island',
'Vermont'],
      'Michigan':['Illinois', 'Indiana', 'Minnesota', 'Ohio', 'Wisconsin'],
      'Minnesota':['Iowa', 'Michigan', 'North Dakota', 'South Dakota', 'Wisconsin'],
      'Mississippi':['Alabama', 'Arkanssas', 'Louisiana', 'Tennessee'],
      'Missouri':['Arkansas', 'Illinois', 'Iowa', 'Kansas', 'Kentucky', 'Nebraska',
'Oklahoma', 'Tennessee'],
      'Montana':['Idaho', 'North Dakota', 'South Dakota', 'Wyoming'],
      'Nebraska' :['Colorado', 'Iowa', 'Kansas', 'Missouri', 'South Dakota',
'Wyoming'],
      'Nevada':['Arizona', 'California', 'Idaho', 'Oregon', 'Utah'],
      'New Hampshire': ['Maine', 'Massachusetts', 'Vermont'],
      'New Jersey':["Delaware", "New York", "Pennsylvania"],
      'New Mexico':['Arizona', 'Colorado', 'Oklahoma', 'Texas', 'Utah'],
      'New York':['Connecticut', 'Massachusetts', 'New Jersey', 'Pennsylvania',
'Rhode Island', 'Vermont'],
      'North Carolina':['Georgia', 'South Carolina', 'Tennessee', 'Virginia'],
      'North Dakota':['Minnesota', 'Montana', 'South Dakota'],
      'Ohio':['Indiana', 'Kentucky', 'Michigan', 'Pennsylvania', 'West Virginia'],
      'Oklahoma' :['Arkansas', 'Colorado', 'Kansas', 'Missouri', 'New Mexico',
'Texas'],
      'Oregon':["California", 'Idaho', 'Nevada', "Washington"],
      'Pennsylvania':['Delaware', 'Maryland', 'New Jersey', 'New York', 'Ohio', 'West
Virginia'],
      'Rhode Island':['Connecticut', 'Massachusetts', 'New York'],
      'South Carolina':['Georgia', 'North Carolina'],
      'South Dakota':['Iowa', 'Minnesota', 'Montana', 'Nebraska', 'North Dakota',
'Wyoming'],
      'Tennessee':['Alabama', 'Arkansas', 'Georgia', 'Kentucky', 'Mississippi',
'Missouri', 'North Carolina', 'Virginia'],
      'Texas':['Arkansas', 'Louisiana', 'New Mexico', 'Oklahoma'],
```

```
        'Utah':['Arizona', 'Colorado', 'Idaho', 'Nevada', 'New Mexico', 'Wyoming'],
        'Vermont':['Massachusetts', 'New Hampshire', 'New York'],
        'Virginia':['Kentucky', 'Maryland', 'North Carolina', 'Tennessee', 'West
Virginia'],
        'Washington':['Idaho', 'Oregon'],
        'West Virginia':['Kentucky', 'Maryland', 'Ohio', 'Pennsylvania', 'Virginia'],
        'Wisconsin':['Illinois', 'Iowa', 'Michigan', 'Minnesota'],
        'Wyoming':['Colorado', 'Idaho', 'Montana', 'Nebraska', 'South Dakota', 'Utah'],
        "Hawai":[],
        "Alaska":[]
        }
singleton_america_states= ['New Hampshire', 'Oklahoma', 'Tennessee', 'Illinois', 'New
Mexico', 'Kentucky', 'West Virginia', 'Maryland', 'Maine', 'Wisconsin', 'Missouri',
'Minnesota', 'Montana', 'Massachusetts', 'South Carolina', 'North Dakota',
'Pennsylvania', 'Arizona', 'South Dakota', 'Ohio', 'Oregon', 'Alabama', 'Indiana',
'Rhode Island', 'Virginia', 'Idaho', 'Nevada', 'Nebraska', 'New York', 'Utah',
'Michigan', 'Kansas', 'Florida', 'Connecticut', 'Iowa', 'Wyoming', 'Louisiana',
'California', 'Vermont', 'Texas', 'Georgia', 'New Jersey', 'North Carolina',
'Washington', 'Delaware', 'Colorado', 'Mississippi', 'Arkansas']

singleton_australia_graph={
        'wa':['nt','sa'],
        'nt':['wa','q','sa'],
        'sa':['wa','q','nsw','nt','v'],
        'q':['nt','sa','nsw'],
        'nsw':['q','v','sa'],
        'v':['sa','nsw']}
singleton_australia_states=['wa','nt','q','nsw','v','sa']
```

5)Node.py

```
class Node:
   def __init__(self,mycolor,myname=None):
       self.next = []
       self.nextnode = None
       self.mycolor = mycolor
       self.myname = myname

   def put_child(self,node):
       self.next.append(node)
```

6)singleton_csp.py

```
#CSP_singleton for American States and Australian States
states=[]
full = []
restriction_graph={}
back_propagation = 0
from copy import deepcopy
from input_files import
csp_singleton_america_states,csp_singleton_america_graph,csp_singleton_australia_state
s,csp_singleton_australia_graph
import time

"""
Intialising the states to assign the corresponding colors to each state
so that we may not run into error.
"""
class State:
```

```
    """
    This is the constructor class for intialising the state object
    with the default.
    """
    def __init__(self,name,domain,status="not visited"):
        self.state_name=name
        self.state_colour_name=None
        self.state_singleton=False
        self.state_domain=domain
        self.state_adjacent=None
        self.state_status=status


    """
    Setting the states adjacent state to find the color of the
    current state.
    """
    def set_state_adjacent(self,state_adjacent):
        self.state_adjacent=state_adjacent
    """
    Getting the adjacent states object to know about their current
    state.
    """
    def get_state_adjacent(self):
        return self.state_adjacent
    """
    Assigning color to the current state.
    """
    def assigning_colour(self,state_color):
        self.state_colour_name=state_color
    """
    Setting the state parent to know about the colors of the
    parent and comparing with the descendents.
    """
    def set_state_predeccors(self,predeccors):
        self.parent=predeccors
    """
    Setting the state domain.
    """
    def setting_state_domain(self,state_domain):
        self.state_domain=state_domain
    """
```

```
    Getting the state_domain.
    These are the getters and the setters function.
    """
    def gettng_state_domain(self):
        return self.state_domain
    """
    To find the current state is singleton or not
    """
    def check_whether_singleton(self):
        if self.state_singleton:
            return self.state_singleton
        return False


"""
This function is used to get the available state from the current
state and seeing whether the states is currently traversed or not.
"""
def states_that_currently_available(objects):
    states_that_traversed=[]
    for _ in objects:
        if _.state_status=="not visited":
            states_that_traversed.append(_)
    return states_that_traversed


"""
This function is used to select the color from the list
to find out whether there is any color available in the list or not.
"""
def colors_that_can_be_assigned(state,state_domain):
    domain_free=state_domain.copy()
    for _ in state.state_adjacent:
        color = _.colour_name
        if color!=None and (color in domain_free) :
            domain_free.remove(color)
    return domain_free


"""
This function is used to change the domain of the current state.
"""
def change_domain(state_object,color):
    for _ in state_object.state_adjacent:
        if (_.state_status=="not visited") and (color in _.state_domain):
```

```
            (_.state_domain).remove(color)

"""
This function is used to get the single domain status of the
states of the graph.
"""
def single_domain_states(objects):
    for _ in objects:
        if _.state_singleton==False and len(_.gettng_state_domain())==1:
            _.state_singleton=True
            color = _.state_domain[0]
            print("The singleton status of the current state.",_.state_domain)
            return _,color
    return None,None

"""
This function is used to generate the singleton propagation of
each states.
"""
def singleton_propagtion(objects):
    print("The singleton propagation of the states")
    singleton_states={}
    while True:
        state,color = single_domain_states(objects)
        if state==None:
            return singleton_states,"The status of the singleton propagation is
success"
        else:
            print("The singleton propagation of the",state.state_name)
            print("The domain status of the states is that",state.state_domain)
            singleton_states[state]=color
            for _ in state.state_adjacent:
                if _.state_status=="not visited" and color in _.state_domain:
                    print("The adjacent the state of the states "+_.state_name)
                    print("The domain adjacent of the state: ",_.state_domain)
                    if len(_.state_domain)==1:
                        print("The state",_.state_name)
                        #print(_.state_domain)
                        return singleton_states,"The singleton propagation is
unsuccessfull"
                    (_.state_domain).remove(color)
```

```
                    print("The state domain status of the state is that :
",_.state_domain)

"""
THIS FUNCTION IS USED TO FOR ARRANGING COLORS FOR DIFFERENET STATES
WHERE EACH STATES IS ASSIGNED WITH VALUE.
"""
def colour_arranging(domain,state_domain):
   domain_1=deepcopy(domain)
   domain_2=deepcopy(domain_1)
   state_domain=deepcopy(state_domain)
   for colour in dom:
       count=0
       for state_colour in state_domain:

           if colour!=state_colour:

               count+=1

       if count==len(state_domain):

           domain_2.remove(colour)

   return domain_2

"""
Assigning colors to the states
"""
def assigning_color(objects,color):
   print("The assigning colors to the state")

   states_updated=[]
   for _ in objects.state_adjacent:
       if _.state_status=="not visited" and color in _.state_domain:
           (_.state_domain).remove(color)
           states_updated.append(_)
   return states_updated

"""
Resetting the values of the status.
"""
def resetting_the_values(state,domain,color,states_updated):
```

```
    print("Inside the resetting method")
    color_position = domain.index(color)
    for _ in state.state_adjacent:
        if _.status=="not visited" and (_ in states_updated):
            (_.domain).insert(color_position,color)
            copy_of_domain= assigning_color(domain,_.state_domain)
            _.domain=deepcopy(copy_of_domain)

"""
Reversing the singleton propagation
"""
def resetting_the_singleton(singleton_states,domain,colour):
    print("The reversing the singleton propagation")
    for state,colour in singleton_states.items():
        temp_var = domain.index(colour)
        for _ in state.state_adjacent:
            if _.status=="not visited" and (_.is_singleton()):
                (_.domain).insert(temp_var,colour)
                domain_1 = colour_arranging(domain,_.domain)
                _.domain=domain_1
        state.state_singleton=False

"""
Constraint satisfaction problem function
"""
def constraint_satisfaction_function(state_objects,domain,states_and_colours):
        global back_propagation
        if len(states_and_colours)==len(state_objects):
            return states_and_colours

        un_assigned_states=states_that_currently_available(state_objects)

        state=un_assigned_states[0]
        print(state.state_name)
        available_domain=deepcopy(state.state_domain)
        iter1 = 0
        for colour in available_domain:
            iter1+=1
            state.state_status="visited"
            state.state_colour_name=colour
            states_and_colours[state.state_name]=colour
```

```
            updated_states=assigning_color(state,colour)
            print("total updated states",len(updated_states))
            singleton_states,status=singleton_propagtion(state_objects)
            print(len(singleton_states),status)

            if status!="unsucessful":

result=constraint_satisfaction_function(state_objects,domain,states_and_colours)
            else:
                result=status
            if result!="unsucessful":
                return result
            del states_and_colours[state.state_name]
            state.colour_name=None
            state.status="not visited"


        back_propagation+=1
        return "unsucessful"

"""
Assigning the object.
"""
def value_to_object(objects,domain_to_Values):
    temp_var = []
    dom=deepcopy(domain_to_Values)
    for _ in objects:
        cr_state_oj=State(_,dom)
        temp_var.append(cr_state_oj)
        dom=deepcopy(dom)
    return temp_var


"""
This is the main function
"""
def main():
    n=int(input("Enter the country for which states to be considered\n 1.America States
\n2.Australia States"))
    if(n==2):

        states = csp_singleton_australia_states
```

```
        restriction_graph = csp_singleton_australia_graph
    elif(n==1):

        states = csp_singleton_america_states
        restriction_graph= csp_singleton_america_graph

    domain=["blue","green","black","red"]
    state_objects = value_to_object(states,domain)

    states_and_colours={}
    for _ in state_objects:
        key = _.state_name
        values=restriction_graph[key]
        neighbours=[]
        for value in values:
            objects=[objects_form for objects_form in state_objects if
objects_form.state_name==value ]
            neighbours.append(objects[0])

        _.set_state_adjacent(neighbours)

state_constraint=constraint_satisfaction_function(state_objects,domain,states_and_colo
urs)
    print(state_constraint)
start_time = time.time()
main()
end_time = time.time()
print("NUMBER OF BACKTRACKING HAPPENED IN THE PROGRAM : " + str(back_propagation))
print()
print("THE TOTAL RUNNING TIME OF THE PROGRAM IS :" + str(end_time - start_time) + "
SECONDS")
```

7) singleton_with_heuristic_csp.py

```
import time
from input_files import
singleton_america_graph,singleton_america_states,singleton_australia_graph,singleton_a
ustralia_states
import random
from Node import Node

"""
Intialising the default values so that they can be used for finding
the answer for the various functions.
"""
colors_for_states = []
states = []
statedict={}
states_present = []
back_propagation = 0

"""
Intialising the states to assign the corresponding colors to each state
so that we may not run into error.
"""
class State:
    """
    This is the constructor class for intialising the state object
    with the default.
    """
    def __init__(self,name,domain,status="not visited"):
        self.state_name=name
        self.state_colour_name=None
        self.state_singleton=False
        self.state_domain=domain
        self.state_adjacent=None
        self.state_status=status


    """
    Setting the states adjacent state to find the color of the
    current state.
    """
```

```
    def set_state_adjacent(self,state_adjacent):
        self.state_adjacent=state_adjacent
    """
    Getting the adjacent states object to know about their current
    state.
    """
    def get_state_adjacent(self):
        return self.state_adjacent
    """
    Assigning color to the current state.
    """
    def assigning_colour(self,state_color):
        self.state_colour_name=state_color
    """
    Setting the state parent to know about the colors of the
    parent and comparing with the descendents.
    """
    def set_state_predeccors(self,predeccors):
        self.parent=predeccors
    """
    Setting the state domain.
    """
    def setting_state_domain(self,state_domain):
        self.state_domain=state_domain
    """
    Getting the state_domain.
    These are the getters and the setters function.
    """
    def gettng_state_domain(self):
        return self.state_domain
    """
    To find the current state is singleton or not
    """
    def check_whether_singleton(self):
        if self.state_singleton:
            return self.state_singleton
        return False


"""
This function is used for initialisation of colors.
"""
def initialisation_of_colors(number_of_colors):
```

```
    for _ in range(number_of_colors):
        colors_for_states.append(random.randint(0,255))
"""
This function is used to assign the states with the
random colors in the begning.
"""
def assigning_random_colors(number_of_colors):
    return tuple(colors_for_states)


"""
THIS FUNCTION IS USED TO GET THE STATE THEIR CURRENT
COLOR.
"""
def getcolors(nodes_state,dict_of_state1):
    colors_of_list = []
    for _ in nodes_state:
        colors_of_list.append(dict_of_state1.get(_,""))
    return colors_of_list

"""
This function is used to generate the colors for the state and to
create a node and assigning values to it.
"""
def assigning_of_colors(objects,k,number_of_colors,given_colors):
    temp_var = []
    for _ in range(number_of_colors):
        temp_var.append(Node(given_colors[_],objects[k]))

    return temp_var

"""
THIS IS THE FUNCTION WHERE THE SINGLETON FUNCTION IS USED AND IMPLEMENTED
SUCCESSFULLY.

"""
def
singleton(dict_of_state1,dict_of_state,number_of_colors,pre_cond_state,states,count):

    global back_propagation
    states_present.append(dict_of_state1.copy())
    for i in range(len(pre_cond_state.next)):
```

```
        dict_of_state1[pre_cond_state.next[0].myname] = pre_cond_state.next[i].mycolor
        if dict_of_state1.get(pre_cond_state.next[0].myname) in
getcolors(dict_of_state[pre_cond_state.next[0].myname],dict_of_state1):
            #print("continued")
            continue
         if count == len(states) - 1:
            return 1,dict_of_state1

        temp_colorlist = colors_for_states.copy()
        remove_colors = getcolors(states[count+1],dict_of_state1)
        temp_colorlist = [x for x in temp_colorlist if x not in remove_colors]
        pre_cond_state.next[i].next =
assigning_of_colors(states,count+1,number_of_colors,temp_colorlist)

        ans =
singleton(dict_of_state1,dict_of_state,number_of_colors,pre_cond_state.next[i],states,
count+1)
        if ans[0] == 1:
            return 1,dict_of_state1

        continue

    back_propagation +=1

    return 0,dict_of_state1


"""
This function is used for intialising the states with the
random colors so that we may have a start point for everything.
"""
def init(objects,statedict,number_of_colors):
    colors_used = colors_for_states
    head = Node(colors_used[0],objects[0])
    for _ in range(number_of_colors):
        head.put_child(Node(colors_used[_],objects[0]))

    return head
if __name__ == "__main__":
    colors_for_states = ["red","blue","green","black"]
    number_of_colors = 4
    initialisation_of_colors(number_of_colors)
```

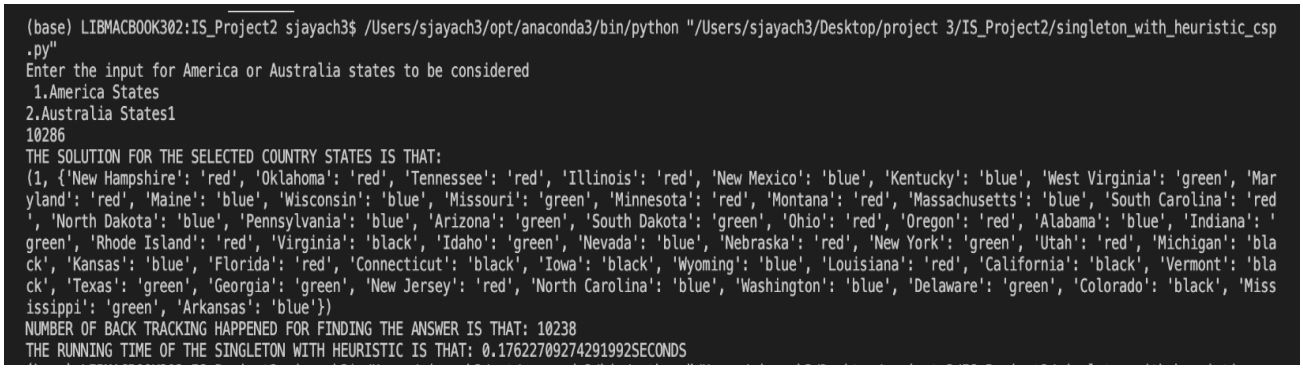
```
   n=int(input("Enter the input for America or Australia states to be considered \n
1.America States \n2.Australia States"))
   if(n==1):
       statedict = singleton_america_graph

       states = singleton_america_states
   elif(n==2):
       states=singleton_australia_states
       statedict  =singleton_australia_graph

   head = init(states,statedict,number_of_colors)
   dict_of_states = {}
   now = time.time()
   result = singleton(dict_of_states,statedict,number_of_colors,head,states,0)
   then = time.time()
   iter = 0
   for _ in result[1]:
       iter+=1
       if result[1][_] in getcolors(statedict[_],dict_of_states):
           print("THE COLORS COULD NOT BE ASSIGNED TO ALL THE VARIABLES.")
   print(len(states_present))
   print("THE SOLUTION FOR THE SELECTED COUNTRY STATES IS THAT: ")
   print(result)
   print("NUMBER OF BACK TRACKING HAPPENED FOR FINDING THE ANSWER IS THAT: "+
str(back_propagation))
   print("THE RUNNING TIME OF THE SINGLETON WITH HEURISTIC IS THAT: " + str(then -
now) + "SECONDS")
```

RESULTS:

(The outputs below have screenshots for first two trials and the values have been tabulated for four trials in the table)

Results for singleton heuristic for America:

Trial 1



Trial 2

Trials	Number of Backtracks	Time taken in second
Trial 1	10238	0.1762270927429199
Trial 2	10238	0.1870741844177246
Trial 3	10238	0.1947384857383928
Trial 4	10238	0.1736458499838748

Results for singleton heuristic for Australia:

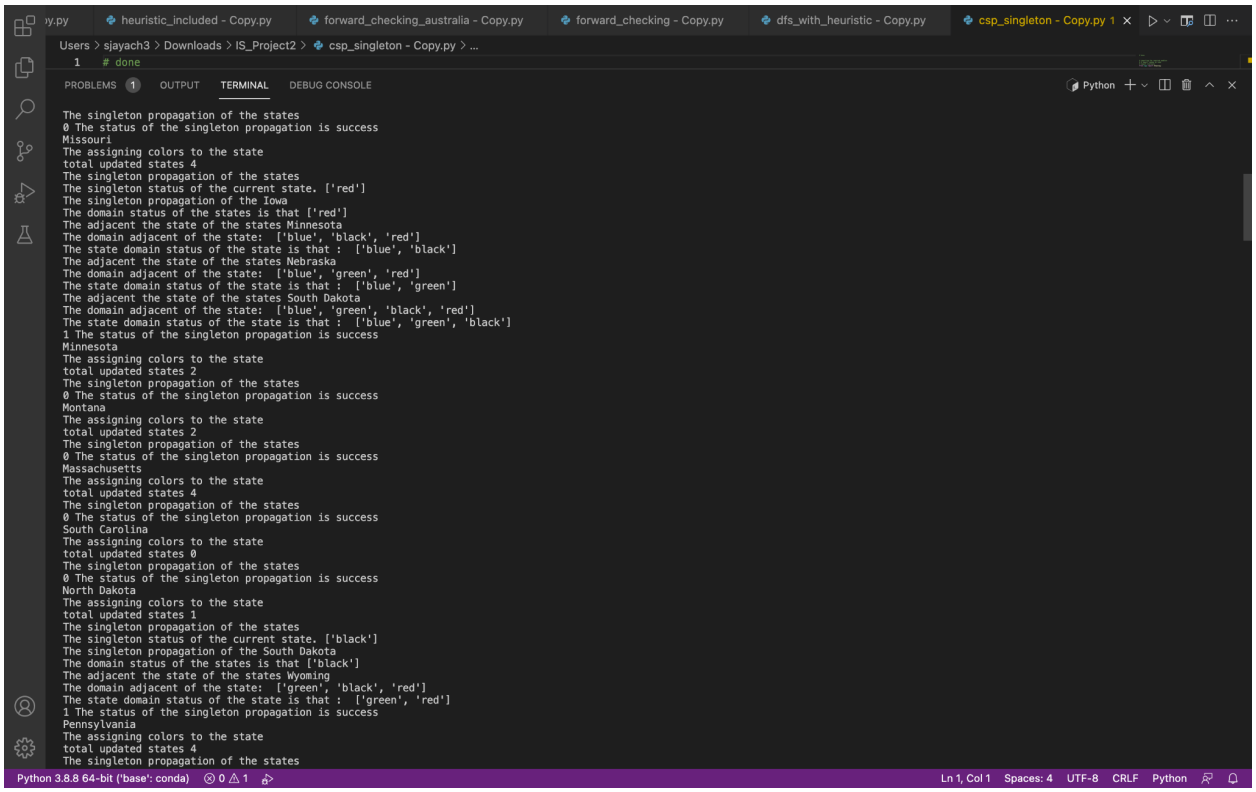
Trial1

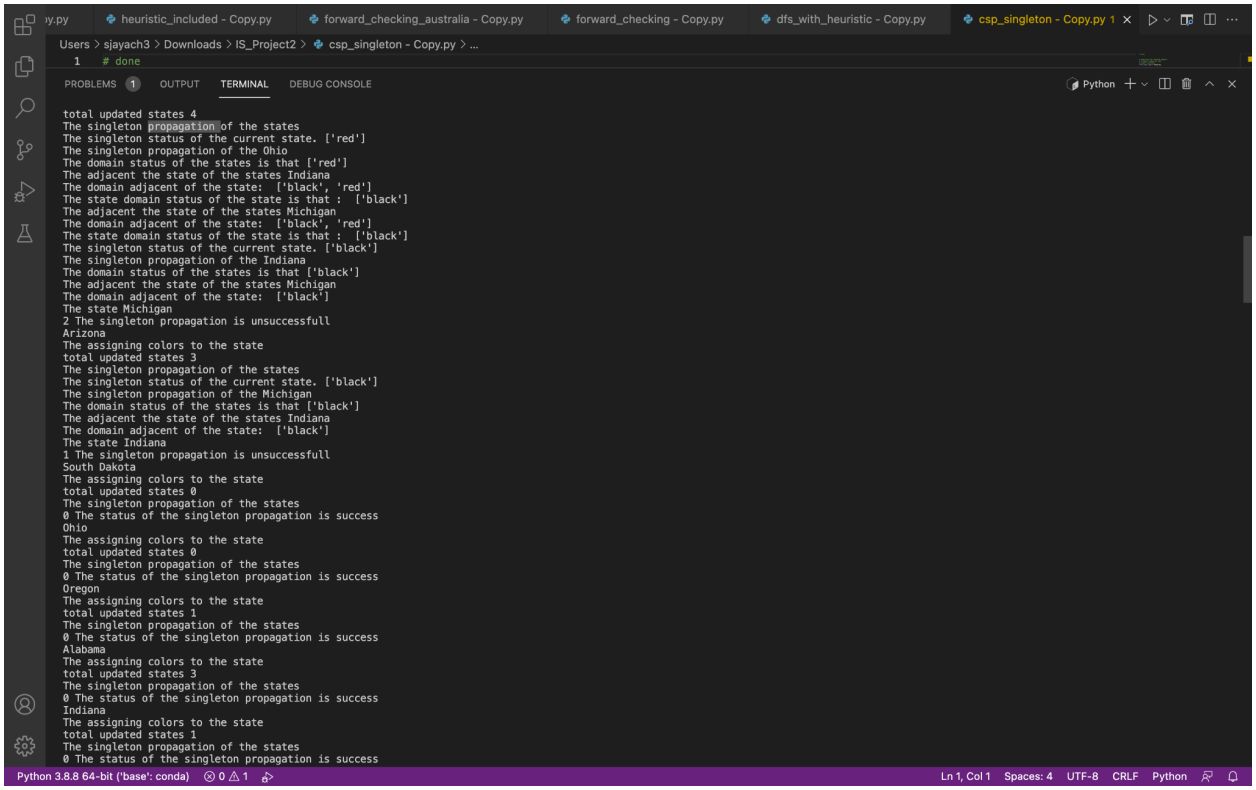
Trial2

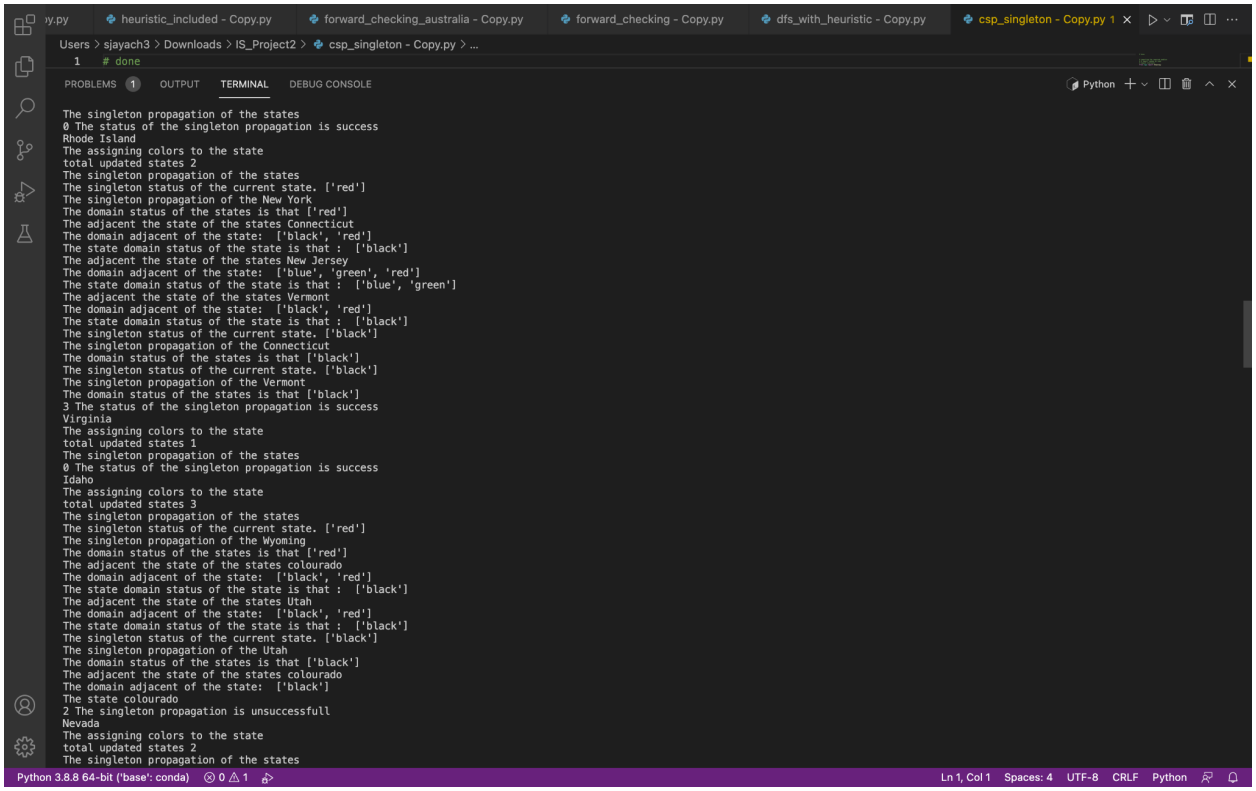
Trials	Number of Backtracks	Time taken in second
Trial 1	0	0.0006167888864135742
Trial 2	0	0.0006170272827148438
Trial 3	0	0.0006123455345453456
Trial 4	0	0.0006143453645675765

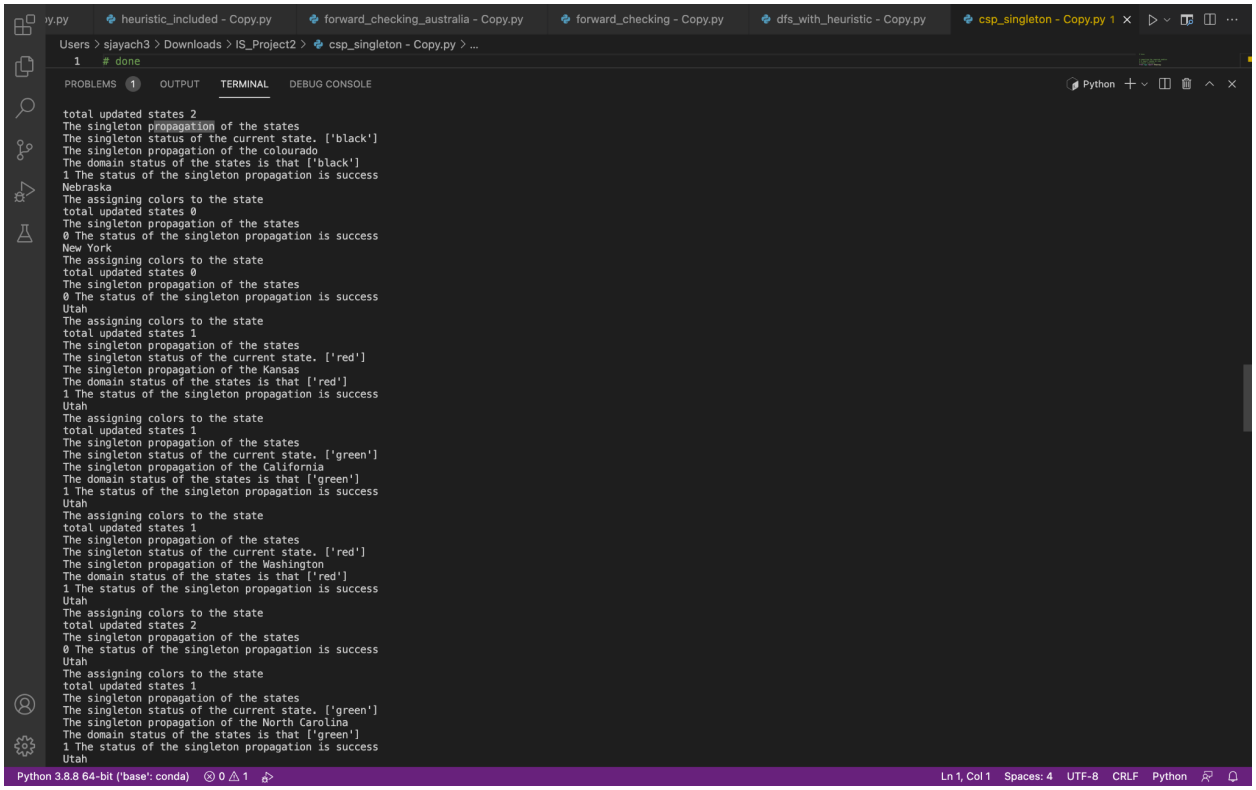
Results for CSP_singleton for America States:

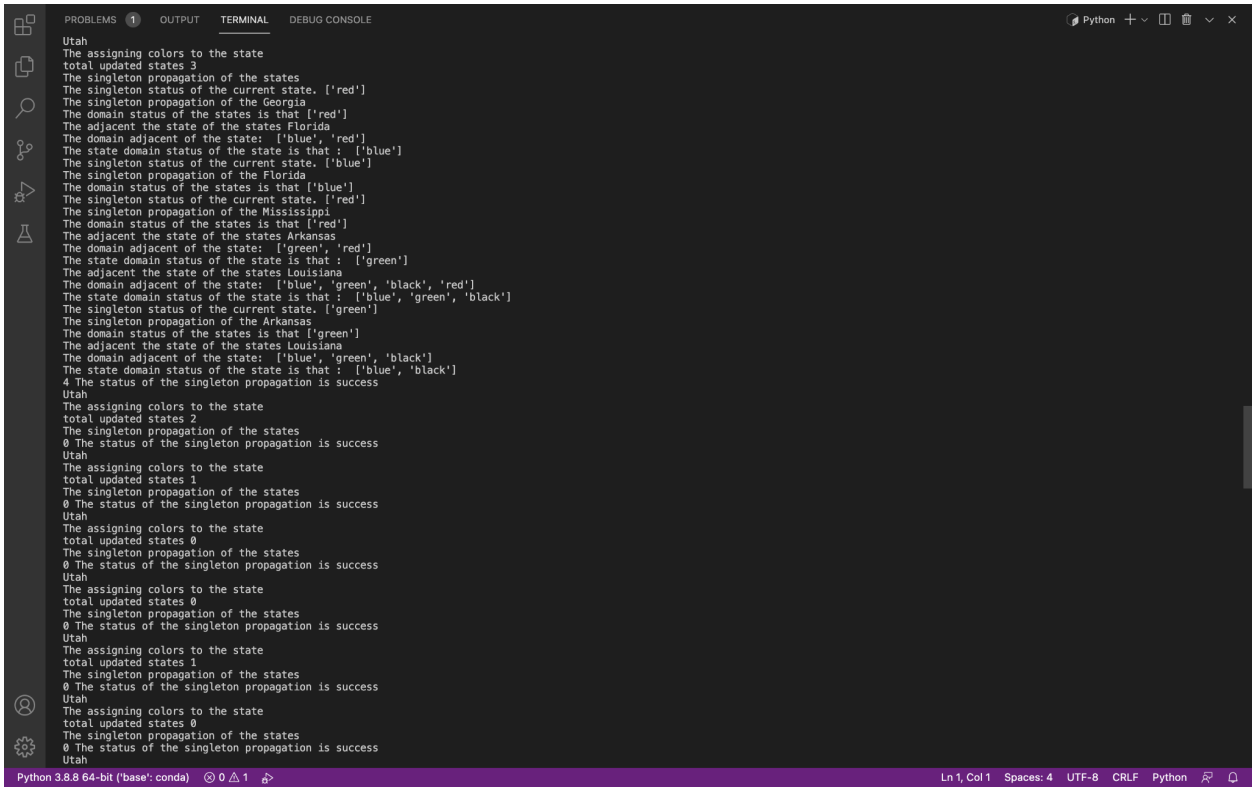
Trial 1




heuristic_included - Copy.py
forward_checking_australia - Copy.py
forward_checking - Copy.py
dfs_with_heuristic - Copy.py
csp_singleton - Copy.py 1
Users > sjayach3 > Downloads > IS_Project2 > csp_singleton - Copy.py > ...
1 # done
PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE
Python
total updated states 4
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Ohio
The domain status of the states is that ['red']
The adjacent the state of the states Indiana
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The adjacent the state of the states Michigan
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The singleton status of the current state. ['black']
The singleton propagation of the Indiana
The domain status of the states is that ['black']
The adjacent the state of the states Michigan
The domain adjacent of the state: ['black']
The state Michigan
2 The singleton propagation is unsuccessfull
Arizona
The assigning colors to the state
total updated states 3
The singleton propagation of the states
The singleton status of the current state. ['black']
The singleton propagation of the Michigan
The domain status of the states is that ['black']
The adjacent the state of the states Indiana
The domain adjacent of the state: ['black']
The state Indiana
1 The singleton propagation is unsuccessfull
South Dakota
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Ohio
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Oregon
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Alabama
The assigning colors to the state
total updated states 3
The singleton propagation of the states
0 The status of the singleton propagation is success
Indiana
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Python 3.8.8 64-bit ('base': conda)
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python


heuristic_included - Copy.py
forward_checking_australia - Copy.py
forward_checking - Copy.py
dfs_with_heuristic - Copy.py
csp_singleton - Copy.py 1
Users > sjayach3 > Downloads > IS_Project2 > csp_singleton - Copy.py > ...
1 # done
PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE
Python
The singleton propagation of the states
0 The status of the singleton propagation is success
Rhode Island
The assigning colors to the state
total updated states 2
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the New York
The domain status of the states is that ['red']
The adjacent the state of the states Connecticut
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The adjacent the state of the states New Jersey
The domain adjacent of the state: ['blue', 'green', 'red']
The state domain status of the state is that : ['blue', 'green']
The adjacent the state of the states Vermont
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The singleton status of the current state. ['black']
The singleton propagation of the Connecticut
The domain status of the states is that ['black']
The singleton status of the current state. ['black']
The singleton propagation of the Vermont
The domain status of the states is that ['black']
3 The status of the singleton propagation is success
Virginia
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Idaho
The assigning colors to the state
total updated states 3
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Wyoming
The domain status of the states is that ['red']
The adjacent the state of the states colourado
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The adjacent the state of the states Utah
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The singleton status of the current state. ['black']
The singleton propagation of the Utah
The domain status of the states is that ['black']
The adjacent the state of the states colourado
The domain adjacent of the state: ['black']
The state colourado
2 The singleton propagation is unsuccessfull
Nevada
The assigning colors to the state
total updated states 2
The singleton propagation of the states
Python 3.8.8 64-bit ('base': conda)
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python


heuristic_included - Copy.py
forward_checking_australia - Copy.py
forward_checking - Copy.py
dfs_with_heuristic - Copy.py
csp_singleton - Copy.py 1
Users > sjayach3 > Downloads > IS_Project2 > csp_singleton - Copy.py > ...
1 # done
PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE
Python
total updated states 2
The singleton propagation of the states
The singleton status of the current state. ['black']
The singleton propagation of the colourado
The domain status of the states is that ['black']
1 The status of the singleton propagation is success
Nebraska
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
New York
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Kansas
The domain status of the states is that ['red']
1 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['green']
The singleton propagation of the California
The domain status of the states is that ['green']
1 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Washington
The domain status of the states is that ['red']
1 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 2
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['green']
The singleton propagation of the North Carolina
The domain status of the states is that ['green']
1 The status of the singleton propagation is success
Utah
Python 3.8.8 64-bit ('base': conda)
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python


PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE
Python
Utah
The assigning colors to the state
total updated states 3
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Georgia
The domain status of the states is that ['red']
The adjacent the state of the states Florida
The domain adjacent of the state: ['blue', 'red']
The state domain status of the state is that : ['blue']
The singleton status of the current state. ['blue']
The singleton propagation of the Florida
The domain status of the states is that ['blue']
The singleton status of the current state. ['red']
The singleton propagation of the Mississippi
The domain status of the states is that ['red']
The adjacent the state of the states Arkansas
The domain adjacent of the state: ['green', 'red']
The state domain status of the state is that : ['green']
The adjacent the state of the states Louisiana
The domain adjacent of the state: ['blue', 'green', 'black', 'red']
The state domain status of the state is that : ['blue', 'green', 'black']
The singleton status of the current state. ['green']
The singleton propagation of the Arkansas
The domain status of the states is that ['green']
The adjacent the state of the states Louisiana
The domain adjacent of the state: ['blue', 'green', 'black']
The state domain status of the state is that : ['blue', 'black']
4 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 2
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
Python 3.8.8 64-bit ('base': conda)
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python

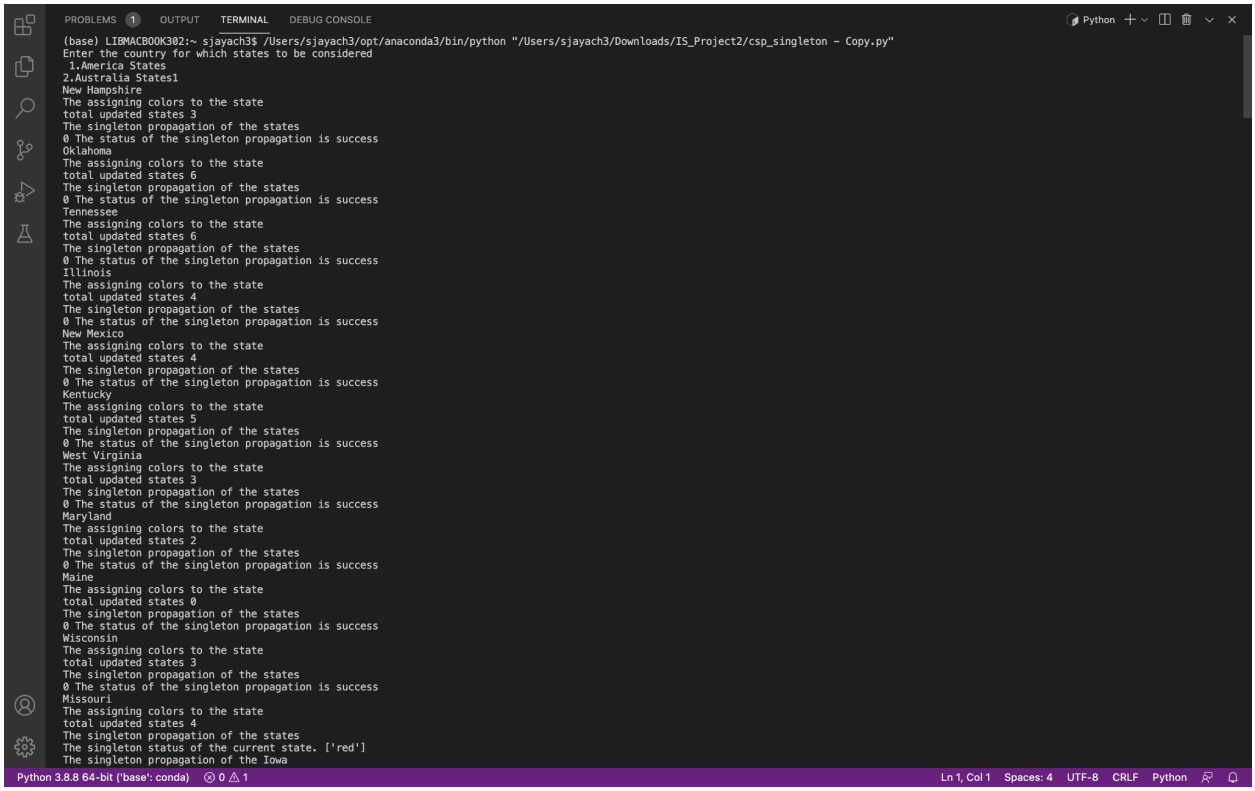
```
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['blue']
The singleton propagation of the Delaware
The domain status of the states is that ['blue']
The adjacent the state of the states New Jersey
The domain adjacent of the state:  ['blue', 'green']
The state domain status of the state is that :  ['green']
The singleton status of the current state. ['green']
The singleton propagation of the New Jersey
The domain status of the states is that ['green']
2 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
```

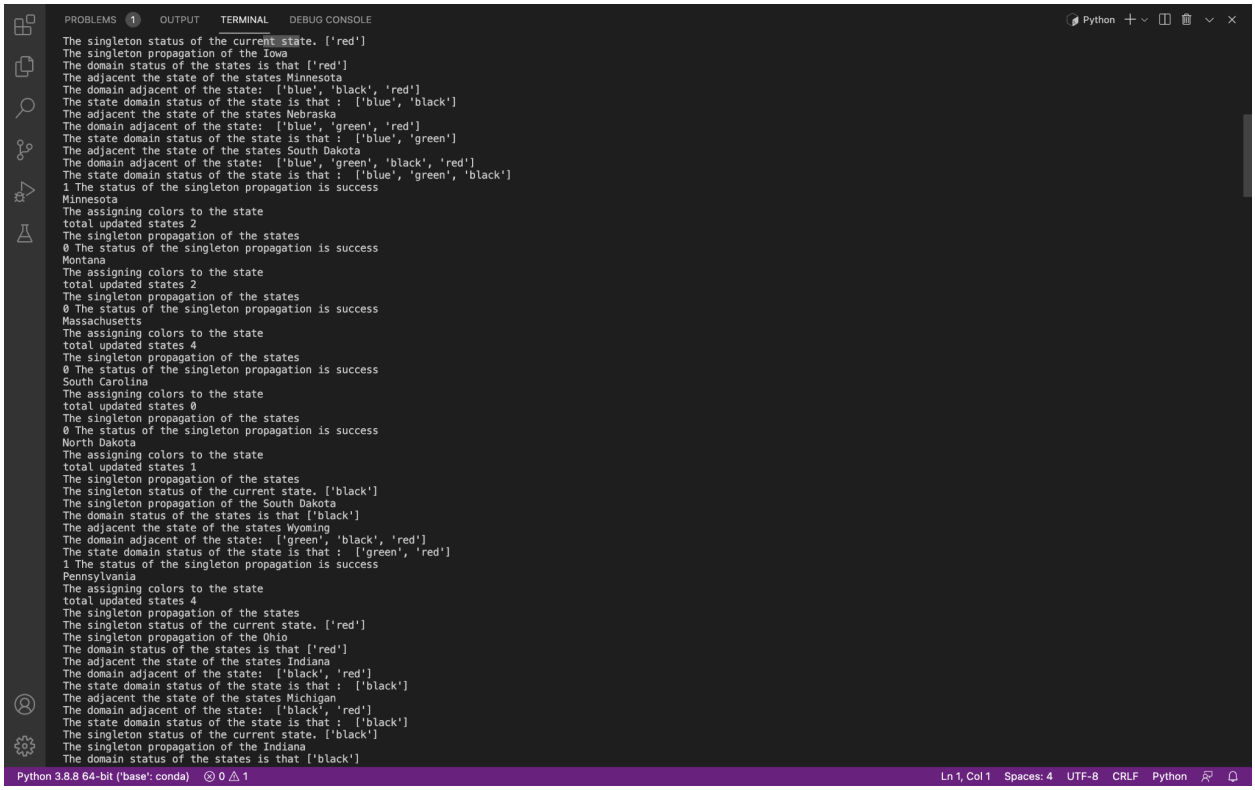
```
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 2
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
```

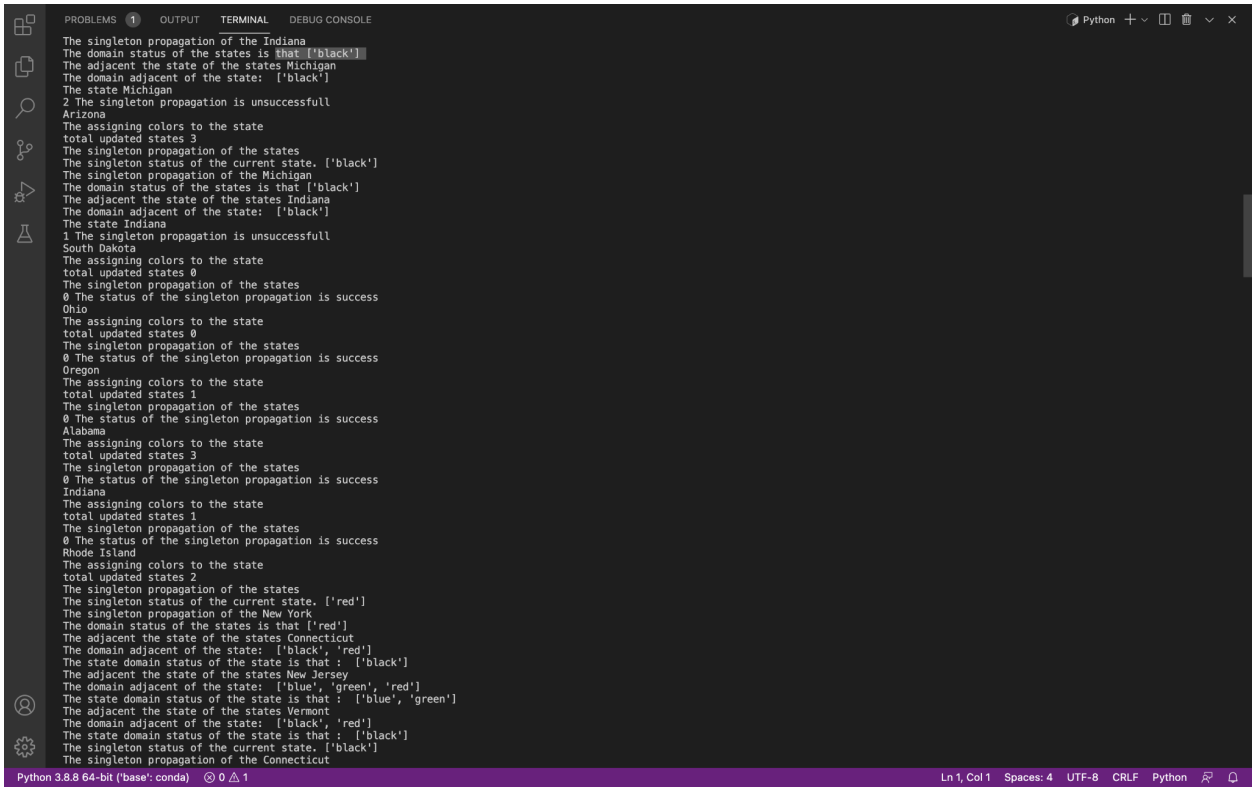
```
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Texas
The domain status of the states is that ['red']
1 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
```

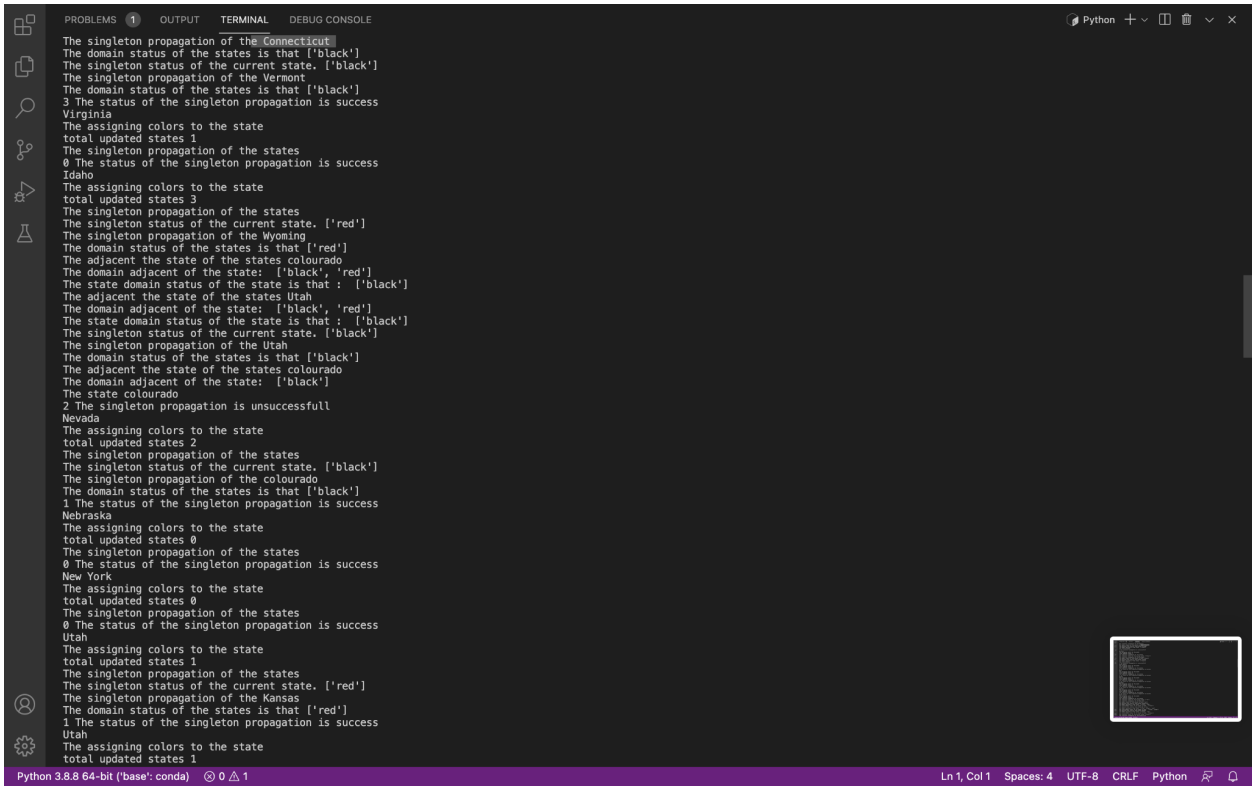
```
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
unsucessful
Time of execution = 1.9914278984069824 in seconds
Number of back_propagation= 81
(base) LIBMACBOOK302:~ sjayach3$
```

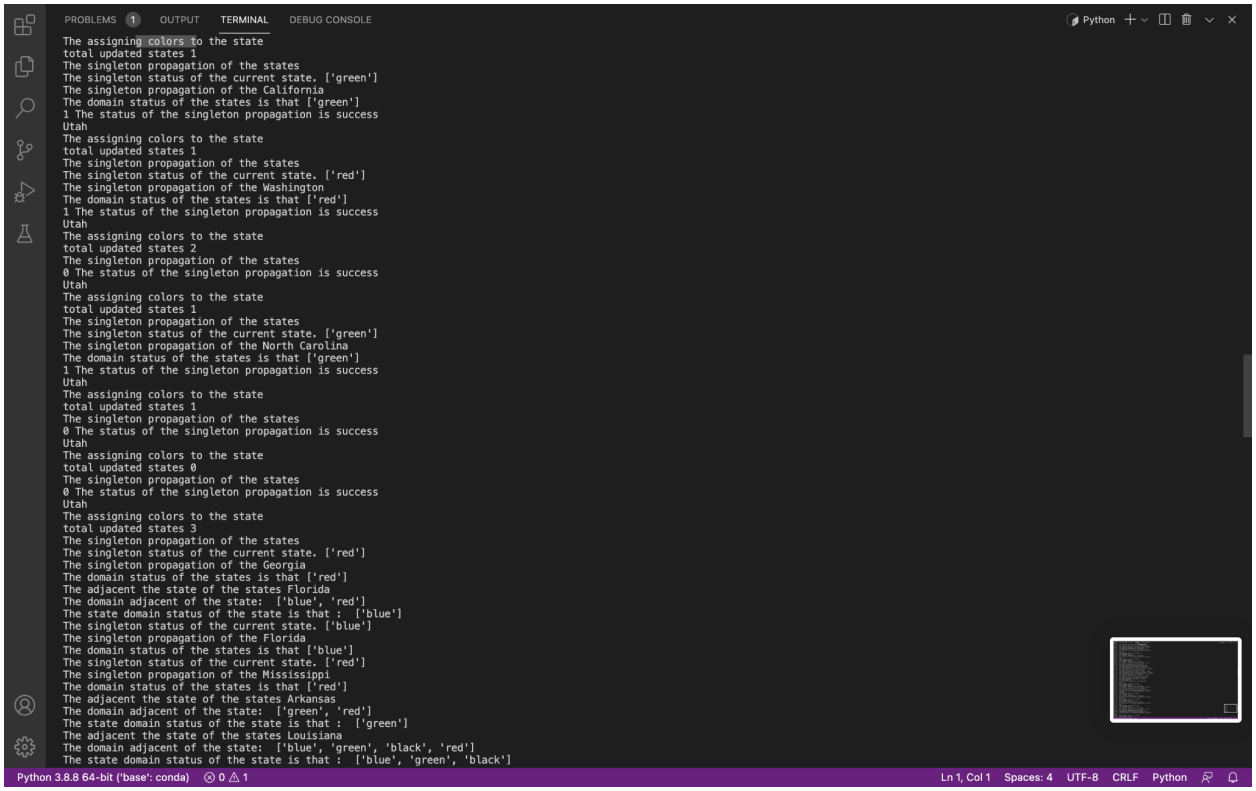
Trial 2

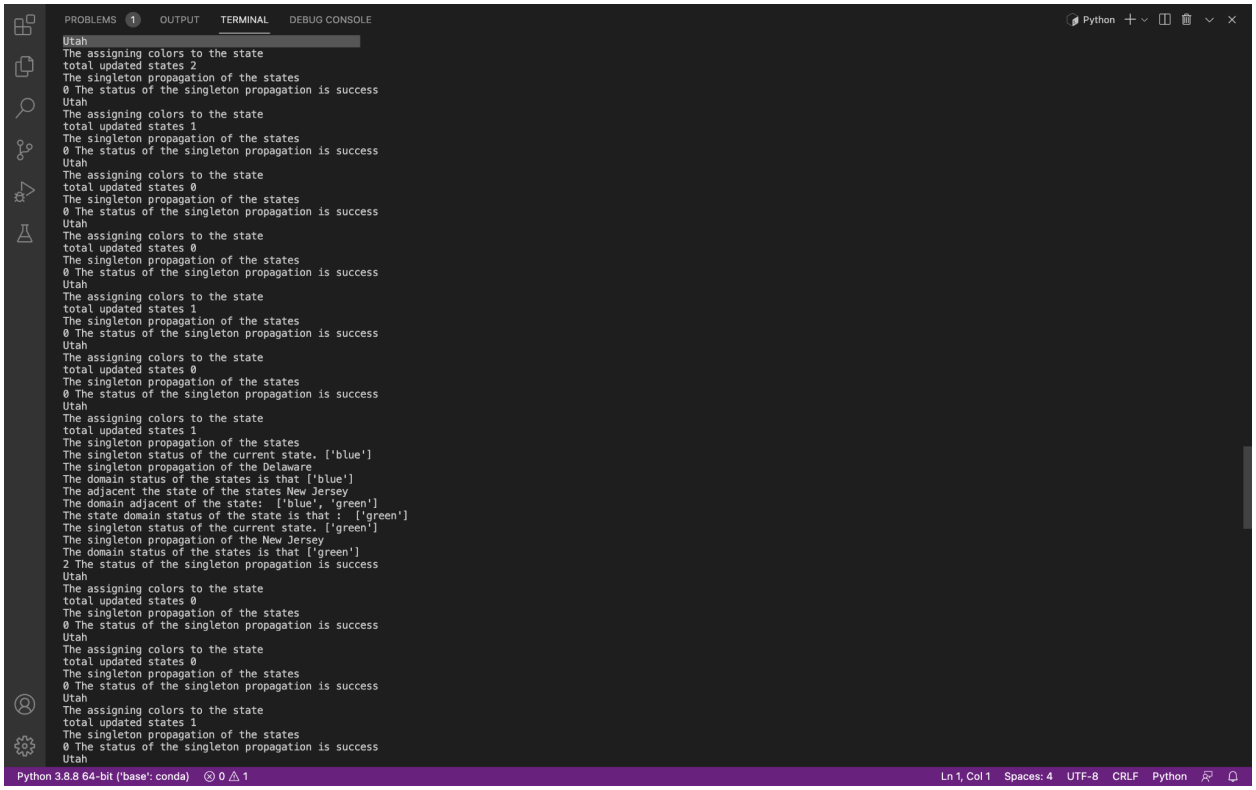

PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
(base) LIBMACBOOK302:~ sjayach3$ /Users/sjayach3/opt/anaconda3/bin/python "/Users/sjayach3/Downloads/IS_Project2/csp_singleton - Copy.py"
Enter the country for which states to be considered
1.America States
2.Australia States1
New Hampshire
The assigning colors to the state
total updated states 3
The singleton propagation of the states
0 The status of the singleton propagation is success
Oklahoma
The assigning colors to the state
total updated states 6
The singleton propagation of the states
0 The status of the singleton propagation is success
Tennessee
The assigning colors to the state
total updated states 6
The singleton propagation of the states
0 The status of the singleton propagation is success
Illinois
The assigning colors to the state
total updated states 4
The singleton propagation of the states
0 The status of the singleton propagation is success
New Mexico
The assigning colors to the state
total updated states 4
The singleton propagation of the states
0 The status of the singleton propagation is success
Kentucky
The assigning colors to the state
total updated states 5
The singleton propagation of the states
0 The status of the singleton propagation is success
West Virginia
The assigning colors to the state
total updated states 3
The singleton propagation of the states
0 The status of the singleton propagation is success
Maryland
The assigning colors to the state
total updated states 2
The singleton propagation of the states
0 The status of the singleton propagation is success
Maine
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Wisconsin
The assigning colors to the state
total updated states 3
The singleton propagation of the states
0 The status of the singleton propagation is success
Missouri
The assigning colors to the state
total updated states 4
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Iowa
Python 3.8.8 64-bit ('base': conda)
0 1
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python

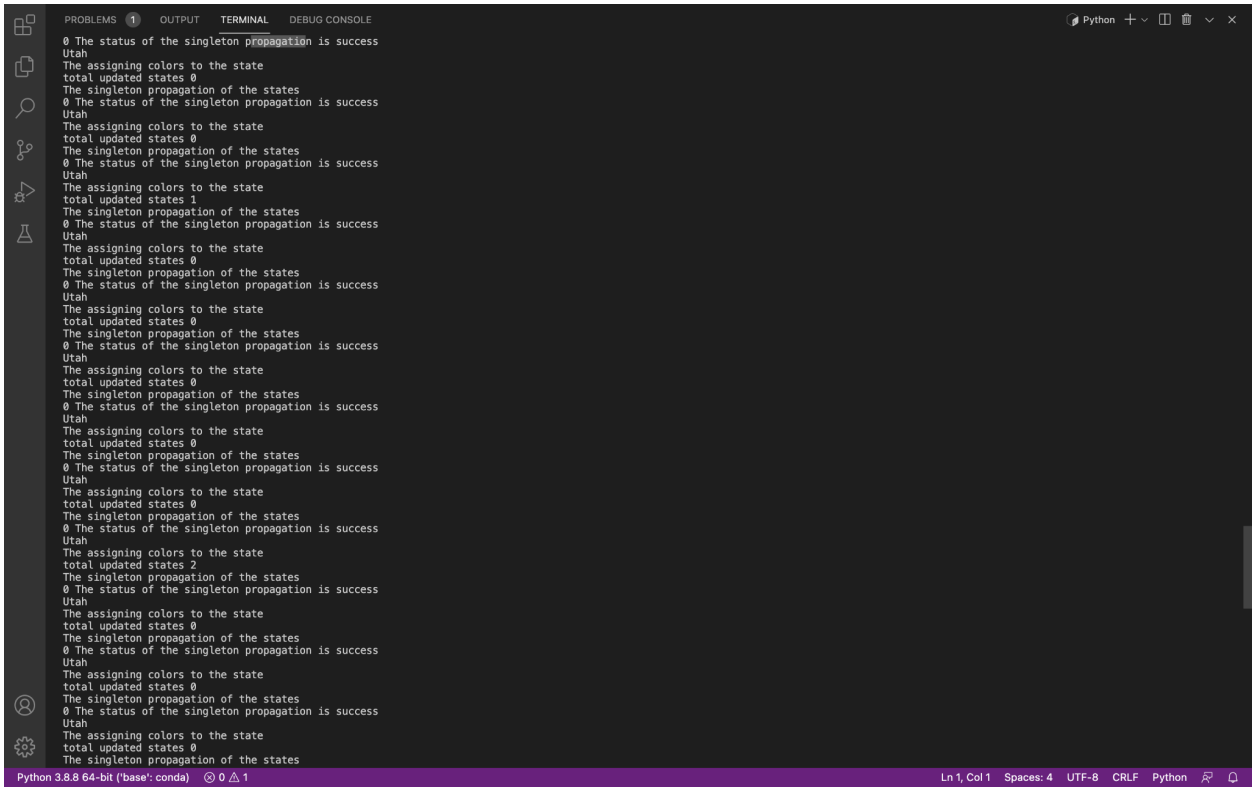

PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
The singleton status of the current state. ['red']
The singleton propagation of the Iowa
The domain status of the states is that ['red']
The adjacent the state of the states Minnesota
The domain adjacent of the state: ['blue', 'black', 'red']
The state domain status of the state is that : ['blue', 'black']
The adjacent the state of the states Nebraska
The domain adjacent of the state: ['blue', 'green', 'red']
The state domain status of the state is that : ['blue', 'green']
The adjacent the state of the states South Dakota
The domain adjacent of the state: ['blue', 'green', 'black', 'red']
The state domain status of the state is that : ['blue', 'green', 'black']
1 The status of the singleton propagation is success
Minnesota
The assigning colors to the state
total updated states 2
The singleton propagation of the states
0 The status of the singleton propagation is success
Montana
The assigning colors to the state
total updated states 2
The singleton propagation of the states
0 The status of the singleton propagation is success
Massachusetts
The assigning colors to the state
total updated states 4
The singleton propagation of the states
0 The status of the singleton propagation is success
South Carolina
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
North Dakota
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['black']
The singleton propagation of the South Dakota
The domain status of the states is that ['black']
The adjacent the state of the states Wyoming
The domain adjacent of the state: ['green', 'black', 'red']
The state domain status of the state is that : ['green', 'red']
1 The status of the singleton propagation is success
Pennsylvania
The assigning colors to the state
total updated states 4
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Ohio
The domain status of the states is that ['red']
The adjacent the state of the states Indiana
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The adjacent the state of the states Michigan
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The singleton status of the current state. ['black']
The singleton propagation of the Indiana
The domain status of the states is that ['black']
Python 3.8.8 64-bit ('base': conda)
0 1
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python

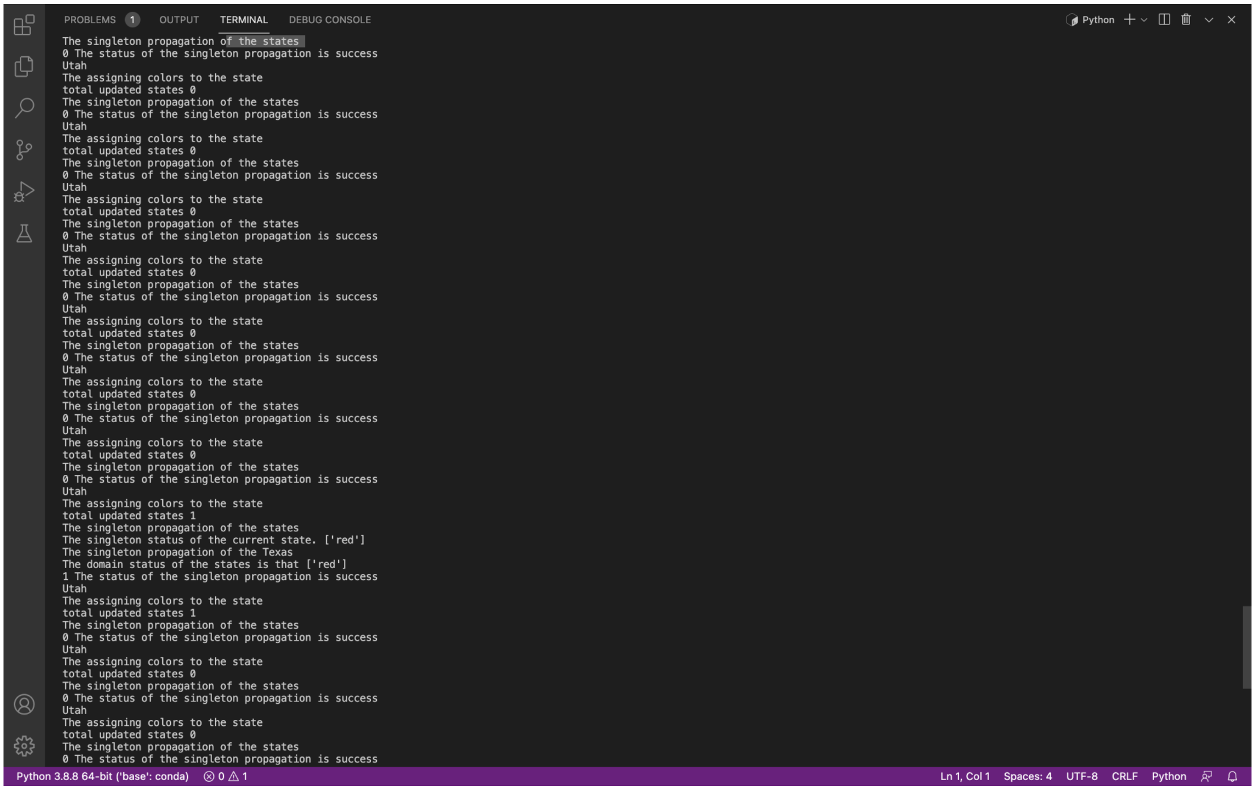

PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
The singleton propagation of the Indiana
The domain status of the states is that ['black']
The adjacent the state of the states Michigan
The domain adjacent of the state: ['black']
The state Michigan
2 The singleton propagation is unsuccessfull
Arizona
The assigning colors to the state
total updated states 3
The singleton propagation of the states
The singleton status of the current state. ['black']
The singleton propagation of the Michigan
The domain status of the states is that ['black']
The adjacent the state of the states Indiana
The domain adjacent of the state: ['black']
The state Indiana
1 The singleton propagation is unsuccessfull
South Dakota
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Ohio
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Oregon
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Alabama
The assigning colors to the state
total updated states 3
The singleton propagation of the states
0 The status of the singleton propagation is success
Indiana
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Rhode Island
The assigning colors to the state
total updated states 2
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the New York
The domain status of the states is that ['red']
The adjacent the state of the states Connecticut
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The adjacent the state of the states New Jersey
The domain adjacent of the state: ['blue', 'green', 'red']
The state domain status of the state is that : ['blue', 'green']
The adjacent the state of the states Vermont
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The singleton status of the current state. ['black']
The singleton propagation of the Connecticut
Python 3.8.8 64-bit ('base': conda)
0 1
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python

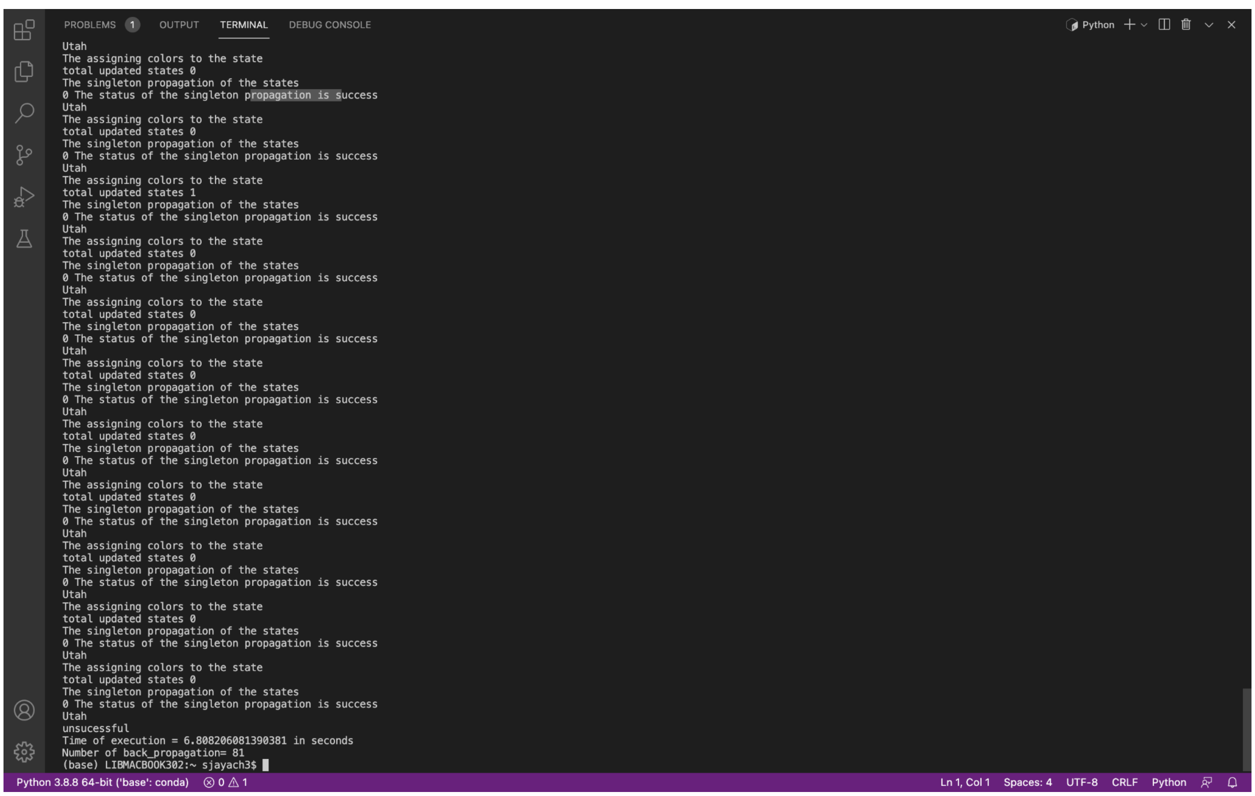

PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
The singleton propagation of the Connecticut
The domain status of the states is that ['black']
The singleton status of the current state. ['black']
The singleton propagation of the Vermont
The domain status of the states is that ['black']
3 The status of the singleton propagation is success
Virginia
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Idaho
The assigning colors to the state
total updated states 3
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Wyoming
The domain status of the states is that ['red']
The adjacent the state of the states colourado
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The adjacent the state of the states Utah
The domain adjacent of the state: ['black', 'red']
The state domain status of the state is that : ['black']
The singleton status of the current state. ['black']
The singleton propagation of the Utah
The domain status of the states is that ['black']
The adjacent the state of the states colourado
The domain adjacent of the state: ['black']
The state colourado
2 The singleton propagation is unsuccessfull
Nevada
The assigning colors to the state
total updated states 2
The singleton propagation of the states
The singleton status of the current state. ['black']
The singleton propagation of the colourado
The domain status of the states is that ['black']
1 The status of the singleton propagation is success
Nebraska
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
New York
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Kansas
The domain status of the states is that ['red']
1 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
Python 3.8.8 64-bit ('base': conda)
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python


PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['green']
The singleton propagation of the California
The domain status of the states is that ['green']
1 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Washington
The domain status of the states is that ['red']
1 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 2
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['green']
The singleton propagation of the North Carolina
The domain status of the states is that ['green']
1 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 3
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Georgia
The domain status of the states is that ['red']
The adjacent the state of the states Florida
The domain adjacent of the state: ['blue', 'red']
The state domain status of the state is that : ['blue']
The singleton status of the current state. ['blue']
The singleton propagation of the Florida
The domain status of the states is that ['blue']
The singleton status of the current state. ['red']
The singleton propagation of the Mississippi
The domain status of the states is that ['red']
The adjacent the state of the states Arkansas
The domain adjacent of the state: ['green', 'red']
The state domain status of the state is that : ['green']
The adjacent the state of the states Louisiana
The domain adjacent of the state: ['blue', 'green', 'black', 'red']
The state domain status of the state is that : ['blue', 'green', 'black']
Python 3.8.8 64-bit ('base': conda)
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python


PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
Utah
The assigning colors to the state
total updated states 2
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['blue']
The singleton propagation of the Delaware
The domain status of the states is that ['blue']
The adjacent the state of the states New Jersey
The domain adjacent of the state: ['blue', 'green']
The state domain status of the state is that : ['green']
The singleton status of the current state. ['green']
The singleton propagation of the New Jersey
The domain status of the states is that ['green']
2 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
Python 3.8.8 64-bit ('base': conda)
0 1
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python


PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 2
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
Python 3.8.8 64-bit ('base': conda)
0 1
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python

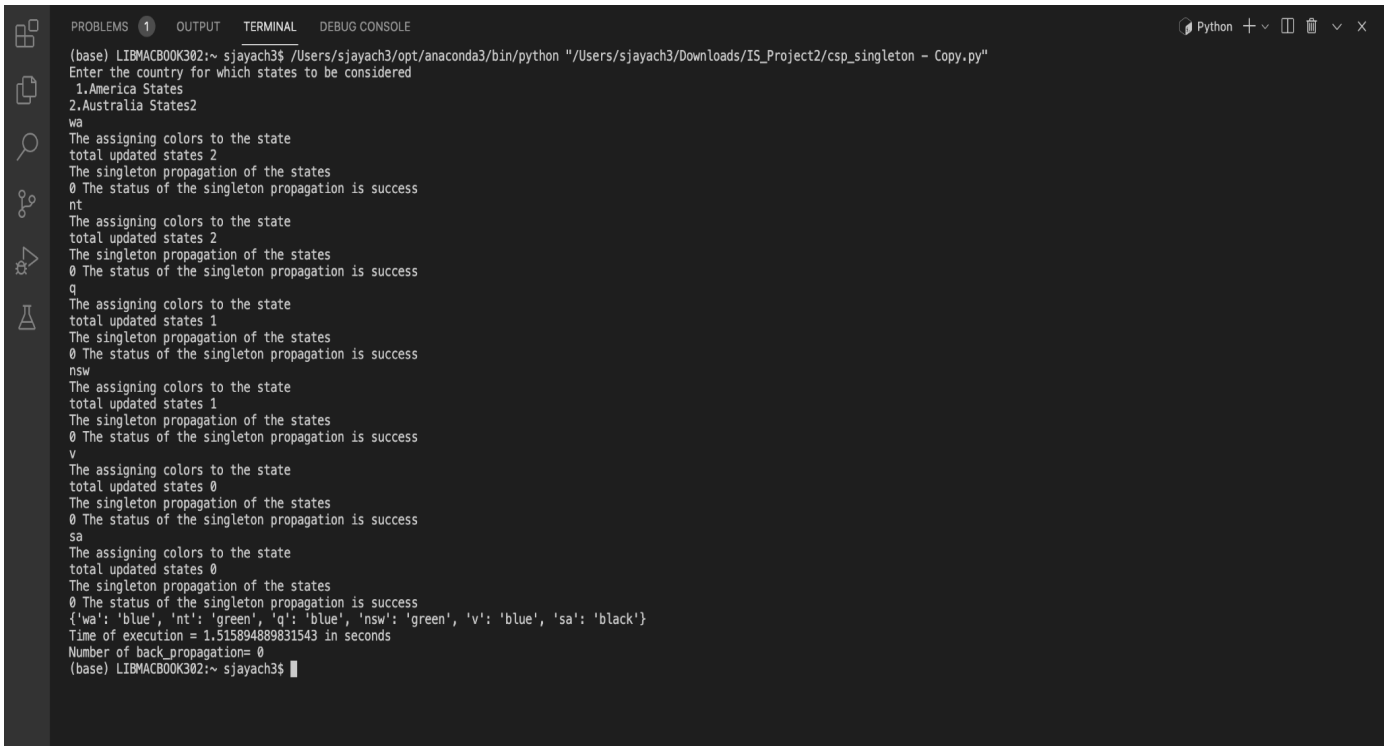

PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
The singleton status of the current state. ['red']
The singleton propagation of the Texas
The domain status of the states is that ['red']
1 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Python 3.8.8 64-bit ('base': conda)
0 1
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python


PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 1
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
The assigning colors to the state
total updated states 0
The singleton propagation of the states
0 The status of the singleton propagation is success
Utah
unsucessful
Time of execution = 6.808206081390381 in seconds
Number of back_propagation= 81
(base) LIBMACBOOK302:~ sjayach3$
Python 3.8.8 64-bit ('base': conda)
0 1
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python

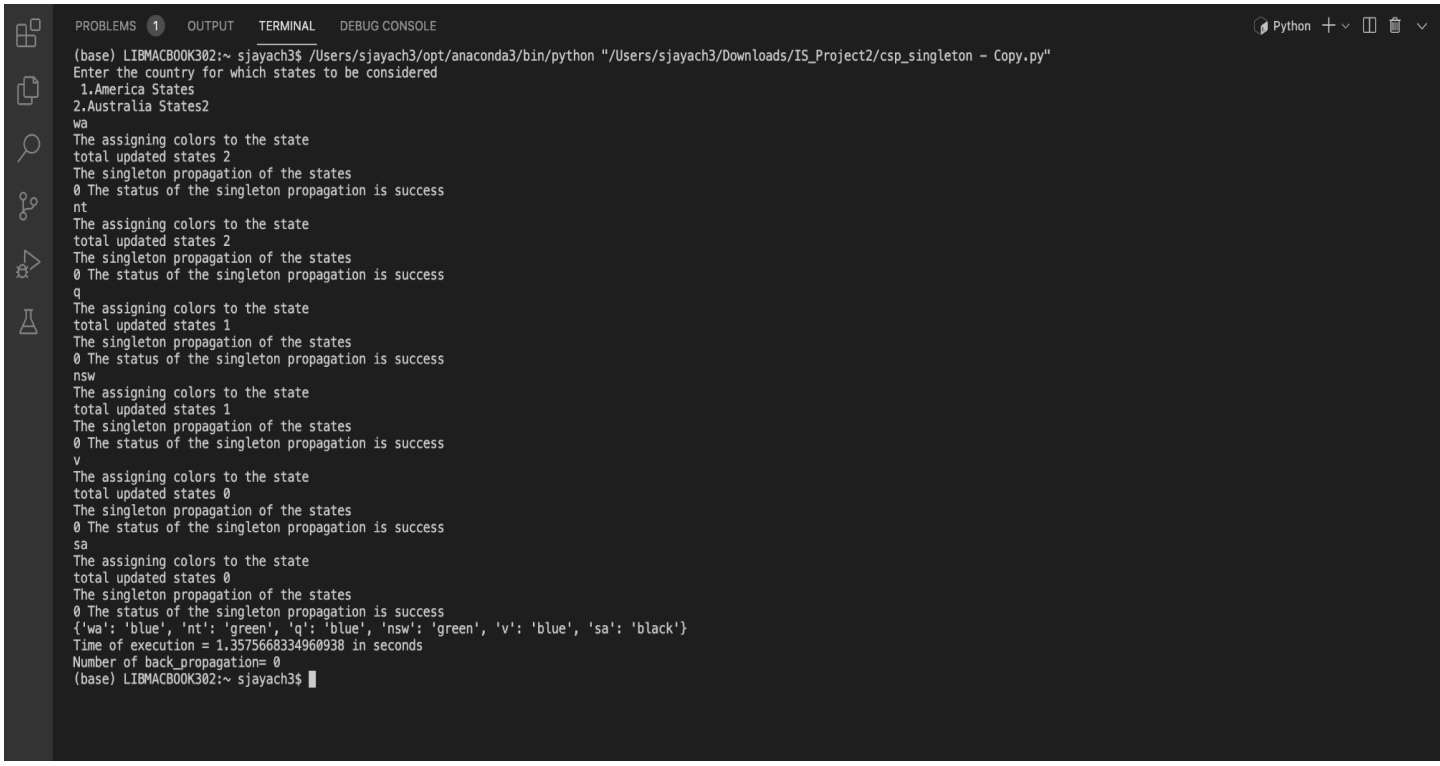
Trials	**Number of Backtracks**	**Time taken in second**
Trial 1	81	1.9914278984069824
Trial 2	81	6.888206081390381
Trial 3	81	2.485658409482340
Trial 4	81	2.772492349832943

Results for CSP_singleton for Australia:

Trial1



Trial 2

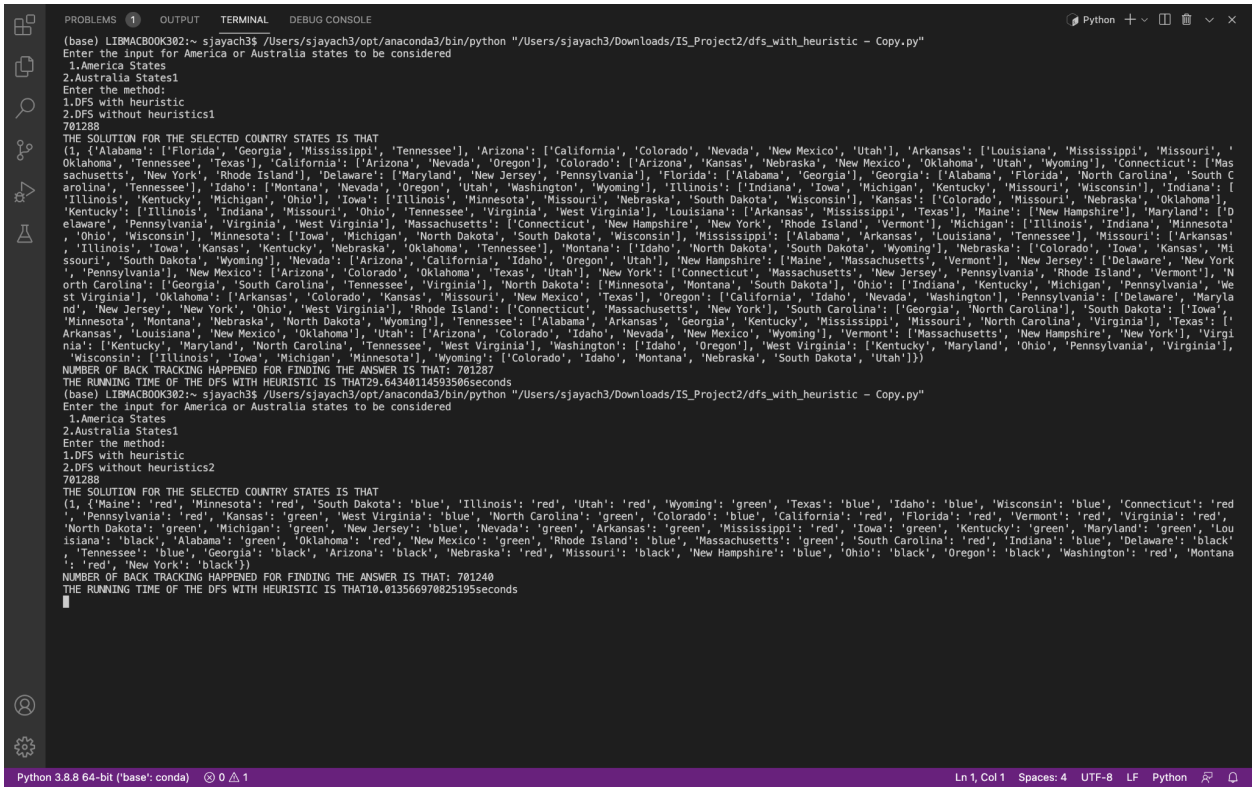


Trials	**Number of Backtracks**	**Time taken in second**
Trial 1	0	1.515894889831543
Trial 2	0	1.3575668334960938
Trial 3	0	7.7993492349249342
Trial 4	0	1.88238482349248234

Results for dfs with and without heuristic America States:

Trial 1

Trial 2


PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Python
(base) LIBMACBOOK302:~ sjayach3$ /Users/sjayach3/opt/anaconda3/bin/python "/Users/sjayach3/Downloads/IS_Project2/dfs_with_heuristic - Copy.py"
Enter the input for America or Australia states to be considered
1.America States
2.Australia States1
Enter the method:
1.DFS with heuristic
2.DFS without heuristics1
701288
THE SOLUTION FOR THE SELECTED COUNTRY STATES IS THAT
(1, {'Alabama': ['Florida', 'Georgia', 'Mississippi', 'Tennessee'], 'Arizona': ['California', 'Colorado', 'Nevada', 'New Mexico', 'Utah'], 'Arkansas': ['Louisiana', 'Mississippi', 'Missouri', '
Oklahoma', 'Tennessee', 'Texas'], 'California': ['Arizona', 'Nevada', 'Oregon'], 'Colorado': ['Arizona', 'Kansas', 'Nebraska', 'New Mexico', 'Oklahoma', 'Utah', 'Wyoming'], 'Connecticut': ['Mas
sachusetts', 'New York', 'Rhode Island'], 'Delaware': ['Maryland', 'New Jersey', 'Pennsylvania'], 'Florida': ['Alabama', 'Georgia'], 'Georgia': ['Alabama', 'Florida', 'North Carolina', 'South C
arolina', 'Tennessee'], 'Idaho': ['Montana', 'Nevada', 'Oregon', 'Utah', 'Washington', 'Wyoming'], 'Illinois': ['Indiana', 'Iowa', 'Michigan', 'Kentucky', 'Missouri', 'Wisconsin'], 'Indiana': [
'Illinois', 'Kentucky', 'Michigan', 'Ohio'], 'Iowa': ['Illinois', 'Minnesota', 'Missouri', 'Nebraska', 'South Dakota', 'Wisconsin'], 'Kansas': ['Colorado', 'Missouri', 'Nebraska', 'Oklahoma'],
'Kentucky': ['Illinois', 'Indiana', 'Missouri', 'Ohio', 'Tennessee', 'Virginia', 'West Virginia'], 'Louisiana': ['Arkansas', 'Mississippi', 'Texas'], 'Maine': ['New Hampshire'], 'Maryland': ['D
elaware', 'Pennsylvania', 'Virginia', 'West Virginia'], 'Massachusetts': ['Connecticut', 'New Hampshire', 'New York', 'Rhode Island', 'Vermont'], 'Michigan': ['Illinois', 'Indiana', 'Minnesota'
, 'Ohio', 'Wisconsin'], 'Minnesota': ['Iowa', 'Michigan', 'North Dakota', 'South Dakota', 'Wisconsin'], 'Mississippi': ['Alabama', 'Arkansas', 'Louisiana', 'Tennessee'], 'Missouri': ['Arkansas'
, 'Illinois', 'Iowa', 'Kansas', 'Kentucky', 'Nebraska', 'Oklahoma', 'Tennessee'], 'Montana': ['Idaho', 'North Dakota', 'South Dakota', 'Wyoming'], 'Nebraska': ['Colorado', 'Iowa', 'Kansas', 'Mi
ssouri', 'South Dakota', 'Wyoming'], 'Nevada': ['Arizona', 'California', 'Idaho', 'Oregon', 'Utah'], 'New Hampshire': ['Maine', 'Massachusetts', 'Vermont'], 'New Jersey': ['Delaware', 'New York
', 'Pennsylvania'], 'New Mexico': ['Arizona', 'Colorado', 'Oklahoma', 'Texas', 'Utah'], 'New York': ['Connecticut', 'Massachusetts', 'New Jersey', 'Pennsylvania', 'Rhode Island', 'Vermont'], 'N
orth Carolina': ['Georgia', 'South Carolina', 'Tennessee', 'Virginia'], 'North Dakota': ['Minnesota', 'Montana', 'South Dakota'], 'Ohio': ['Indiana', 'Kentucky', 'Michigan', 'Pennsylvania', 'We
st Virginia'], 'Oklahoma': ['Arkansas', 'Colorado', 'Kansas', 'Missouri', 'New Mexico', 'Texas'], 'Oregon': ['California', 'Idaho', 'Nevada', 'Washington'], 'Pennsylvania': ['Delaware', 'Maryla
nd', 'New Jersey', 'New York', 'Ohio', 'West Virginia'], 'Rhode Island': ['Connecticut', 'Massachusetts', 'New York'], 'South Carolina': ['Georgia', 'North Carolina'], 'South Dakota': ['Iowa',
'Minnesota', 'Montana', 'Nebraska', 'North Dakota', 'Wyoming'], 'Tennessee': ['Alabama', 'Arkansas', 'Georgia', 'Kentucky', 'Mississippi', 'Missouri', 'North Carolina', 'Virginia'], 'Texas': ['
Arkansas', 'Louisiana', 'New Mexico', 'Oklahoma'], 'Utah': ['Arizona', 'Colorado', 'Idaho', 'Nevada', 'New Mexico', 'Wyoming'], 'Vermont': ['Massachusetts', 'New Hampshire', 'New York'], 'Virgi
nia': ['Kentucky', 'Maryland', 'North Carolina', 'Tennessee', 'West Virginia'], 'Washington': ['Idaho', 'Oregon'], 'West Virginia': ['Kentucky', 'Maryland', 'Ohio', 'Pennsylvania', 'Virginia'],
'Wisconsin': ['Illinois', 'Iowa', 'Michigan', 'Minnesota'], 'Wyoming': ['Colorado', 'Idaho', 'Montana', 'Nebraska', 'South Dakota', 'Utah']})
NUMBER OF BACK TRACKING HAPPENED FOR FINDING THE ANSWER IS THAT: 701287
THE RUNNING TIME OF THE DFS WITH HEURISTIC IS THAT29.64340114593506seconds
(base) LIBMACBOOK302:~ sjayach3$ /Users/sjayach3/opt/anaconda3/bin/python "/Users/sjayach3/Downloads/IS_Project2/dfs_with_heuristic - Copy.py"
Enter the input for America or Australia states to be considered
1.America States
2.Australia States1
Enter the method:
1.DFS with heuristic
2.DFS without heuristics2
701288
THE SOLUTION FOR THE SELECTED COUNTRY STATES IS THAT
(1, {'Maine': 'red', 'Minnesota': 'red', 'South Dakota': 'blue', 'Illinois': 'red', 'Utah': 'red', 'Wyoming': 'green', 'Texas': 'blue', 'Idaho': 'blue', 'Wisconsin': 'blue', 'Connecticut': 'red
', 'Pennsylvania': 'red', 'Kansas': 'green', 'West Virginia': 'blue', 'North Carolina': 'green', 'Colorado': 'blue', 'California': 'red', 'Florida': 'red', 'Vermont': 'red', 'Virginia': 'red',
'North Dakota': 'green', 'Michigan': 'green', 'New Jersey': 'blue', 'Nevada': 'green', 'Arkansas': 'green', 'Mississippi': 'red', 'Iowa': 'green', 'Kentucky': 'green', 'Maryland': 'green', 'Lou
isiana': 'black', 'Alabama': 'green', 'Oklahoma': 'red', 'New Mexico': 'green', 'Rhode Island': 'blue', 'Massachusetts': 'green', 'South Carolina': 'red', 'Indiana': 'blue', 'Delaware': 'black'
, 'Tennessee': 'blue', 'Georgia': 'black', 'Arizona': 'black', 'Nebraska': 'red', 'Missouri': 'black', 'New Hampshire': 'blue', 'Ohio': 'black', 'Oregon': 'black', 'Washington': 'red', 'Montana
': 'red', 'New York': 'black'})
NUMBER OF BACK TRACKING HAPPENED FOR FINDING THE ANSWER IS THAT: 701240
THE RUNNING TIME OF THE DFS WITH HEURISTIC IS THAT10.013566970825195seconds
Python 3.8.8 64-bit ('base': conda)
0 1
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python

WITH HEURISTIC:		
Trial	**Number of backtrack**	**Run Time in seconds**
Trial 1	701287	29.031689167022705
Trial 2	701287	29.64340114593506
Trial 3	701287	29.39450034590345
Trial 4	701287	29.44034853405093

WITHOUT HEURISTIC:		
Trial	**Number of backtrack**	**Run Time in seconds**
Trial 1	701240	9.150140047073364
Trail 2	701240	10.013566978825195
Trial 3	701240	10.345834599384593
Trial 4	701240	9.0495834500039804 8

Results for dfs with and without heuristic Australia States:

Trial 1

Trial 2

WITH HEURISTIC:		
Trial	**Number of backtrack**	**Time in seconds**
Trial 1	5	0.0002708435058593 75
Trial 2	5	0.0003230571746826172
Trial 3	5	0.0002434532456353454
Trial 4	5	0.0002353453452423423

WITHOUT HEURISTIC:		
Trial	**Number of backtrack**	**Time in seconds**
Trial 1	0	0.00013511833343505859 4
Trial 2	0	6.604194641113281234344
Trial 3	0	0.0015757283242444342 43
Trial 4	0	0.0001173044554390394 84

Results for Forward Checking America states:-

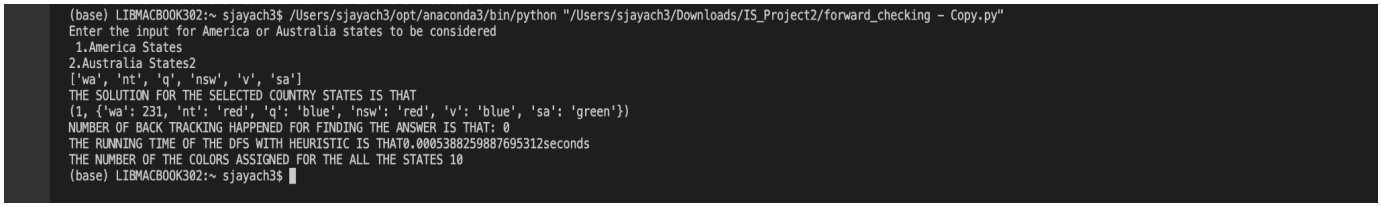
Trial1

Trial 2

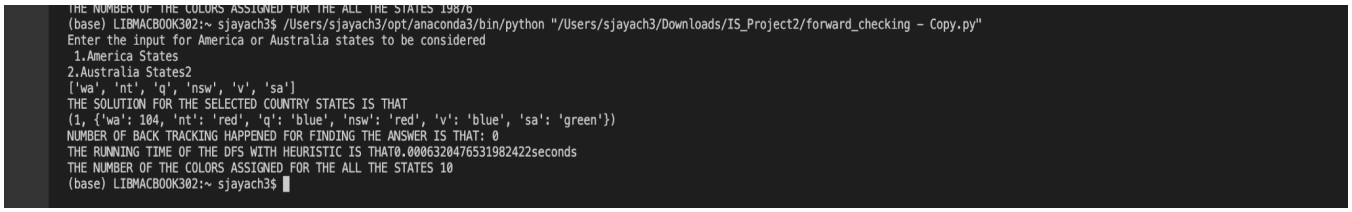
Trial	Number of backtrack	Time taken in seconds	Number of the colors assigned for the states.
Trial 1	19824	1.3070881366729736	19876
Trial 2	19824	1.1848280429840088	19872
Trial 3	19824	1.21434534534545	19872
Trial 4	19824	1.367354433	19872

Results for Forward Checking Australia states:-

Trial 1



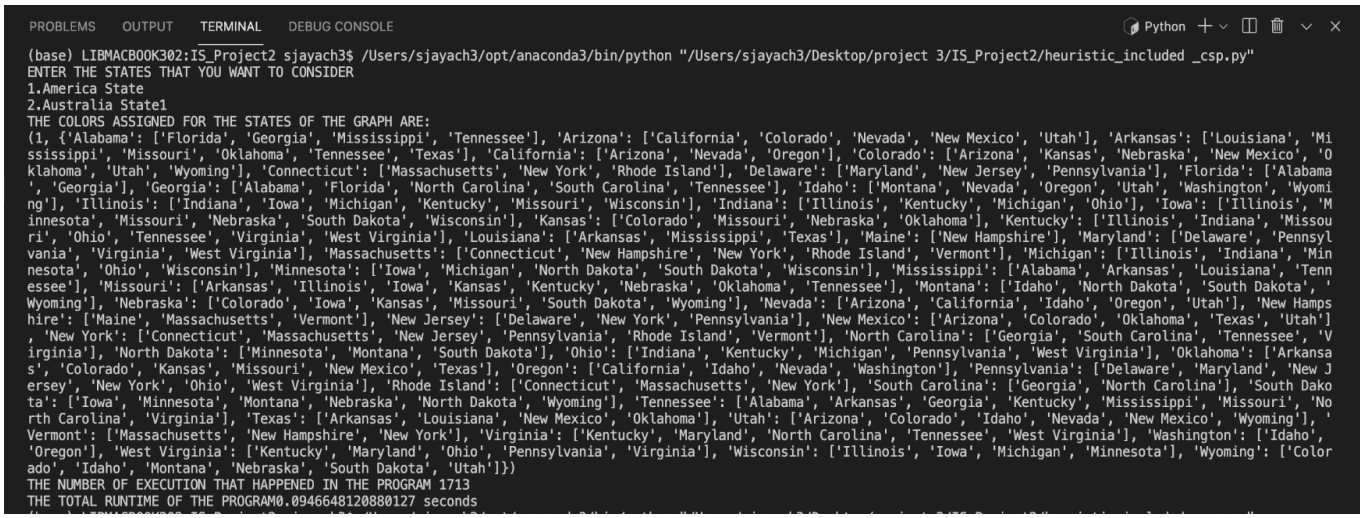
Trial 2



Trial	**Number of backtrack**	**Time taken in seconds**	**Number of the colors assigned for the states.**
Trial 1	0	0.0005388259887695312	10
Trial 2	0	0.0006320476531982422	10
Trial 3	0	0.000442423423434	10
Trial 4	0	0.0003433345345234	10

Results for Heuristic included America states:-

Trial 1




```
(base) LIBMACBOOK302:IS_Project2 sjayach3$ /Users/sjayach3/opt/anaconda3/bin/python "/Users/sjayach3/Desktop/project 3/IS_Project2/heuristic_included _csp.py"
ENTER THE STATES THAT YOU WANT TO CONSIDER
1.America State
2.Australia State1
THE COLORS ASSIGNED FOR THE STATES OF THE GRAPH ARE:
(1, {'Alabama': ['Florida', 'Georgia', 'Mississippi', 'Tennessee'], 'Arizona': ['California', 'Colorado', 'Nevada', 'New Mexico', 'Utah'], 'Arkansas': ['Louisiana', 'Mi
ssissippi', 'Missouri', 'Oklahoma', 'Tennessee', 'Texas'], 'California': ['Arizona', 'Nevada', 'Oregon'], 'Colorado': ['Arizona', 'Kansas', 'Nebraska', 'New Mexico', 'O
klahoma', 'Utah', 'Wyoming'], 'Connecticut': ['Massachusetts', 'New York', 'Rhode Island'], 'Delaware': ['Maryland', 'New Jersey', 'Pennsylvania'], 'Florida': ['Alabama
', 'Georgia'], 'Georgia': ['Alabama', 'Florida', 'North Carolina', 'South Carolina', 'Tennessee'], 'Idaho': ['Montana', 'Nevada', 'Oregon', 'Utah', 'Washington', 'Wyomi
ng'], 'Illinois': ['Indiana', 'Iowa', 'Michigan', 'Kentucky', 'Missouri', 'Wisconsin'], 'Indiana': ['Illinois', 'Kentucky', 'Michigan', 'Ohio'], 'Iowa': ['Illinois', 'M
innesota', 'Missouri', 'Nebraska', 'South Dakota', 'Wisconsin'], 'Kansas': ['Colorado', 'Missouri', 'Nebraska', 'Oklahoma'], 'Kentucky': ['Illinois', 'Indiana', 'Missou
ri', 'Ohio', 'Tennessee', 'Virginia', 'West Virginia'], 'Louisiana': ['Arkansas', 'Mississippi', 'Texas'], 'Maine': ['New Hampshire'], 'Maryland': ['Delaware', 'Pennsyl
vania', 'Virginia', 'West Virginia'], 'Massachusetts': ['Connecticut', 'New Hampshire', 'New York', 'Rhode Island', 'Vermont'], 'Michigan': ['Illinois', 'Indiana', 'Min
nesota', 'Ohio', 'Wisconsin'], 'Minnesota': ['Iowa', 'Michigan', 'North Dakota', 'South Dakota', 'Wisconsin'], 'Mississippi': ['Alabama', 'Arkansas', 'Louisiana', 'Tenn
essee'], 'Missouri': ['Arkansas', 'Illinois', 'Iowa', 'Kansas', 'Kentucky', 'Nebraska', 'Oklahoma', 'Tennessee'], 'Montana': ['Idaho', 'North Dakota', 'South Dakota', '
Wyoming'], 'Nebraska': ['Colorado', 'Iowa', 'Kansas', 'Missouri', 'South Dakota', 'Wyoming'], 'Nevada': ['Arizona', 'California', 'Idaho', 'Oregon', 'Utah'], 'New Hamps
hire': ['Maine', 'Massachusetts', 'Vermont'], 'New Jersey': ['Delaware', 'New York', 'Pennsylvania'], 'New Mexico': ['Arizona', 'Colorado', 'Oklahoma', 'Texas', 'Utah']
, 'New York': ['Connecticut', 'Massachusetts', 'New Jersey', 'Pennsylvania', 'Rhode Island', 'Vermont'], 'North Carolina': ['Georgia', 'South Carolina', 'Tennessee', 'V
irginia'], 'North Dakota': ['Minnesota', 'Montana', 'South Dakota'], 'Ohio': ['Indiana', 'Kentucky', 'Michigan', 'Pennsylvania', 'West Virginia'], 'Oklahoma': ['Arkansa
s', 'Colorado', 'Kansas', 'Missouri', 'New Mexico', 'Texas'], 'Oregon': ['California', 'Idaho', 'Nevada', 'Washington'], 'Pennsylvania': ['Delaware', 'Maryland', 'New J
ersey', 'New York', 'Ohio', 'West Virginia'], 'Rhode Island': ['Connecticut', 'Massachusetts', 'New York'], 'South Carolina': ['Georgia', 'North Carolina'], 'South Dako
ta': ['Iowa', 'Minnesota', 'Montana', 'Nebraska', 'North Dakota', 'Wyoming'], 'Tennessee': ['Alabama', 'Arkansas', 'Georgia', 'Kentucky', 'Mississippi', 'Missouri', 'No
rth Carolina', 'Virginia'], 'Texas': ['Arkansas', 'Louisiana', 'New Mexico', 'Oklahoma'], 'Utah': ['Arizona', 'Colorado', 'Idaho', 'Nevada', 'New Mexico', 'Wyoming'], '
Vermont': ['Massachusetts', 'New Hampshire', 'New York'], 'Virginia': ['Kentucky', 'Maryland', 'North Carolina', 'Tennessee', 'West Virginia'], 'Washington': ['Idaho',
'Oregon'], 'West Virginia': ['Kentucky', 'Maryland', 'Ohio', 'Pennsylvania', 'Virginia'], 'Wisconsin': ['Illinois', 'Iowa', 'Michigan', 'Minnesota'], 'Wyoming': ['Color
ado', 'Idaho', 'Montana', 'Nebraska', 'South Dakota', 'Utah']})
THE NUMBER OF EXECUTION THAT HAPPENED IN THE PROGRAM 1713
THE TOTAL RUNTIME OF THE PROGRAM0.0946648120880127 seconds
```

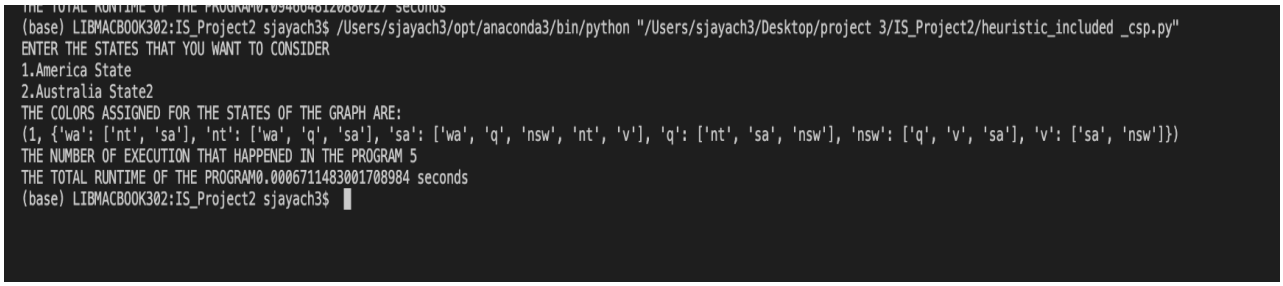
Trial 2

```
(base) LIBMACBOOK302:IS_Project2 sjayach3$ /Users/sjayach3/opt/anaconda3/bin/python "/Users/sjayach3/Desktop/project 3/IS_Project2/heuristic_included _csp.py"
ENTER THE STATES THAT YOU WANT TO CONSIDER
1.America State
2.Australia State1
THE COLORS ASSIGNED FOR THE STATES OF THE GRAPH ARE:
(1, {'Alabama': ['Florida', 'Georgia', 'Mississippi', 'Tennessee'], 'Arizona': ['California', 'Colorado', 'Nevada', 'New Mexico', 'Utah'], 'Arkansas': ['Louisiana', 'Mississippi', 'Missouri', '
Oklahoma', 'Tennessee', 'Texas'], 'California': ['Arizona', 'Nevada', 'Oregon'], 'Colorado': ['Arizona', 'Kansas', 'Nebraska', 'New Mexico', 'Oklahoma', 'Utah', 'Wyoming'], 'Connecticut': ['Mas
sachusetts', 'New York', 'Rhode Island'], 'Delaware': ['Maryland', 'New Jersey', 'Pennsylvania'], 'Florida': ['Alabama', 'Georgia'], 'Georgia': ['Alabama', 'Florida', 'North Carolina', 'South C
arolina', 'Tennessee'], 'Idaho': ['Montana', 'Nevada', 'Oregon', 'Utah', 'Washington', 'Wyoming'], 'Illinois': ['Indiana', 'Iowa', 'Michigan', 'Kentucky', 'Missouri', 'Wisconsin'], 'Indiana': [
'Illinois', 'Kentucky', 'Michigan', 'Ohio'], 'Iowa': ['Illinois', 'Minnesota', 'Missouri', 'Nebraska', 'South Dakota', 'Wisconsin'], 'Kansas': ['Colorado', 'Missouri', 'Nebraska', 'Oklahoma'],
'Kentucky': ['Illinois', 'Indiana', 'Missouri', 'Ohio', 'Tennessee', 'Virginia', 'West Virginia'], 'Louisiana': ['Arkansas', 'Mississippi', 'Texas'], 'Maine': ['New Hampshire'], 'Maryland': ['D
elaware', 'Pennsylvania', 'Virginia', 'West Virginia'], 'Massachusetts': ['Connecticut', 'New Hampshire', 'New York', 'Rhode Island', 'Vermont'], 'Michigan': ['Illinois', 'Indiana', 'Minnesota'
, 'Ohio', 'Wisconsin'], 'Minnesota': ['Iowa', 'Michigan', 'North Dakota', 'South Dakota', 'Wisconsin'], 'Mississippi': ['Alabama', 'Arkansas', 'Louisiana', 'Tennessee'], 'Missouri': ['Arkansas'
, 'Illinois', 'Iowa', 'Kansas', 'Kentucky', 'Nebraska', 'Oklahoma', 'Tennessee'], 'Montana': ['Idaho', 'North Dakota', 'South Dakota', 'Wyoming'], 'Nebraska': ['Colorado', 'Iowa', 'Kansas', 'Mi
ssouri', 'South Dakota', 'Wyoming'], 'Nevada': ['Arizona', 'California', 'Idaho', 'Oregon', 'Utah'], 'New Hampshire': ['Maine', 'Massachusetts', 'Vermont'], 'New Jersey': ['Delaware', 'New York
', 'Pennsylvania'], 'New Mexico': ['Arizona', 'Colorado', 'Oklahoma', 'Texas', 'Utah'], 'New York': ['Connecticut', 'Massachusetts', 'New Jersey', 'Pennsylvania', 'Rhode Island', 'Vermont'], 'N
orth Carolina': ['Georgia', 'South Carolina', 'Tennessee', 'Virginia'], 'North Dakota': ['Minnesota', 'Montana', 'South Dakota'], 'Ohio': ['Indiana', 'Kentucky', 'Michigan', 'Pennsylvania', 'We
st Virginia'], 'Oklahoma': ['Arkansas', 'Colorado', 'Kansas', 'Missouri', 'New Mexico', 'Texas'], 'Oregon': ['California', 'Idaho', 'Nevada', 'Washington'], 'Pennsylvania': ['Delaware', 'Maryla
nd', 'New Jersey', 'New York', 'Ohio', 'West Virginia'], 'Rhode Island': ['Connecticut', 'Massachusetts', 'New York'], 'South Carolina': ['Georgia', 'North Carolina'], 'South Dakota': ['Iowa',
'Minnesota', 'Montana', 'Nebraska', 'North Dakota', 'Wyoming'], 'Tennessee': ['Alabama', 'Arkansas', 'Georgia', 'Kentucky', 'Mississippi', 'Missouri', 'North Carolina', 'Virginia'], 'Texas': ['
Arkansas', 'Louisiana', 'New Mexico', 'Oklahoma'], 'Utah': ['Arizona', 'Colorado', 'Idaho', 'Nevada', 'New Mexico', 'Wyoming'], 'Vermont': ['Massachusetts', 'New Hampshire', 'New York'], 'Virgi
nia': ['Kentucky', 'Maryland', 'North Carolina', 'Tennessee', 'West Virginia'], 'Washington': ['Idaho', 'Oregon'], 'West Virginia': ['Kentucky', 'Maryland', 'Ohio', 'Pennsylvania', 'Virginia'],
 'Wisconsin': ['Illinois', 'Iowa', 'Michigan', 'Minnesota'], 'Wyoming': ['Colorado', 'Idaho', 'Montana', 'Nebraska', 'South Dakota', 'Utah']})
THE NUMBER OF EXECUTION THAT HAPPENED IN THE PROGRAM 1713
THE TOTAL RUNTIME OF THE PROGRAM0.09508705139160156 seconds
```

Trial	**Number of executions that happened in the program**	**Time taken in seconds**
Trial 1	1713	0.0946648120880127
Trial 2	1713	0.9508705139160156
Trial 3	1713	0.83485385735845435
Trial 4	1713	0.03457348583458345

Results for Heuristic included Australia states:-

Trial 1



Trial2

Trial	Number of executions that happened in the program	Time taken in seconds
Trial 1	5	0.0006711483001708984
Trial 2	5	0.0004191398620605469
Trial 3	5	0.0005234325345353454
Trial 4	5	0.0003453405353045034

CONCLUSION:

This project was a good opportunity for us to learn about many combinatorial problems in operational research, such as scheduling and timetabling, We were able to understand that these problems can be formulated as CSPs. We learnt about the significance of constraint programming and its application. Solving this problem was a very good learning experience overall.

REFERENCES:

- Geeks for Geeks
- Youtube
- Lecture slides
- StackOverflow
- https://study.com/academy/lesson/constraint-satisfaction-problems-definition-examples.html
- https://www.cs.cmu.edu/~arielpro/mfai_papers/lecture4.pdf