UCS1504 - Artificial Intelligence Lab **Name:** ***Krithika Swaminathan***
AY: ***2022-23*** **Roll No.:** ***205001057***

Exercise 1 – Uninformed Search Strategies (BFS, DFS)

Date: 18/08/2022

Aim:

Given a box with a combination of two colour balls, to write a python program that implements the following functions:

1. Generate a sequence of n balls as states.
2. Keep track and represent these states using an appropriate data structure.
3. Print the sequence of states using BFS.
4. Print the sequence of states using DFS.

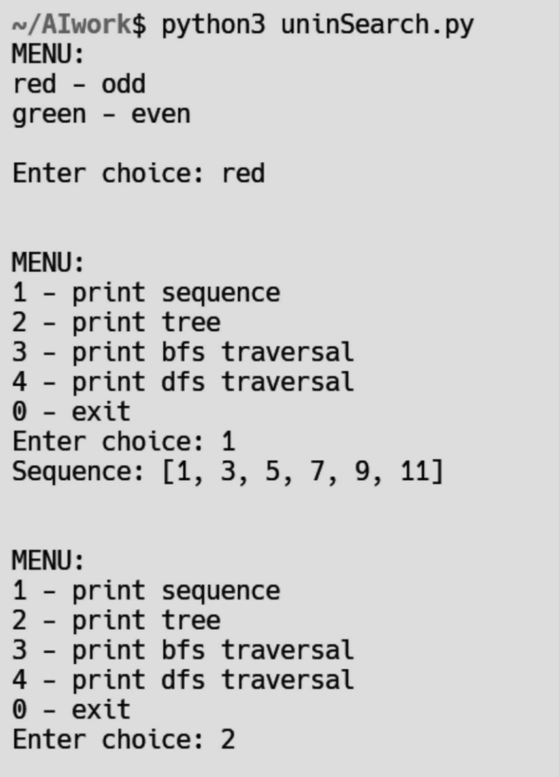
Code:

```
'''You are given a box with a combination of two-color balls (RED, GREEN). Assume that
you are counting the balls in the box as a sequence of ODD and EVEN numbers for RED
and GREEN respectively.'''

#class and function definitions

from collections import deque as queue

class Node:

    def __init__(self, data):
        self.left = None
        self.right = None
        self.data = data

    def insert(self, data):
        if self.left is None:
            self.left = Node(data)
        elif self.right is None:
            self.right = Node(data)
        elif self.left.right is None:
            self.left.insert(data)
        else:
            self.right.insert(data)
```

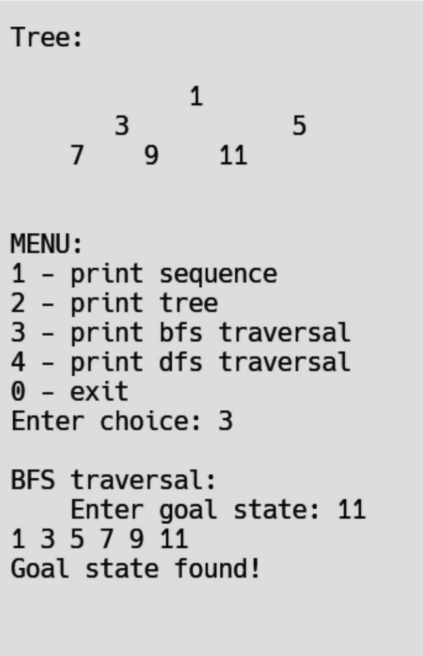
```
def printTree(self):
    print(self.data, end=" ")
    if self.left:
        self.left.printTree()
    if self.right:
        self.right.printTree()

def bfsTree(self, root = None, goal = None):
    if root is None:
        return

    queue = [root]

    while len(queue) > 0:
        cur_node = queue.pop(0)

        if cur_node.data == goal:
            print(cur_node.data)
            print("Goal state found!")
            break

        if cur_node.left is not None:
            queue.append(cur_node.left)

        if cur_node.right is not None:
            queue.append(cur_node.right)

        print(cur_node.data, end=" ")

def dfsTree(self, root = None, goal = None, flag = None):
    if root is None:
        root = self

    if root.data == goal:
        print(root.data)
        flag = 1
        print("Goal state found!")
        return

    if flag == 0:
        print(root.data, end=" ")
    if root.left:
        root.left.dfsTree(goal = goal, flag = flag)
        if root.right:
            root.right.dfsTree(goal = goal, flag = flag)


def levelOrder(root):
    print()
    if (root is None):
        return
    q = queue()

    q.append(root)
    q.append(None)
    ht = 3
    h = 0

    while (len(q) > 1):
        curr = q.popleft()
        if (curr is None):
            q.append(None)
            h += 1
            print()

        else:
            if (curr.left):
                q.append(curr.left)
            if (curr.right):
                q.append(curr.right)

            print(" "*((2**(ht-h))+(ht-h)),curr.data, end="")
            if (h%2 == 1):
                print(end = " "*(4*h))


def createEven(N):
    evens = []
    for i in range(2, (2*N)+1, 2):
        evens.append(i)
    return evens


def createOdd(N):
    odds = []
    for i in range(0, N):
        odds.append((2 * i) + 1)
    return odds


def constrTree(nodes):
    root = Node(nodes[0])
    for i in range(1, len(nodes)):
        root.insert(nodes[i])
    return root


#main program

print("MENU: \nred - odd\ngreen - even")
choice = input("\nEnter choice: ")

seq = None
if (choice.lower() == "red"):
    seq = createOdd(6)
    tree = constrTree(seq)
elif (choice.lower() == "green"):
    seq = createEven(6)
    tree = constrTree(seq)
else:
    print("Invalid choice!")
    exit()
print()

ch = 1
while (ch > 0):
    print("\nMENU: \n1 - print sequence\n2 - print tree\n3 - print bfs traversal\n4 - print dfs
traversal\n0 - exit")
    ch = int(input("Enter choice: "))
    if (ch == 1):
        print("Sequence:", seq)
    elif (ch == 2):
        print("\nTree: ")
        levelOrder(tree)
        print()
    elif (ch == 3):
        print("\nBFS traversal: ")
        gl = int(input("\tEnter goal state: "))
        tree.bfsTree(root = tree, goal = gl)
        print()
    elif (ch == 4):
        print("\nDFS traversal (pre-order): ")
        gl = int(input("\tEnter goal state: "))
        tree.dfsTree(root = tree, goal = gl, flag = 0)
        print()
    elif (ch == 0):
        break
    else:
        print("Invalid choice! Try again.")
    print()
```

Output:

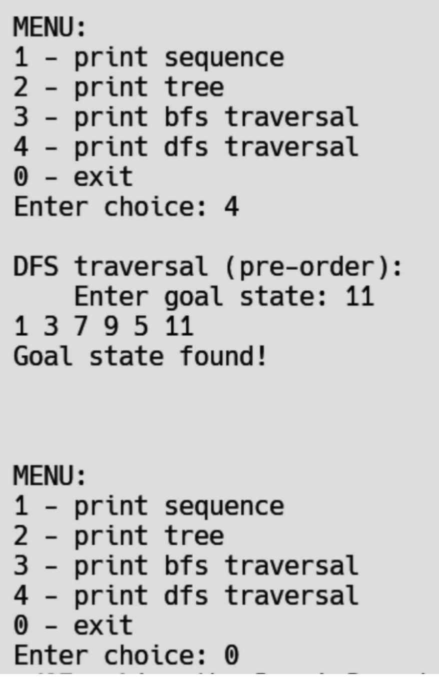
Test case 1:



```
~/AIwork$ python3 uninSearch.py
MENU:
red - odd
green - even

Enter choice: red


MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 1
Sequence: [1, 3, 5, 7, 9, 11]


MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 2
```

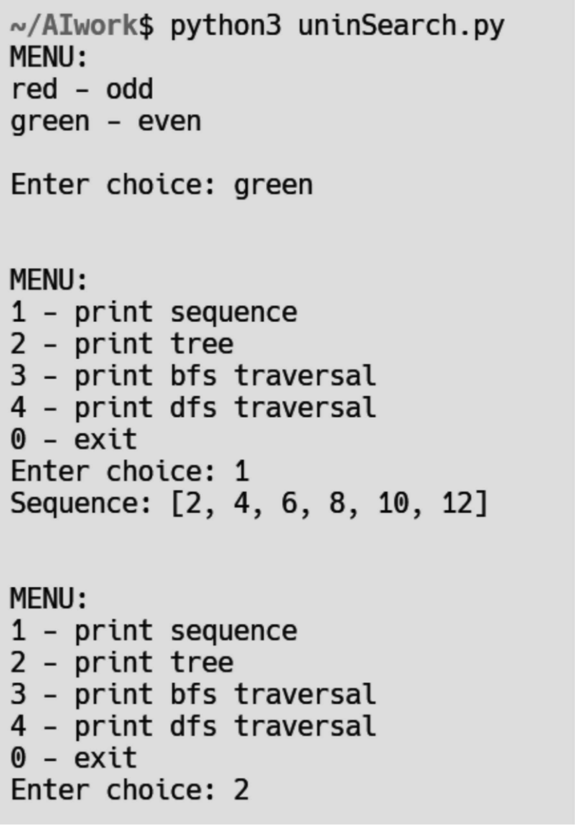


```
Tree:

            1
       3           5
    7    9    11


MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 3

BFS traversal:
    Enter goal state: 11
1 3 5 7 9 11
Goal state found!
```

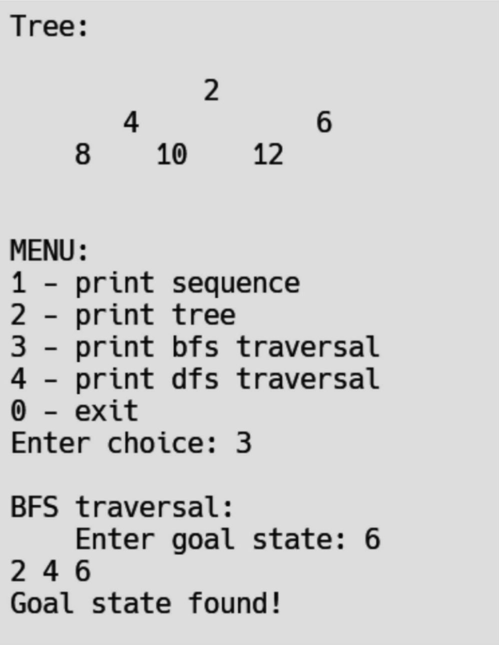


```
MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 4

DFS traversal (pre-order):
    Enter goal state: 11
1 3 7 9 5 11
Goal state found!



MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 0
```

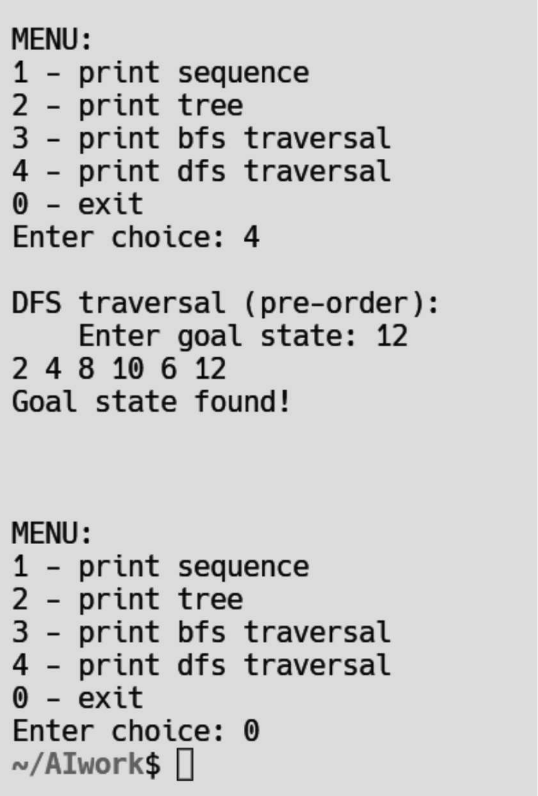
Test case 2:



```
~/AIwork$ python3 uninSearch.py
MENU:
red - odd
green - even

Enter choice: green


MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 1
Sequence: [2, 4, 6, 8, 10, 12]


MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 2
```



```
Tree:

            2
       4           6
    8    10    12


MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 3

BFS traversal:
    Enter goal state: 6
2 4 6
Goal state found!
```



```
MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 4

DFS traversal (pre-order):
    Enter goal state: 12
2 4 8 10 6 12
Goal state found!



MENU:
1 - print sequence
2 - print tree
3 - print bfs traversal
4 - print dfs traversal
0 - exit
Enter choice: 0
~/AIwork$
```