
Exercise 2 – Uninformed Search Strategies (DLS, IDS)

Date: 25/08/2022

Aim:

To write a python program to implement the following:

1. Generate a random sequence for n distinct numbers by specifying the initial value and end value. Print the list.
2. Use a suitable data structure to keep track of the parent of every state and show the structure as a state space representation for the list created.
3. Print the sequence of states and actions from the initial state to the goal state using DLS with suitable data structure.
4. Print the sequence of states and actions from the initial state to the goal state for every level using IDS with suitable data structure.

Code:

```
'''Implementation of DLS and IDS'''

#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 dlsTree(self, root=None, goal=None, limit=None):
        if root is None:
            root = self

        print(root.data, end=" ")

        if root.data == goal:
            #print(root.data)
            print("\nGoal state found!")
            return True

        if limit <= 0:
            return False

        if root.left:
            root.left.dlsTree(goal=goal, limit=limit - 1)
        if root.right:
            root.right.dlsTree(goal=goal, limit=limit - 1)

    def idsTree(self, root=None, goal=None, limit=None):
        if root is None:
            root = self

        for i in range(limit + 1):
            print("\nlimit", i, "\b:", end=" ")
            if (root.dlsTree(root=root, goal=goal, limit=i)):
                return True

        return False


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 createSeq(start, end, n):
    seq = []
    for i in range(start, end, (end - start) // n):
        seq.append(i)
    return seq


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


#main program

init = int(input("Enter initial value: "))
fin = int(input("Enter final value: "))
```

```
n = int(input("Enter no. of numbers: "))
seq = createSeq(init, fin, n)
print("\nSequence: ", seq)
tree = constrTree(seq)
print()
levelOrder(tree)
print("\n")

gl = int(input("Enter goal state: "))
lt = int(input("Enter limit value: "))

print("\nDLS: ")
tree.dlsTree(root=tree, goal=gl, limit=lt)

print("\nIDS: ")
tree.idsTree(root=tree, goal=gl, limit=lt)
```

Output:

```
~/AIwork$ python3 dlsids.py
Enter initial value: 10
Enter final value: 20
Enter no. of numbers: 5

Sequence:  [10, 12, 14, 16, 18]


            10
       12           14
    16    18

Enter goal state: 14
Enter limit value: 2

DLS:
10 12 16 18 14
Goal state found!

IDS:

limit 0: 10
limit 1: 10 12 14
Goal state found!

limit 2: 10 12 16 18 14
Goal state found!
```