AI Assignment - 2
Depth Limited Search and Iterative Depth Search

Sabarivasan Velayutham
205001085
CSE-B

Function Description:
Depth Limited Search is similar to dfs but it is limited to search only till the specified limit/level to prevent infinite loops. Iterative Depth Search is DLS run with different limits until the goal is reached

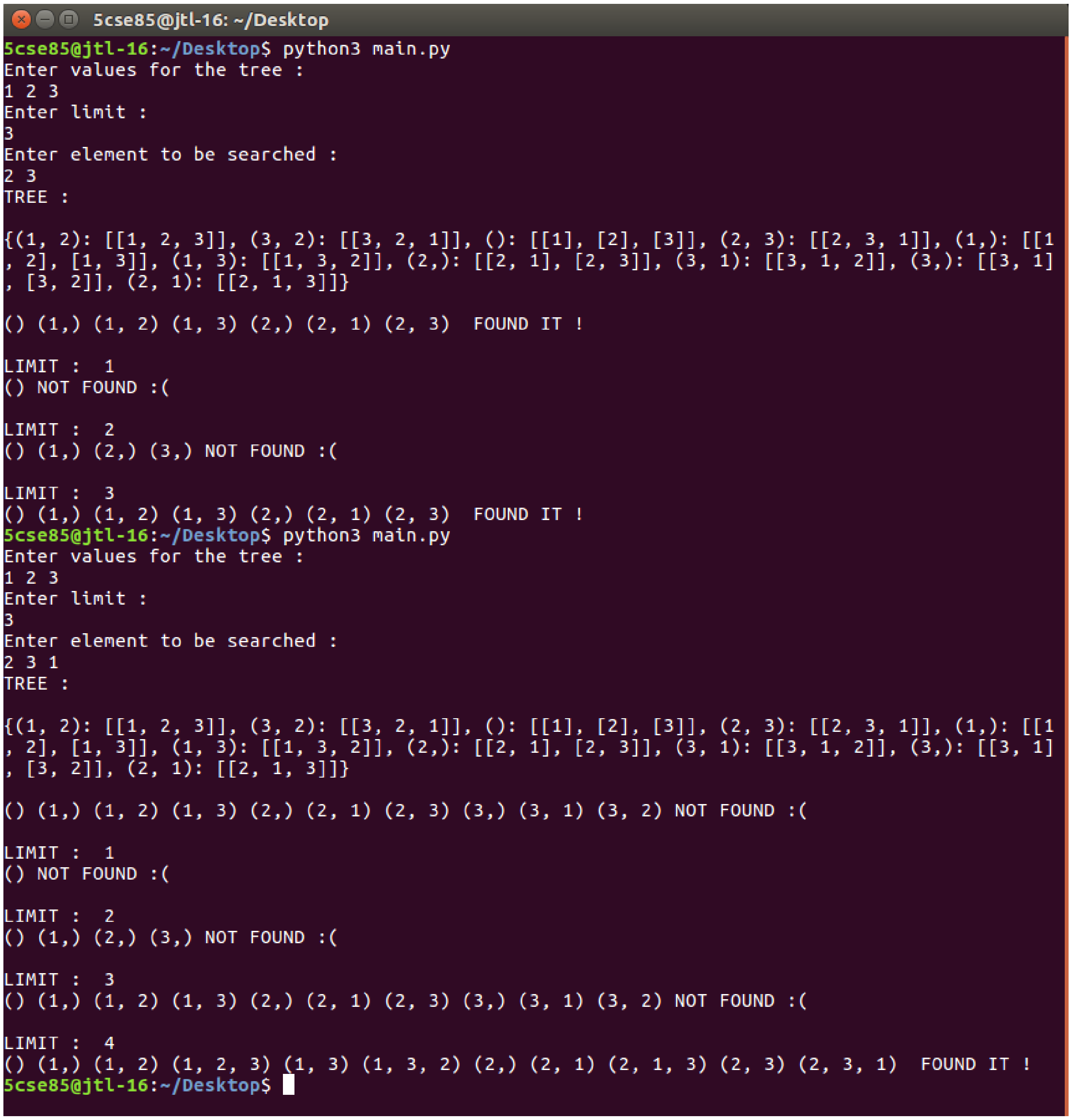
Data Structure : Dictionary,Array/List

Code :

```
from collections import defaultdict
from itertools import permutations


def dls(graph, visited, root, limit, search):

        if limit == 0:
        return 0
        if root == search:
        print(root, " FOUND IT !")
        return 1

        else:
        if root not in visited:
        print(root, end=' ')
        visited.add(root)
        if root in graph.keys():
                for neighbour in graph[root]:
                ans = dls(graph, visited, neighbour, limit-1, search)
                if ans == 1:
                return 1
        return 0
        return 0


def ids(graph, search):
        limit = len(values) + 1
```

```
    for i in range(1, limit+1):
    print("LIMIT : ", i)
    if dls(graph, set(), (), i, search) == 1:
    break
    else:
    print("NOT FOUND :(")
    print()


graph = dict()


def permute(values, temp_array=[], root=tuple([])):
    if len(values) == 0:
    return
    for i in values:
    temp_array.append(i)
    if root in graph.keys():
    graph[root].append(temp_array.copy())
    else:
    graph[root] = [temp_array.copy()]

    permute([x for x in values if x != i],
        temp_array.copy(), tuple(temp_array.copy()))
    temp_array.pop()

# graph = defaultdict(list)
# graph = {0: [1, 2], 1: [3,4], 2: [5,6] , 3: [7,8], 4: [9,10] }
# visited = set()
# ans = dls(graph, visited, 0, 4, 10)


print("Enter values for the tree : ")
values = list(map(int, input().split()))
print("Enter limit : ")
limit = int(input())
print("Enter element to be searched : ")
search = tuple(map(int, input().split()))


permute(values)

print("TREE : ")
print()
print(graph)
print()

for i in graph.keys():
        for j in range(len(graph[i])):
        graph[i][j] = tuple(graph[i][j])


ans = dls(graph, set(), (), limit, search)
if ans != 1:
        print("NOT FOUND :(")

print()

ids(graph, search)
```

Output :



```
5cse85@jtl-16: ~/Desktop
5cse85@jtl-16:~/Desktop$ python3 main.py
Enter values for the tree :
1 2 3
Enter limit :
3
Enter element to be searched :
2 3
TREE :

{(1, 2): [[1, 2, 3]], (3, 2): [[3, 2, 1]], (): [[1], [2], [3]], (2, 3): [[2, 3, 1]], (1,): [[1
, 2], [1, 3]], (1, 3): [[1, 3, 2]], (2,): [[2, 1], [2, 3]], (3, 1): [[3, 1, 2]], (3,): [[3, 1]
, [3, 2]], (2, 1): [[2, 1, 3]]}

() (1,) (1, 2) (1, 3) (2,) (2, 1) (2, 3)  FOUND IT !

LIMIT :  1
() NOT FOUND :(

LIMIT :  2
() (1,) (2,) (3,) NOT FOUND :(

LIMIT :  3
() (1,) (1, 2) (1, 3) (2,) (2, 1) (2, 3)  FOUND IT !
5cse85@jtl-16:~/Desktop$ python3 main.py
Enter values for the tree :
1 2 3
Enter limit :
3
Enter element to be searched :
2 3 1
TREE :

{(1, 2): [[1, 2, 3]], (3, 2): [[3, 2, 1]], (): [[1], [2], [3]], (2, 3): [[2, 3, 1]], (1,): [[1
, 2], [1, 3]], (1, 3): [[1, 3, 2]], (2,): [[2, 1], [2, 3]], (3, 1): [[3, 1, 2]], (3,): [[3, 1]
, [3, 2]], (2, 1): [[2, 1, 3]]}

() (1,) (1, 2) (1, 3) (2,) (2, 1) (2, 3) (3,) (3, 1) (3, 2) NOT FOUND :(

LIMIT :  1
() NOT FOUND :(

LIMIT :  2
() (1,) (2,) (3,) NOT FOUND :(

LIMIT :  3
() (1,) (1, 2) (1, 3) (2,) (2, 1) (2, 3) (3,) (3, 1) (3, 2) NOT FOUND :(

LIMIT :  4
() (1,) (1, 2) (1, 2, 3) (1, 3) (1, 3, 2) (2,) (2, 1) (2, 1, 3) (2, 3) (2, 3, 1)  FOUND IT !
5cse85@jtl-16:~/Desktop$
```