Experiment 02

DFS:

```
visited = set()


def dfs(visited, graph, node):
  if node not in visited:
        print(node, end = " ")
        visited.add(node)
        for neighbour in graph[node]:
        dfs(visited, graph, neighbour)


#
#         5
#         / \
#         3   7
#   / \   \
#  2   4   8
#
graph = {
  '5': ['3', '7'],
  '3': ['2', '4'],
  '7': ['8'],
  '2': [],
  '4': [],
  '8': []
}


print("Following is the Depth-First Search")
dfs(visited, graph, '5')
```

Output:
Following is the Depth-First Search
5 3 2 4 7 8
Process finished with exit code 0

```
DLS:
#
#          A
#          / \
#   B    C
#  / \    \
# D   E   F

def dls(graph, node, goal, depth_limit):
  return dls_recursive(graph, node, goal, depth_limit, 0, set())


def dls_recursive(graph, node, goal, depth_limit, current_depth, visited):
  print(node, end='')


  if node == goal:
        return True


  if current_depth == depth_limit:
        return False


  visited.add(node)


  for neighbor in graph[node]:
        if neighbor not in visited:
        if dls_recursive(graph, neighbor, goal, depth_limit, current_depth + 1, visited):
        return True


  return False


graph = {
  'A': ['B', 'C'],
  'B': ['D', 'E'],
  'C': ['F'],
  'D': [' '],
  'E': [' '],
  'F': [' ']
```

```
}


start_node = input("Enter the start node : ")
goal_node = input("Enter the goal node : ")
maxDepth = int(input("Enter the max depth : "))
print("DFS path", end=' -> ')


result = dls(graph, start_node, goal_node, maxDepth)
print()
if result:
  print(f"Goal '{goal_node}' found within depth limit.")
else:
  print(f"Goal '{goal_node}' not found within depth limit.")
Output:
Enter the start node : A
Enter the goal node : F
Enter the max depth : 1
DFS path -> ABC
Goal 'F' not found within depth limit.

Process finished with exit code 0



Enter the start node : A
Enter the goal node : F
Enter the max depth : 2
DFS path -> ABDECF
Goal 'F' found within depth limit.

Process finished with exit code 0




IDFS:
dfs_path = ''
def dls(graph, node, goal, depth_limit):
  return dls_recursive(graph, node, goal, depth_limit, 0, set())


def dls_recursive(graph, node, goal, depth_limit, current_depth, visited):
```

```
    global dfs_path
    dfs_path += node


    if node == goal:
            return True


    if current_depth == depth_limit:
            return False


    visited.add(node)


    for neighbor in graph[node]:
            if neighbor not in visited:
            if dls_recursive(graph, neighbor, goal, depth_limit, current_depth + 1, visited):
            return True


    return False


def idfs(graph, start, goal):
    max_depth_limit = len(graph)
    depth_limit = 0


    while depth_limit <= max_depth_limit:
            print(f"Trying depth limit: {depth_limit}")
            result = dls(graph, start, goal, depth_limit)
            if result:
            print(f"Goal '{goal}' found at depth {depth_limit}")
            return True
            global dfs_path
            dfs_path += ' '
            depth_limit += 1


    print("Maximum depth limit reached. Goal not found.")
    return False
#
#       A
```

```
#          / \
#   B    C
#  / \   \
# D   E   F


graph = {
   'A': ['B', 'C'],
   'B': ['D', 'E'],
   'C': ['F'],
   'D': [],
   'E': [],
   'F': []
}


start_node = input("Enter the start node : ")
goal_node = input("Enter the goal node : ")


result = idfs(graph, start_node, goal_node)


if result:
   print('Path: ', dfs_path)
   print(f"Goal '{goal_node}' found using IDFS.")
else:
   print('Path: ', dfs_path)
   print(f"Goal '{goal_node}' not found using IDFS.")
```

Output:

```
Enter the start node : A
Enter the goal node : F
Trying depth limit: 0
Trying depth limit: 1
Trying depth limit: 2
Goal 'F' found at depth 2
Path:  A ABC ABDECF
Goal 'F' found using IDFS.
```

```
Process finished with exit code 0
```