|  |  |  |
| --- | --- | --- |
|  | **LAB: Artificial Intelligence**  **TASK** | |
|  |  | |
| **Lab Instructor** |  | Ms. Saba Aslam |
| **Department** | | Computer Science |

**Lab No 4 : Uninformed Searches**

|  |
| --- |
|  |
| 1. Make a word document paste your code and output there.  Shape     Description automatically generated with medium confidence 2. Comments in the code explaining chunks of the code is important. 3. Plagiarism is strictly prohibited, 0 mark would be given to students who cheat. |

**Objectives:**

* In this Lab we will study and implement basic algorithms of Uninformed searches
* We will familiarize ourselves breath first search, depth first search, depth limited search, iterative deepening search

**Lab Tasks:**

A businessman flight landed in Dunc ich, England. He has a meeting in some city.

Write a program and ask him at which city he want to reach and show him path through following searches

1. **Depth first search**

**CODE:**

class dfs:  
 graph = {  
 'Dunwich': ['Blaxhall', 'Harwich'],  
 'Blaxhall': ['Dunwich', 'Harwich', 'Feering'],  
 'Harwich': ['Dunwich', 'Blaxhall', 'Tip tree', 'Clacton'],  
 'Feering': ['Blaxhall', 'Tip tree', 'Maldon'],  
 'Tip tree': ['Harwich', 'Feering', 'Clacton', 'Maldon'],  
 'Clacton': ['Harwich', 'Tip tree', 'Maldon'],  
 'Maldon': ['Feering', 'Clacton', 'Tip tree'],  
 }  
  
  
  
 def find\_path(self, graph, start, end, path=None):  
 if path is None:  
 path = []  
 path = path + [start]  
 if start == end:  
 return path  
  
 if start not in graph:  
 return None  
 for node in graph[start]:  
 if node not in path:  
 newpath = self.find\_path(graph, node, end, path)  
 if newpath:  
 return newpath  
 return None  
  
  
g = dfs()  
select = str(input('Enter the name of city witch you want to reach : '))  
print('\nPath : ',end=' ')  
print(str(g.find\_path(g.graph, 'Dunwich', select)))

**OUTPUT:**

Text

Description automatically generated

1. **Depth limited search**

**CODE:**

class dfs:  
 graph = {  
 'Dunwich': ['Blaxhall', 'Harwich'],  
 'Blaxhall': ['Dunwich', 'Harwich', 'Feering'],  
 'Harwich': ['Dunwich', 'Blaxhall', 'Tip tree', 'Clacton'],  
 'Feering': ['Blaxhall', 'Tip tree', 'Maldon'],  
 'Tip tree': ['Harwich', 'Feering', 'Clacton', 'Maldon'],  
 'Clacton': ['Harwich', 'Tip tree', 'Maldon'],  
 'Maldon': ['Feering', 'Clacton', 'Tip tree'],  
 }  
  
 def DLS(self, start, end, Depth):  
  
 if start == end: return True  
  
 if Depth <= 0: return False  
  
 for i in self.graph[start]:  
 if self.DLS(i, end, Depth - 1):  
 return True  
 return False  
  
 def show(self, start, end, Depth):  
 for i in range(Depth):  
 if self.DLS(start, end, i):  
 return True  
 return False  
  
g = dfs()  
select = str(input('Enter the name of city witch you want to reach : '))  
print('\nPath : ',end=' ')  
  
Depth = 2  
  
if g.show('Dunwich', select, Depth):  
  
 print("Target is reachable from source within max depth")  
else:  
 print("Target is NOT reachable from source within max depth")

**OUTPUT:**

Text

Description automatically generated

Text

Description automatically generated

1. **iterative deepening search**

**CODE:**

graph = {  
 'Dunwich': ['Blaxhall', 'Harwich'],  
 'Blaxhall': ['Dunwich', 'Harwich', 'Feering'],  
 'Harwich': ['Dunwich', 'Blaxhall', 'Tip tree', 'Clacton'],  
 'Feering': ['Blaxhall', 'Tip tree', 'Maldon'],  
 'Tip tree': ['Harwich', 'Feering', 'Clacton', 'Maldon'],  
 'Clacton': ['Harwich', 'Tip tree', 'Maldon'],  
 'Maldon': ['Feering', 'Clacton', 'Tip tree'],  
 }  
  
  
def IDDFS(root, goal):  
 depth = 0  
 while True:  
 print ("Looping at depth ", depth)  
 result = DLS(root, goal, depth)  
 print ('Result: ',result, ' Goal: ', goal)  
 print('\n')  
 if result == goal:  
 return result  
 depth = depth +1  
  
def DLS(node, goal, depth):  
 print ('Node :',node ,' Goal :', goal ,' Depth :' ,depth)  
 if depth == 0 and node == goal:  
 print( "\n --- Found goal, returning --- ")  
 return node  
 elif depth > 0:  
 print ("Looping through children : " ,graph.get(node, []))  
 for child in graph.get(node, []):  
 if goal == DLS(child, goal, depth-1):  
 return goal  
  
  
print(IDDFS('Dunwich','Maldon'))

**OUTPUT:**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Note: *Implement Depth first search using recursive method*

*For depth limited search, you can limit the level by your own choice*

*Extra marks will be for the students who implement graphical user interface as  well*

Graphical user interface, application

Description automatically generated