|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Finolex Academy of Management and Technology, Ratnagiri** | | | | | | | | | |
| **Department of Information Technology** | | | | | | | | | |
| Subject name: Intelligent Systems | | | | | | | | Subject Code: BEITC703 | | | |
| Class | | BE IT | | Semester – VII (CBGS) | | | | Academic year: 2018-19 | | | |
| Name of Student | |  | | | | | **QUIZ Score :** | | | | |
| Roll No | |  | | | Assignment/Experiment No. | | | | | 02 | |
| Title:  **Implementing state space search using 1. BFS. , 2. DFS ( Un-Informed Search)** | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **1. Course objectives applicable:**  **COB2**.Understand the different searching techniques to solve the different AI problems on the basis of knowledge of agent | | | | | | | | | | | |
| **2. Course outcomes applicable:**  **CO2** –Solve the problems based on searching techniques. | | | | | | | | | | | |
| **3. Learning Objectives:**   1. To understand different searching techniques to solve AI problems 2. To understand concept of BFS, DFS 3. To get the output which will give the correct rule applied for required steps | | | | | | | | | | | |
| **4. Practical applications of the assignment/experiment:** Euler circuits,Bioconnected graphs | | | | | | | | | | | |
| **5. Prerequisites**:   1. To learn the use of intelligent agents in uninformed search. 2. To understand the programming methodology for water jug problem. | | | | | | | | | | | |
| **6. Hardware Requirements**:   1. PC with minimum 2GB RAM   **7. Software Requirements:**  1. Windows installed  2. JDK/Net beans | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **8. Quiz Questions (if any): (Online Exam will be taken separately batchwise, attach the certificate/ Marks obtained)**   1. What is AI? 2. What is an Agent? 3. What is PEAS? 4. What is not an Uninformed search technique? | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **9. Experiment/Assignment Evaluation:** | | | | | | | | | | | |
| **Sr. No.** | **Parameters** | | | | | | | | **Marks obtained** | | **Out of** |
| **1** | Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used - | | | | | | | |  | | 6 |
| **2** | Neatness/presentation | | | | | | | |  | | 2 |
| **3** | Punctuality | | | | | | | |  | | 2 |
| **Date of performance (DOP)** | | |  | | | **Total marks obtained** | | |  | | **10** |
| **Date of checking (DOC)** | | |  | | | **Signature of teacher** | | | | | |

**13. Observations**

1. The output will give the sequence of nodes to find the solution path or goal node.

**14. Program**

**Breadth First Search**

from collections import defaultdict

class Graph:

def \_\_init\_\_(self):

self.graph = defaultdict(list)

def addEdge(self,u,v):

self.graph[u].append(v)

def BFS(self, s):

visited = [False] \* (len(self.graph))

queue = []

queue.append(s)

visited[s] = True

while queue:

s = queue.pop(0)

print (s, end = " ")

for i in self.graph[s]:

if visited[i] == False:

queue.append(i)

visited[i] = True

g = Graph()

g.addEdge(0, 1)

g.addEdge(0, 2)

g.addEdge(1, 2)

g.addEdge(2, 0)

g.addEdge(2, 3)

g.addEdge(3, 3)

print ("Following is Breadth First Traversal"

" (starting from vertex 2)")

g.BFS(2)

**Depth First Search**

from collections import defaultdict

class Graph:

def \_\_init\_\_(self):

# default dictionary to store graph

self.graph = defaultdict(list)

# function to add an edge to graph

def addEdge(self,u,v):

self.graph[u].append(v)

# Function to print a DFS of graph

def DFS(self, s):

# Mark all the vertices as not visited

visited = [False] \* (len(self.graph))

# Create a queue for BFS

queue = []

# Mark the source node as

# visited and enqueue it

queue.append(s)

visited[s] = True

while queue:

# Dequeue a vertex from queue and print it

s = queue.pop(0)

print (s, end = " ")

# Get all adjacent vertices of the dequeued vertex s. If a adjacent has

# not been visited, then mark it visited and enqueue it

for i in self.graph[s]:

if visited[i] == False:

queue.append(i)

visited[i] = True

# Driver code

# Create a graph given in

# the above diagram

g = Graph()

g.addEdge(0, 1)

g.addEdge(0, 2)

g.addEdge(1, 2)

g.addEdge(2, 0)

g.addEdge(2, 3)

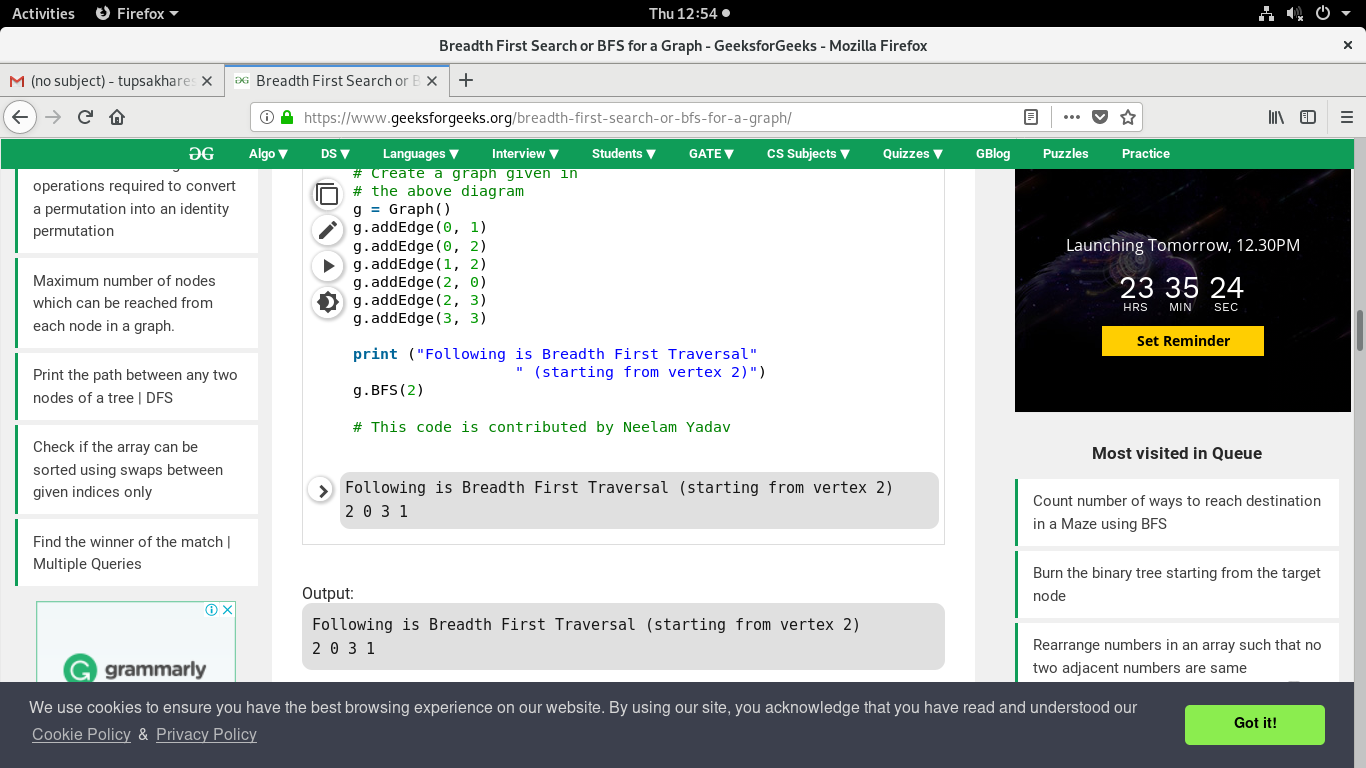
g.addEdge(3, 3)

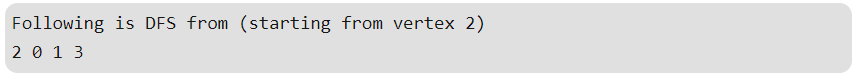
print ("Following is Breadth First Traversal"

" (starting from vertex 2)")

g.DFS(2)

**15. Result:**





**16. Learning Outcomes Achieved**

1. Understanding the concept of uninformed search.
2. Understanding the water jug problem solved by uninformed search technique.

**17. Conclusion:**

1. **Applications of the studied technique in industry**
   1. DFS and BFS algorithms used to develop intelligent systems which solve mathematical problems based on graph.
2. **Engineering Relevance** 
   1. Such algorithms are very useful in searching techniques where number of solutions are more than one.
3. **Skills Developed**
   1. Implementation of BFS for water jug problem.

**18. References** :

[1] G. Görz, C.-R. Rollinger, J. Schneeberger (Hrsg.) “Handbuch der künstlichen

Intelligenz” Oldenbourg Verlag, 2003, Fourth edition

• [2] Turing, A. "Computing Machinery and Intelligence", Mind LIX (236): 433–460,

Ocotober, 1950.

• [3] Aristotle “On Interpretation”, 350 B.C.E, see:

http://classics.mit.edu/Aristotle/interpretation.html

• [4] Newell, A., Simon, H.A. “Human Problem Solving” Englewood Cliffs, N.J.: Prentice

Hall, 1972

• [5] Newell, A. “The Knowledge Level”, AI Magazine 2 (2), 1981, p. 1-20.