**LAB FILE**

**ARTIFICIAL INTELLIGENCE**

**[CSE401]**

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING



**Submitted To: Submitted By:**

 Dr. S. K. Dubey Shefali Bansal Associate Professor A2305218263 CSE Department, ASET B. Tech (CSE)

7CSE4Y

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY UTTAR PRADESH

NOIDA-201301

**INDEX**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Experiment No.** | **Category**  **of Assignment** | **Code** | **Name of Experiment** | **Date of Allotment of experiment** | **Date of Evaluation** | **Max.**  **Marks** | **Marks obtained** | **Sign.**  **of Faculty** |
|  | **Mandatory Experiment** | **LR (0)** | Write a program to implement A\* algorithm in python | 23-07-21 | 30-07-21 | **1** |  |  |
|  | **Mandatory Experiment** |  | Write a program to implement Single Player Game | 30-07-21 | 13-08-21 | **1** |  |  |
|  | **Mandatory Experiment** |  | Write a program to implement Tic-Tac-Toe game problem | 13-08-21 | 27-08-21 | **1** |  |  |
|  | **Mandatory Experiment** |  | Implement Brute force solution to the Knapsack problem in Python | 27-08-21 | 03-09-21 | **1** |  |  |
|  | **Mandatory Experiment** |  | Implement Graph coloring problem using python | 03-09-21 | 09-09-21 | **1** |  |  |
|  | **Mandatory Experiment** |  | Write a program to implement BFS for water jug problem using Python | 09-09-21 | 17-09-21 | **1** |  |  |
|  | **Mandatory Experiment** |  | Write a program to implement DFS using Python | 10-09-21 | 17-09-21 | **1** |  |  |
|  | **Mandatory Experiment** |  | Tokenization of word and Sentences with the help of NLTK package | 17-09-21 | 24-09-21 | **1** |  |  |
|  | **Mandatory Experiment** |  | Design an XOR truth table using Python | 24-09-21 | 01-10-21 | **1** |  |  |
|  | **Mandatory Experiment** |  | Study of SCIKIT fuzzy | 01-10-21 | 22-10-21 | **1** |  |  |

**EXPERIMENT 1**

**DATE:** 23/07/2021

**AIM:** Write a program to implement the A\* algorithm in python.

**SOFTWARE USED:** Python

**THEORY:**

A \* algorithm is a searching algorithm that searches for the shortest path between the initial and the final state. It is used in various applications, such as maps. In maps the A\* algorithm is used to calculate the shortest distance between the source (initial state) and the destination (final state).

A\* (pronounced "A-star") is a [graph traversal](https://en.wikipedia.org/wiki/Graph_traversal) and [path search](https://en.wikipedia.org/wiki/Pathfinding) [algorithm](https://en.wikipedia.org/wiki/Algorithm), which is often used in many fields of computer science due to its completeness, optimality, and optimal efficiency.One major practical drawback is its O(b^d) space complexity, as it stores all generated nodes in memory. Thus, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, as well as memory-bounded approaches; however, A\* is still the best solution in many cases.

**CODE:**

def aStarAlgo(start\_node, stop\_node):

sum1=1

open\_set = set(start\_node)

closed\_set = set()

g = {}

parents = {}

g[start\_node] = 0

parents[start\_node] = start\_node

while len(open\_set) > 0:

n = None

for v in open\_set:

if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):

n = v

if n == stop\_node or Graph\_nodes[n] == None:

pass

else:

for (m, weight) in get\_neighbors(n):

if m not in open\_set and m not in closed\_set:

open\_set.add(m)

parents[m] = n

g[m] = g[n] + weight

else:

if g[m] > g[n] + weight:

g[m] = g[n] + weight

sum1 = sum1 + g[m]

parents[m] = n

if m in closed\_set:

closed\_set.remove(m)

open\_set.add(m)

if n == None:

print('Path does not exist!')

return None

if n == stop\_node:

path = []

while parents[n] != n:

path.append(n)

n = parents[n]

path.append(start\_node)

path.reverse()

print('Shortest Path found for graph: {}'.format(path))

print('Cost: '+str(sum1))

return path

open\_set.remove(n)

closed\_set.add(n)

print('Shortest Path does not exist!')

return None

def get\_neighbors(v):

if v in Graph\_nodes:

return Graph\_nodes[v]

else:

return None

def heuristic(n):

H\_dist = {'A': 10, 'B': 8, 'C': 5, 'D': 7, 'E': 3, 'F': 6, 'G': 5, 'H': 3, 'I': 1, 'J': 0}

return H\_dist[n]

Graph\_nodes = {

'A': [('B', 6), ('F', 3)],

'B': [('D', 2),('C', 3)],

'C': [('B', 3),('D', 1),('E',5)],

'D': [('B', 2),('C',1),('E',8)],

'E': [('C', 5),('D',8),('I',5),('J',5)],

'F': [('A',3),('G',1), ('H',7)],

'G': [('F',1),('I',3)],

'H': [('F',7),('I',2)],

'I': [('G',3), ('E',5),('J',3),('H',2)],

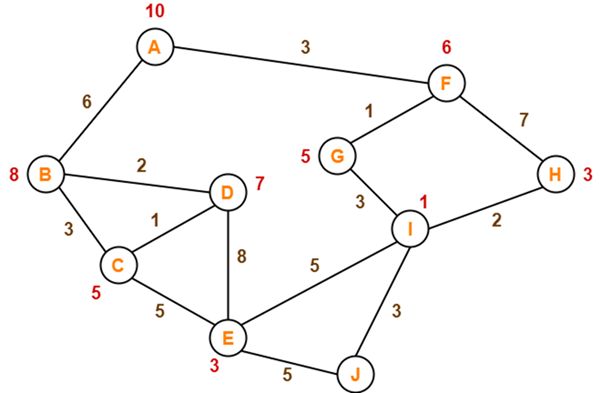
'J': [('E',5),('I',3)]

}

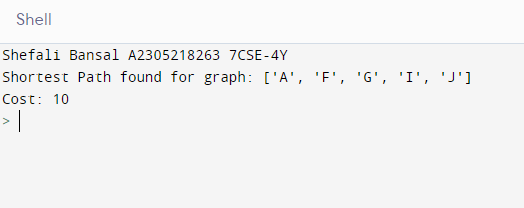
print("Shefali Bansal A2305218263 7CSE-4Y")

aStarAlgo('A', 'J')

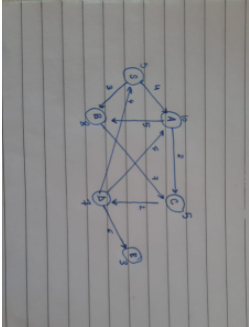
**INPUT:**

****

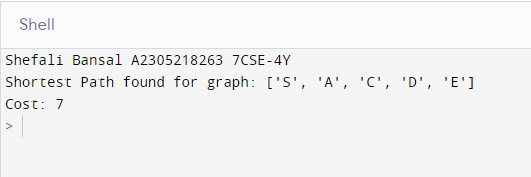
**OUTPUT:**

****

**INPUT:**

****

**OUTPUT:**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

**EXPERIMENT 2**

**DATE:** 30/07/2021

**AIM:** Write a program to implement Single Player Game.

**SOFTWARE USED:** Python

**THEORY:**

A singleplayer video game is a video game where input from only one player is expected throughout the course of the gaming session. Here, the created game is snake, weasel, and tiger. The PC is prompted to store a random choice and the user is asked to enter their final choice as well.

User input:

1. Snake
2. Weasel
3. Tiger

Rules:

* It’s a draw if user input matches random choice generation done by the computer.
* If choices made are weasel and snake, the one choosing weasel wins.
* If choices made are weasel and tiger, the one choosing tiger wins.
* If choices made are tiger and snake, the one choosing snake wins.

**CODE:**

print('Shefali Bansal A2305218263 7CSE-4Y\n')

import random

options = ["snake","weasel","tiger"]

pc = random.choice(options)

while True:

user = input("\nPick either snake, weasel or tiger: ")

print('PC chose: ', pc)

print('User chose: ', user)

if user == 'snake':

if pc == 'snake':

print('Tie Game')

elif pc == 'weasel':

print('PC Wins')

else:

print ('User Wins')

if user == 'weasel':

if pc == 'snake':

print ('User Wins')

elif pc == 'weasel':

print ('Tie Game')

else:

print ('PC Wins')

if user == 'tiger':

if pc == 'snake':

print ('PC Wins')

elif pc == 'weasel':

print ('User Wins')

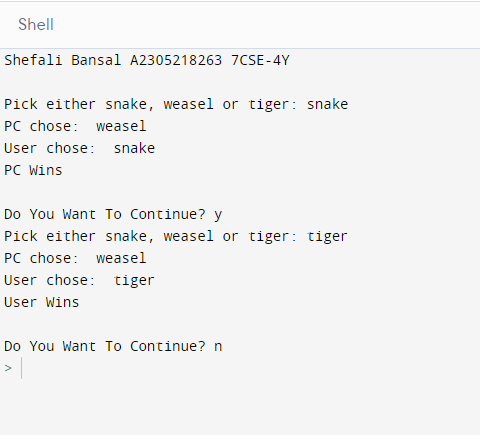
else:

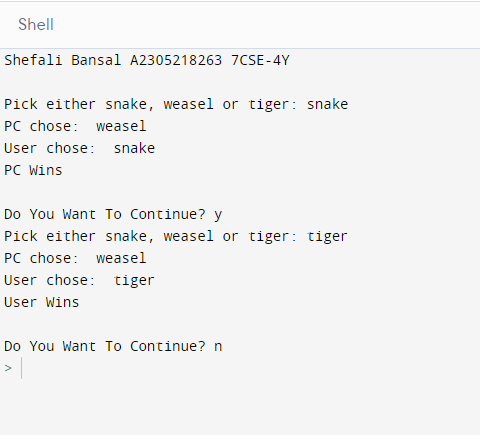
print ('Tie Game')

if input('\nDo You Want To Continue? ') not in ['y','Y']:

break

**OUTPUT:**





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

**EXPERIMENT 3**

**DATE:** 13/08/2021

**AIM:** Write a program to implement a Tic-Tac-Toe game problem.

**SOFTWARE USED:** Python

**THEORY:**

Tic-tac-toe, noughts and crosses, or Xs and Os is a paper-and-pencil game for two players who take turns marking the spaces in a three-by-three grid with X or O. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner. It is a [solved game](https://en.wikipedia.org/wiki/Solved_game), with a forced draw assuming [best play](https://en.wikipedia.org/wiki/Best_response) from both players. Tic-tac-toe is played on a three-by-three grid by two players, who alternately place the marks X and O in one of the nine spaces in the grid. Players soon discover that the best play from both parties leads to a [draw](https://en.wikipedia.org/wiki/Draw_(tie)). Hence, tic-tac-toe is often played by young children who may not have discovered the optimal strategy.

Because of the simplicity of tic-tac-toe, it is often used as a [pedagogical](https://en.wikipedia.org/wiki/Pedagogical) tool for teaching the concepts of good [sportsmanship](https://en.wikipedia.org/wiki/Sportsmanship) and the branch of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) that deals with the searching of [game trees](https://en.wikipedia.org/wiki/Game_tree). It is straightforward to write a [computer program](https://en.wikipedia.org/wiki/Computer_program) to play tic-tac-toe perfectly or to enumerate the 765 essentially different positions (the [state space complexity](https://en.wikipedia.org/wiki/State_space_complexity)) or the 26,830 possible games [up to](https://en.wikipedia.org/wiki/Up_to) rotations and reflections (the [game tree complexity](https://en.wikipedia.org/wiki/Game_tree_complexity)) on this space.If played optimally by both players, the game always ends in a draw, making tic-tac-toe a [futile game](https://en.wikipedia.org/wiki/Futile_game). The game can be generalized to an [*m*, *n*, *k*-game](https://en.wikipedia.org/wiki/M,n,k-game), in which two players alternate placing stones of their own color on an *m*-by-*n* board with the goal of getting *k* of their own color in a row. Tic-tac-toe is the 3, 3, 3-game.

**CODE:**

theBoard = {'7': ' ' , '8': ' ' , '9': ' ' ,

'4': ' ' , '5': ' ' , '6': ' ' ,

'1': ' ' , '2': ' ' , '3': ' ' }

board\_keys = []

for key in theBoard:

board\_keys.append(key)

def printBoard(board):

print(board['7'] + '|' + board['8'] + '|' + board['9'])

print('-+-+-')

print(board['4'] + '|' + board['5'] + '|' + board['6'])

print('-+-+-')

print(board['1'] + '|' + board['2'] + '|' + board['3'])

def game():

turn = 'X'

count = 0

print("\n")

for i in range(10):

printBoard(theBoard)

move = input("\nTurn:" + turn + "\tMove: ")

print("")

if theBoard[move] == ' ':

theBoard[move] = turn

count += 1

else:

print("Place already filled.\nMove: ")

continue

if count >= 5:

if theBoard['7'] == theBoard['8'] == theBoard['9'] != ' ':

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['4'] == theBoard['5'] == theBoard['6'] != ' ':

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['1'] == theBoard['2'] == theBoard['3'] != ' ':

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['1'] == theBoard['4'] == theBoard['7'] != ' ':

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['2'] == theBoard['5'] == theBoard['8'] != ' ':

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['3'] == theBoard['6'] == theBoard['9'] != ' ':

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['7'] == theBoard['5'] == theBoard['3'] != ' ':

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['1'] == theBoard['5'] == theBoard['9'] != ' ':

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

if count == 9:

print("\nGame Over.\n")

print("It's a Tie!!")

if turn =='X':

turn = 'O'

else:

turn = 'X'

restart = input("Do want to play Again?(y/n)")

if restart == "y" or restart == "Y":

for key in board\_keys:

theBoard[key] = " "

game()

if \_\_name\_\_ == "\_\_main\_\_":

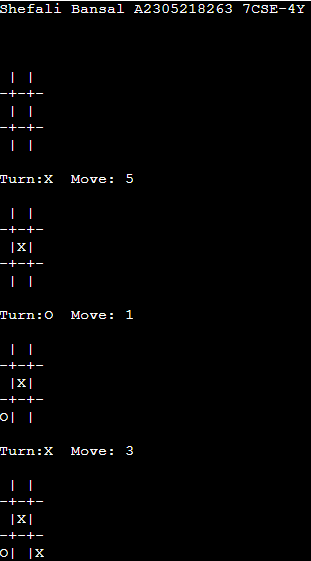
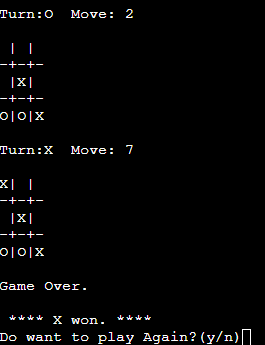
print('Shefali Bansal A2305218263 7CSE-4Y\n')

game()

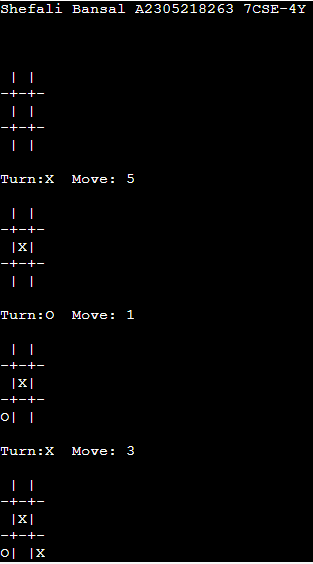
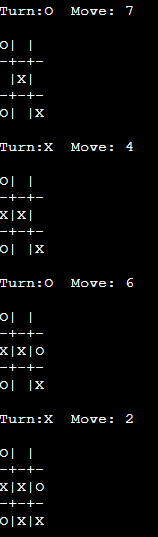
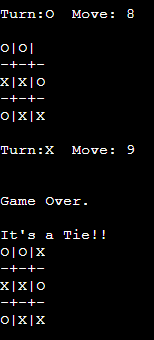
**OUTPUT:**

**PLAYER X PLAYER:**

*Win:*

***Draw***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

**EXPERIMENT 4**

**DATE:** 27/08/2021

**AIM:** Implement the Brute force solution to the Knapsack problem in Python.

**SOFTWARE USED:** Python

**THEORY:**

The knapsack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item included in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. The problem often arises in resource allocation where the decision makers have to choose from a set of non-divisible projects or tasks under a fixed budget or time constraint, respectively. The knapsack problems have a variety of real-life applications including financial modeling, production and inventory management systems, stratified sampling, design of queuing network models in manufacturing, and control of traffic overload in telecommunication systems.

**CODE:**

print('Shefali Bansal A2305218263 7CSE-4Y\n')

def knapSack(W, wt, val, n):

# Base Case

if n == 0 or W == 0:

return 0

# If weight of the nth item > W, exclude.

if (wt[n-1] > W):

return knapSack(W, wt, val, n-1)

# return the maximum of two cases:

# (1) nth item included

# (2) not included

else:

inc = val[n-1] + knapSack( W-wt[n-1], wt, val, n-1)

exc = knapSack(W, wt, val, n-1)

print("Profit including item",n,":",inc,sep=" ")

print("Profit excluding item",n,":",exc,sep=" ")

return max( inc, exc )

n=int(input("Enter no. of items: "))

wt=[]

val=[]

for i in range(n):

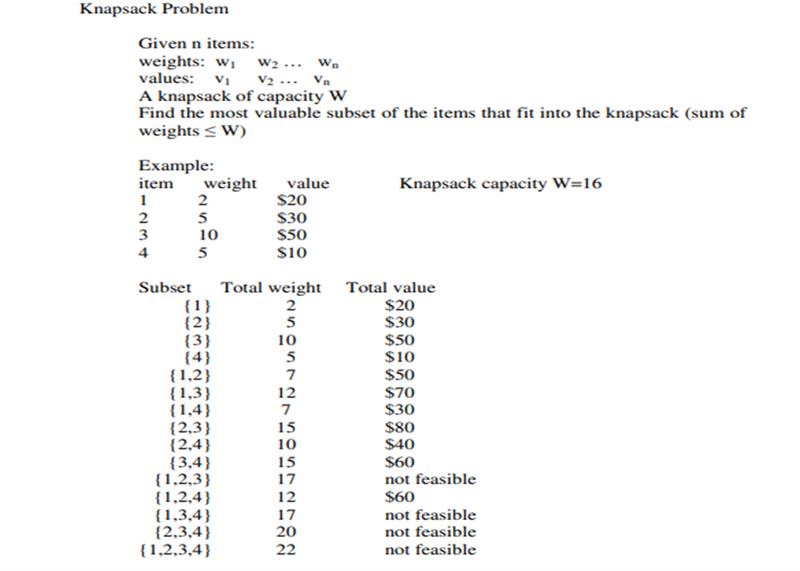
wt.append(int(input("Item "+str(i+1)+" weight: ")))

val.append(int(input("Item "+str(i+1)+" value: ")))

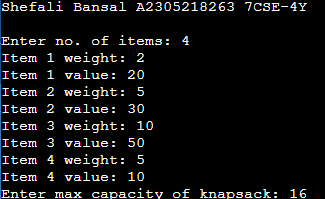
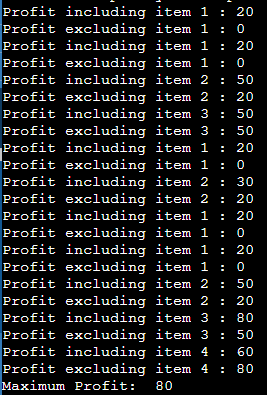
W=int(input("Enter max capacity of knapsack: "))

print("Maximum Profit: ", knapSack(W, wt, val, n))

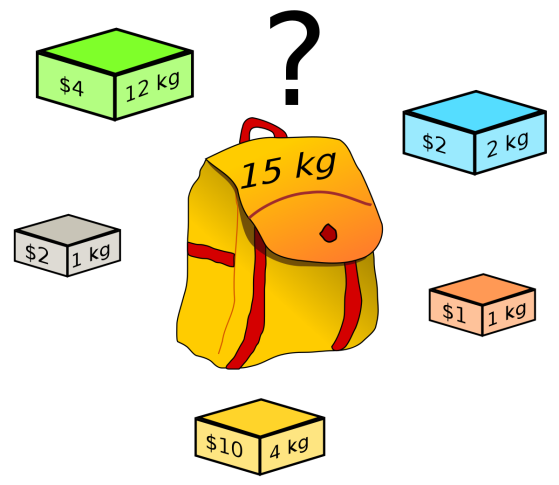
**INPUT:**



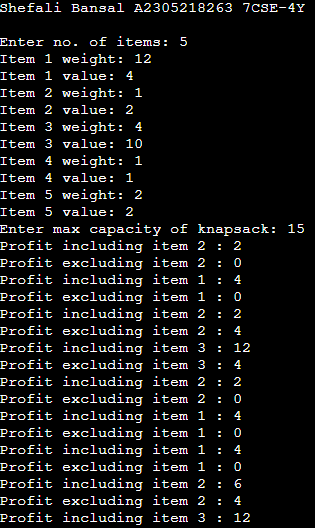
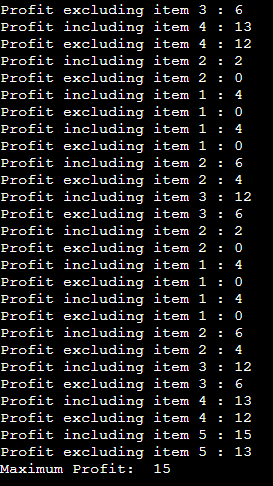
**OUTPUT:**

**INPUT:**



**OUTPUT:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

**EXPERIMENT 5**

**DATE:** 03/09/2021

**AIM:** Implement the Graph coloring problem using python.

**SOFTWARE USED:** Python

**THEORY:**

Graph coloring problem is to assign colors to certain elements of a graph subject to certain constraints. Vertex coloring is the most common graph coloring problem. The problem is, given m colors, find a way of coloring the vertices of a graph such that no two adjacent vertices are colored using the same color.

In [graph theory](https://en.wikipedia.org/wiki/Graph_theory), graph coloring is a special case of [graph labeling](https://en.wikipedia.org/wiki/Graph_labeling); it is an assignment of labels traditionally called "colors" to elements of a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) subject to certain constraints. In its simplest form, it is a way of coloring the [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) of a graph such that no two adjacent vertices are of the same color; this is called a [vertex coloring](https://en.wikipedia.org/wiki/Vertex_coloring). Similarly, an [edge coloring](https://en.wikipedia.org/wiki/Edge_coloring) assigns a color to each edge so that no two adjacent edges are of the same color, and a face coloring of a [planar graph](https://en.wikipedia.org/wiki/Planar_graph) assigns a color to each face or region so that no two faces that share a boundary have the same color.

**CODE:**

print('Shefali Bansal A2305218263 7CSE-4Y\n')

colors = ['Red', 'Blue', 'Green', 'Yellow', 'Black', 'Orange', 'Pink', 'Purple', 'White', 'Brown', ]

nodes = ['a','b','c','d','e','f']

neighbours={}

neighbours['a'] = ['b','c','d']

neighbours['b'] = ['a','c']

neighbours['c'] = ['a','b','d','e','f']

neighbours['d'] = ['a','c','e']

neighbours['e'] = ['c','d','f']

neighbours['f'] = ['c','e']

color\_nodes={}

def function(node,color):

for neighbour in neighbours.get(node):

color\_neighbour = color\_nodes.get(neighbour)

if color\_neighbour == color:

return False

return True

def colorNode(node):

for color in colors:

if function(node, color):

return color

for node in nodes:

color\_nodes[node] = colorNode(node)

print(color\_nodes)

used=[]

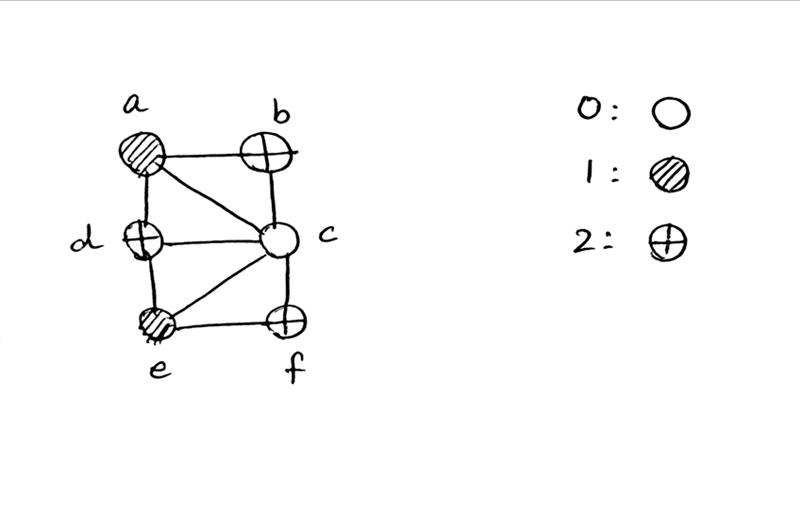
for cn in color\_nodes.values():

if cn not in used:

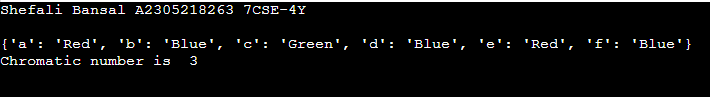
used.append(cn)

print('Chromatic number is ',len(used))

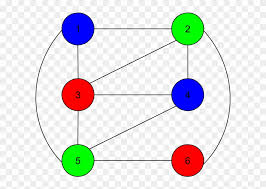
**INPUT:**

****

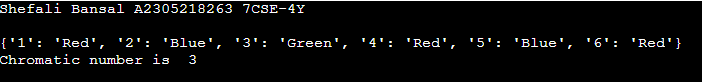
**OUTPUT:**

****

**INPUT:**



**OUTPUT:**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

**EXPERIMENT 6**

**DATE:** 09/09/2021

**AIM:** Write a program to implement BFS for water jug problems using Python.

**SOFTWARE USED:** Python

**THEORY:**

Water pouring puzzles are a class of puzzles involving a finite collection of water jugs of known integer capacities (in terms of a liquid measure such as litres or gallons).

BFS stands for Breadth-First Search is a vertex-based technique for finding the shortest path in the graph. It uses a Queue data structure that follows first in first out. In BFS, one vertex is selected at a time when it is visited and marked then its adjacent are visited and stored in the queue.

Using a GPS navigation system BFS is used to find neighbouring places. In networking, when we want to broadcast some packets, we use the BFS algorithm. The pathfinding algorithm is based on BFS or DFS. BFS is used in the Ford-Fulkerson algorithm to find maximum flow in a network.

**CODE:**

print('Shefali Bansal A2305218263 7CSE-4Y\n')

m=int(input("Capacity of Jug1: "))

n=int(input("Capacity of Jug2: "))

print("Capacities: (",m,",",n,")")

print("Initial state: (0,0)")

x=y=0

print("Goal state: (2,y)")

print("\n")

print("Rules:\n 1. fill jug1 \n 2. fill jug2 \n 3. empty jug1 \n 4. empty jug2 \n 5. pour all from jug2 to jug1 ")

print(" 6. pour all from jug1 to jug2 \n 7. pour from jug2 to jug1 until full \n 8. pour from jug1 to jug2 until full")

while x!=2:

print("\nState: (",x,y,")")

r=int(input("Enter Rule:"))

if r==1:

# fill 4 gallon jug

if x < m:

x=m

if r==2:

# fill 3 gallon jug

if y < n:

y=n

if r==3:

# empty 4 gallon jug

if x > 0:

x=0

if r==4:

# empty 3 gallon jug

if y > 0:

y=0

if r==5:

# pour all from 3 gj to 4 gj

if ((x + y < m ) or (x + y == m)) and (y > 0) :

x+=y

y=0

if r==6:

# pour all from 4 gj to 3 gj

if ((x + y < n ) or (x + y == n)) and (x > 0) :

y+=x

x=0

if r==7:

# pour from 3 gj to 4 gj until full

if ((x + y > m ) or (x + y == m)) and (y > 0) :

y=y-(m-x)

x=m

if r==8:

# pour from 4 gj to 3 gj until full

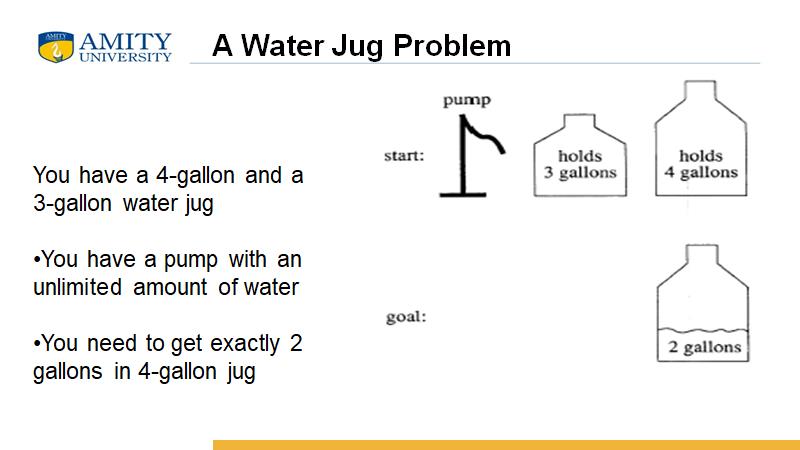
if ((x + y > n ) or (x + y == n)) and (x > 0) :

x=x-(n-y)

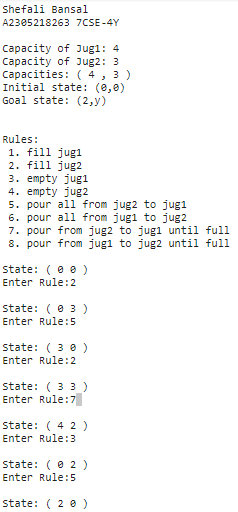
y=n

print("\nState: (",x,y,")")

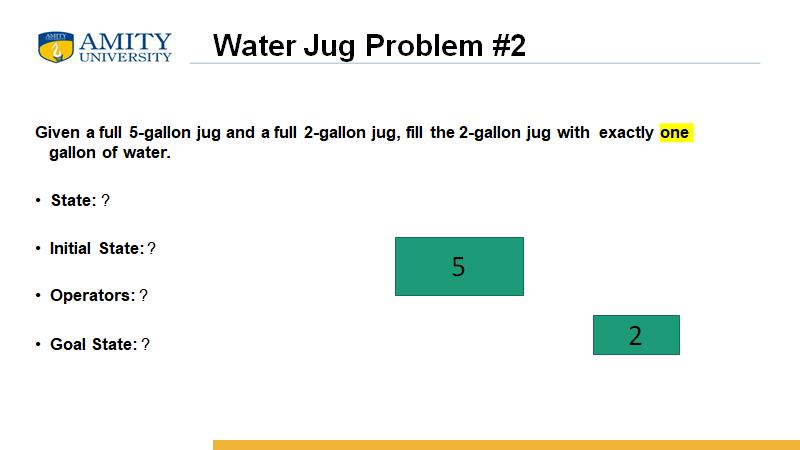
**INPUT:**

****

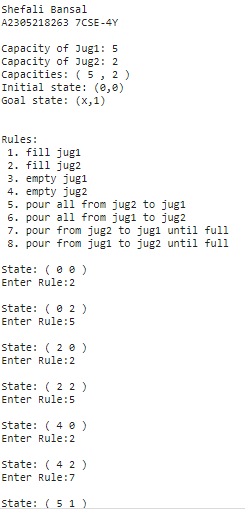
**OUTPUT:**

****

**INPUT:**

****

**OUTPUT:**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

**EXPERIMENT 7**

**DATE:** 10/09/2021

**AIM:** Write a program to implement DFS using Python.

**SOFTWARE USED:** Python

**THEORY:**

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. The time and space analysis of DFS differs according to its application area. In theoretical computer science, DFS is typically used to traverse an entire graph, and takes time, where |V| is the number of vertices and |E| the number of edges. This is linear in the size of the graph. In these applications it also uses space in the worst case to store the stack of vertices on the current search path as well as the set of already-visited vertices. Thus, in this setting, the time and space bounds are the same as for breadth-first search and the choice of which of these two algorithms to use depends less on their complexity and more on the different properties of the vertex orderings the two algorithms produce.

Applications: Depth-first search is used in topological sorting, scheduling problems, cycle detection in graphs, and solving puzzles with only one solution, such as a maze or a Sudoku puzzle. Other applications involve analyzing networks, for example, testing if a graph is bipartite.

**CODE:**

print('Shefali Bansal A2305218263 7CSE-4Y\n')

adjacency\_list = {

'A': ['B', 'C'],

'B': ['D', 'E'],

'C': ['F'],

'D': [],

'E': ['F'],

'F': []

}

class Graph:

def get\_neighbors(self, v):

return adjacency\_list[v]

def traverse(self, v, visited):

visited.add(v)

print(v)

for neighbour in adjacency\_list[v]:

if neighbour not in visited:

self.traverse(neighbour, visited)

def DFS(self, v):

visited = set([])

self.traverse(v, visited)

def getpath(self, start, stop):

visited = set([])

visited.add(start)

path = [start]

while len(path) > 0:

n = path[-1]

if n == stop:

#print('Path found: {}'.format(path))

return path

has\_unvisited = False

for m in self.get\_neighbors(n):

if m not in visited:

path.append(m)

visited.add(m)

has\_unvisited = True

break

if (not has\_unvisited):

path.pop()

print('Path doesnt exist')

return None

g = Graph()

print("\nOrder of DFS Traversal (from node A):")

g.DFS('A')

path = g.getpath('A','F')

print('\nPath found (from A to F): ')

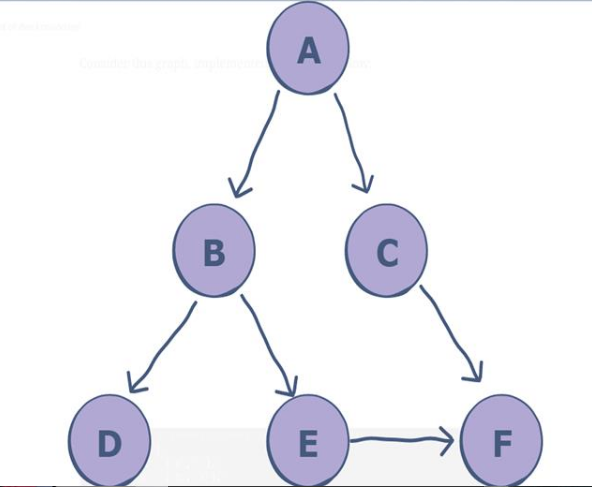
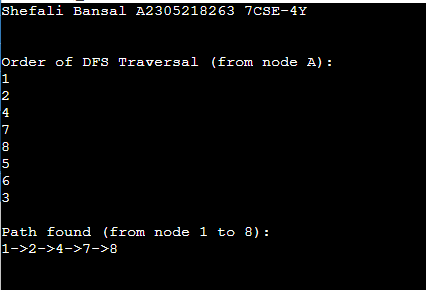
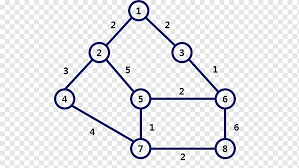
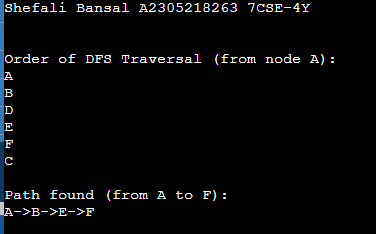
for i in path:

if i != path[-1]:

print(i, end='->')

print(path[-1])

**INPUT AND OUTPUT:**

**** 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

**EXPERIMENT 8**

**DATE:** 17/09/2021

**AIM:** Tokenization of word and Sentences with the help of NLTK package.

**SOFTWARE USED:** Python

**THEORY:**

**Tokenization** is the process by which a large quantity of text is divided into smaller parts called tokens. These tokens are very useful for finding patterns and are considered as a base step for stemming and lemmatization. Tokenization also helps to substitute sensitive data elements with non-sensitive data elements. Tokenization is a common task in Natural Language Processing (NLP). It’s a fundamental step in both traditional NLP methods like Count Vectorizer and Advanced Deep Learning-based architectures like Transformers. Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be words, characters, or sub words. Hence, tokenization can be broadly classified into 3 types – word, character, and sub word (n-gram characters) tokenization. These tokens are often loosely referred to as terms or words, but it is sometimes important to make a type/token distinction. A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing. Natural language processing is used for building applications such as Text classification, [intelligent chatbot](https://www.guru99.com/best-ai-chatbots.html), sentimental analysis, language translation, etc. It becomes vital to understand the pattern in the text to achieve the above-stated purpose.

**CODE:**

import nltk

from nltk.tokenize import word\_tokenize

nltk.download('punkt')

text = "Tokenization is the process by which a large quantity of text is divided into smaller parts called tokens. "+ \

"These tokens are very useful for finding patterns and are considered as a base step for stemming and lemmatization. " + \

"Tokenization also helps to substitute sensitive data elements with non-sensitive data elements. " + \

"Tokenization is a common task in Natural Language Processing (NLP). " + \

"It’s a fundamental step in both traditional NLP methods like Count Vectorizer and Advanced Deep Learning-based architectures like Transformers."+\

"Tokenization is a way of separating a piece of text into smaller units called tokens. "+\

"Here, tokens can be words, characters, or sub words."+\

"Hence, tokenization can be broadly classified into 3 types – word, character, and sub word (n-gram characters) tokenization. "+\

"These tokens are often loosely referred to as terms or words, but it is sometimes important to make a type/token distinction. "+\

"A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing."+\

" Natural language processing is used for building applications such as Text classification, intelligent chatbot, sentimental analysis, language translation, etc. "+\

"It becomes vital to understand the pattern in the text to achieve the above-stated purpose."

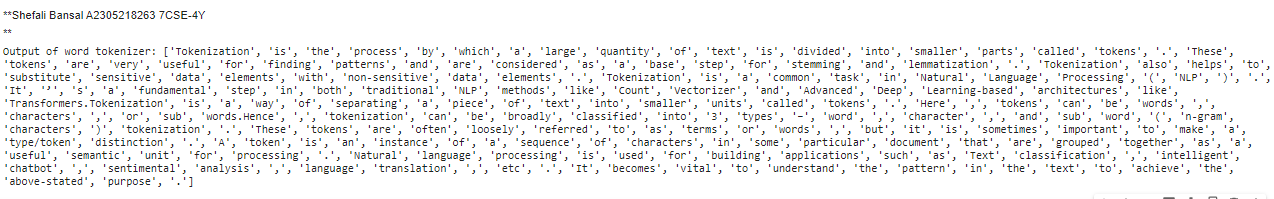
print('Shefali Bansal A2305218263 7CSE-4Y\n')

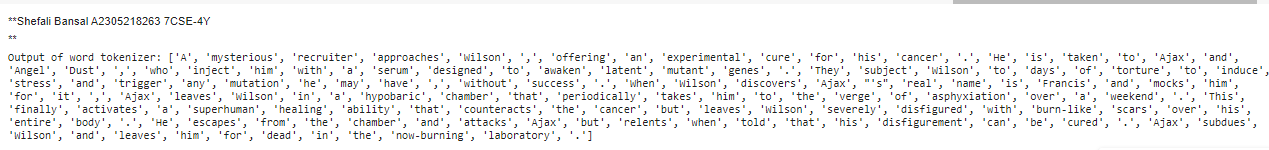
print("\n\nOutput of word tokenizer: \n",word\_tokenize(text))

from nltk.tokenize import sent\_tokenize

print("\n\nOutput of sent tokenizer: \n",sent\_tokenize(text))

**OUTPUT:**

****



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

**EXPERIMENT 9**

**DATE:** 24/09/2021

**AIM:** Design an XOR truth table using Python.

**SOFTWARE USED:** Python

**THEORY:**

Exclusive OR or exclusive disjunction is a logical operation that is true if and only if its arguments differ (one is true, the other is false).It is symbolized by the prefix operator J and by the infix operators XOR EOR, EXOR, ⊻, ⩒, ⩛, ⊕, and ≢. The negation of XOR is the logical bi-conditional, which yields true if and only if the two inputs are the same. It gains the name "exclusive or" because the meaning of "or" is ambiguous when both operands are true; the exclusive or operator excludes that case. This is sometimes thought of as "one or the other but not both". This could be written as "A or B, but not, A and B". Since it is associative, it may be considered to be an n-ary operator which is true if and only if an odd number of arguments are true. That is, a XOR b XOR ... may be treated as XOR (a,b,...). Exclusive disjunction essentially means 'either one, but not both nor none'. In other words, the statement is true if and only if one is true and the other is false. For example, if two horses are racing, then one of the two will win the race, but not both of them.

**CODE:**

***For two inputs:***

print('Shefali Bansal A2305218263 7CSE-4Y\n')

print("|-------------------------------------------------------|")

print("| XOR GATE |")

print("|-------------------------------------------------------|")

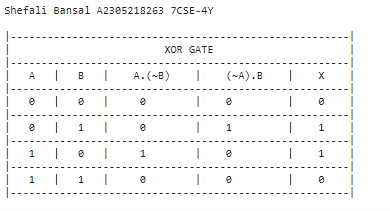
print("| A | B | A.(~B) | (~A).B | X |")

for A in range(0,2):

for B in range(0,2): print("|-------------------------------------------------------|")

print("| ",A," | ",B," | ",A&(~B)," | ",(~A)&B," | ",A^B," |")

print("|-------------------------------------------------------|")



***For three inputs:***

print('Shefali Bansal A2305218263 7CSE-4Y\n')

A=[0,1]

B=[0,1]

C=[0,1]

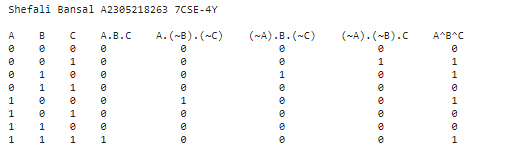
print("A","B","C","A.B.C","A.(~B).(~C)","(~A).B.(~C)","(~A).(~B).C","A^B^C",sep=" ")

for i in range(2):

for j in range(2):

for k in range(2):

print(A[i],B[j],C[k],A[i]&B[j]&C[k],"\t",A[i]&(~B[j])&(~C[k]),"\t",(~A[i])&B[j]&(~C[k]),"\t", (~A[i])&(~B[j])&C[k]," ", A[i]^B[j]^C[k],sep=" ")



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

**EXPERIMENT 10**

**DATE:** 01/10/2021

**AIM:** Study of SCIKIT fuzzy.

**PROBLEM STATEMENT:**

The Tipping Problem: A Measurement of Customer Satisfaction Using the Fuzzy Logic Approach

**ABSTRACT:**

The quality of service is a significant aspect for any customer oriented business. It is necessary to analyze and comprehend the customer’s desires, in order to improve the service quality. The quality service is accomplished only when the services or products meet the customers’ expectations. The ‘tipping problem’ is a good way to demonstrate the usage of fuzzy logic principles [1]. The aim is to produce multifaceted outputs by utilizing brief but intuitive set of conditions. The chief objective is to design a fuzzy control system that facsimiles the process of tipping at a restaurant. Tipping depends entirely on customer satisfaction and factors such as the quality of service and food are rated accordingly.

**INTRODUCTION:**

Satisfaction is an individual’s emotions of pleasure or disappointment that result from paralleling a product’s alleged performance or result to ones’ anticipations. If the performance falls short of expectations, it invokes a feeling of displeasure. If it exceeds the expectations, it invokes a feeling of delight [2]. Due to the gruesome business competition, each service provider must provide optimal services so as to increase customer satisfaction and inspire customer loyalty. In a restaurant, after receiving service and food, customers usually leave tips that reflect the restaurant's service and food quality. Measure of food quality depend appearance, scent, temperature, consistency, flavor and other idiosyncratic and independent attributes. The measure of service depends on tangible elements like customization, approachability, communication, understanding the customer, safety, and price [2]. Food quality and service quality both are very important for restaurant business. Furthermore, consumers who is highly satisfied with the service may left tips a little amount or consumer who is somehow satisfied with the service may left tips a considerably big amount. Amount of tips sometimes may measure the customer satisfaction.

**ARCHITECTURE OF THE MODEL:**

The antecedents involved are ‘service quality’ and ‘food quality’ and the consequents inferred is ‘tip’ which will be produced by the fuzzy logic system. In case of output variable, i.e. tip or customer satisfaction, the value of output is in the range of the unit interval [0, 1] [3].

**Input:**

**Output:**

**Rules:**

The customer satisfaction is contingent on the amount of customer tips received from the customer after they have partaken all the restaurant has to offer. Hence, overall satisfaction, implying the tip received, is entirely dependent upon the satisfaction function of service and food quality of a restaurant.

Therefore satisfaction can be expressed in the form of tips by [4]:

Use case: Satisfaction (Excellent, Delicious) → High

At a restaurant, the service quality is determined by an assortment of attributes and features. The states of satisfaction in consideration that are related with service quality are poor, good, and excellent. The state of service quality is to be determined [4]:

Use case: Service quality (e.g. Competent staff, nicely furnished) → Excellent

At a restaurant, the food quality is determined by an assortment of attributes and features. The states of satisfaction in consideration that are related with service quality are bad, decent, and delicious. The state of service quality is to be determined [4]:

Use Case: Food quality (Appealing, Delicious and Reasonably Priced) → Delicious

**EXPERIMENT AND RESULT:**

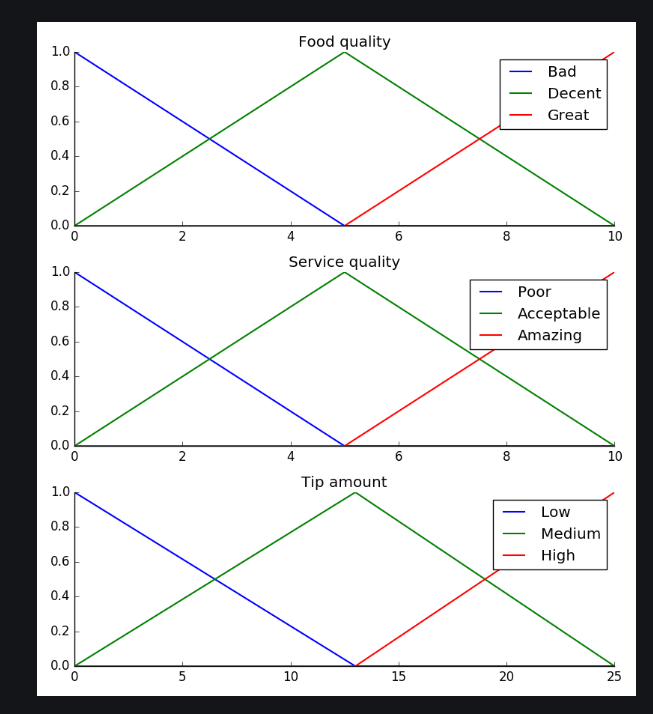


Fig. 1: Input and Output variables represented using Triangular type membership functions [3].

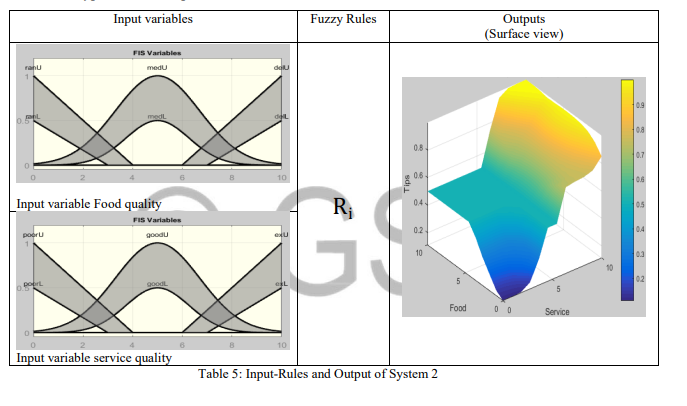


Fig. 2: A 'Sugeno' Type IT2FLS is constructed by taking a combination of Triangular and Gaussian type membership functions [1].

**OUTCOME:**

The Fuzzy systems permit complex and intuitive behavior based on a sparse system of rules with minimal overhead. Fuzzy logic systems are found in applies for measuring the service quality of different service providing enterprise. The results indicate that linguistic assessment is a feasible and meaningful method for determining consumer satisfaction. We can determine the service quality using Fuzzy by determining the difference between customer expectations and consumer perceptions, determining the degree of importance between consumer expectations and consumer perceptions, determining the level of service quality, and recommending improvements dependent on priority by examining the gap of each quality parameter.

**REFERENCES:**

[1]<https://pythonhosted.org/scikit-> fuzzy/auto\_examples/plot\_tipping\_problem\_newapi.html#example-plot-tipping-problem-newapi-py

[2] Chalis Fajri Hasibuan, IOP Conference Series: Materials Science and Engineering, Volume 909, International Conference on Advanced Mechanical and Industrial engineering 8-9 July 2020, Banten, Indonesia

[3] https://pythonhosted.org/scikitfuzzy/auto\_examples/plot\_tipping\_problem.html#example-plot-tipping-problem-py

[4] Shahjalal, Muhammad & Saikh, Shahjahan & Miah, Saikh. (2021). Measuring consumer satisfaction to restaurant service: Type -2 Fuzzy Logic Approach.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP) | | | | |
| Programme | B. Tech CSE | | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | | Semester | 7 |
| Student Name | Shefali Bansal | | Enrollment No. | A2305218263 |
| Marking Criteria | | | | |
| Criteria | Total Marks | Marks Obtained | | Comments |
| Concept (A) | 2 |  | |  |
| Implementation (B) | 2 |  | |  |
| Performance (C) | 2 |  | |  |
| Total | 6 |  | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Internal Assessment (Viva Component) Sheet for Lab Experiment Department of Computer Science & Engineering Amity University, Noida (UP)** | | | |
| Programme | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE401] | Semester | 7 |
| Student Name | Shefali Bansal | Enrollment No. | A2305218263 |
| **Marking Criteria** | | | |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Clarity of the Subject (H) | 4 |  |  |
| Quality of theoretical Discussion (I) | 6 |  |  |
| Total | 10 |  |  |